Universidad Nacional Federico Villarreal

VRIN | VICERRECTORADO DE INVESTIGACIÓN

ESCUELA UNIVERSITARIA DE POSGRADO

SISTEMA DE PROGRAMACIÓN DE ÓRDENES EN AMBIENTE INDUSTRIAL JOBSHOP-ENSAMBLE UTILIZANDO ALGORITMOS HEURÍSTICOS Y RECURSIVOS PARA MEJORAR LA GESTIÓN INDUSTRIAL

Línea de investigación:

Ingeniería de software, simulación y desarrollo de TICs

Tesis para optar el grado académico de Maestro en Ingeniería de Sistemas

Autor:

Avendaño Muñoz, Rooll

Asesor:

Soto Soto, Luis

(ORCID: 0000-0002-3799-645X)

Jurado:

Mujica Ruiz, Oscar Hugo

Tello Malpartida, Omart Demetrio

Pumaricra Padilla, Raul Valentin

Lima - Perú

2023

Reporte de Análisis de Similitud

Archivo:	1A_AVENDAÑO_MUÑOZ_ROOLL_MAESTRÍA_2022.docx
Fecha del Análisis:	12/10/2022
Analizado por:	Astete Llerena, Johnny Tomas
Correo del analista:	jastete@unfv.edu.pe
Porcentaje:	1 %
Título:	SISTEMA DE PROGRAMACION DE ORDENES EN AMBIENTE INDUSTRIAL JOBSHOP-ENSAMBLE UTILIZANDO ALGORITMOS HEURISTICOS Y RECURSIVOS PARA MEJORAR LA GESTION INDUSTRIAL
Enlace:	https://secure.urkund.com/old/view/139499329-705676-914159#q1bKLVayijbQMQQiQx1DYx0jHSMgMtEx1bGM1VEqzkzPy0zLTE7MS05VsjLQMzCwNDQ0tjA3MTS3MDewMDE0qQUA

UNIVERSIDAD NACIONAL FEDERICO VILLARREAL
OFICINA DE GRADOS Y GESTIÓN DEL EGRESADO
ESCUELA UNIVERSITARIA DE POSGRADO

DRA. MIRIAM LILIANA FLORES CORONADO
JEFA DE GRADOS Y GESTIÓN DEL EGRESADO

Universidad Nacional
Federico Villarreal

VRIN | VICERRECTORADO DE INVESTIGACIÓN

ESCUELA UNIVERSITARIA DE POSGRADO

SISTEMA DE PROGRAMACIÓN DE ÓRDENES EN AMBIENTE INDUSTRIAL JOBSHOP-ENSAMBLE UTILIZANDO ALGORITMOS HEURÍSTICOS Y RECURSIVOS PARA MEJORAR LA GESTIÓN INDUSTRIAL

Línea de Investigación:

Ingeniería de software, simulación y desarrollo de TICs.

Tesis para optar el Grado Académico de

Maestro en Ingeniería de Sistemas

Autor:

Avendaño Muñoz, Rooll

Asesor:

Soto Soto, Luis

(ORCID: 0000-0002-3799-645X)

Jurado:

Mujica Ruiz, Oscar Hugo

Tello Malpartida, Omart Demetrio

Pumaricra Padilla, Raul Valentin

Lima - Perú

2023

Dedicatoria

A mi padre Elfer, que me inculcó valores de amor a los libros, a la ciencia y a la cultura. Que me enseñó a siempre cultivar la mente y a adquirir nuevos conocimientos para ser una mejor persona. Continúo tu gran legado y siempre estaremos juntos en este camino papá. Y a mi madre Marcelina por su apoyo incondicional en mi carrera.

ÍNDICE

RESUMEN **7**
ABSTRACT **8**
I INTRODUCCIÓN **9**
1.1 Planteamiento del Problema **11**
1.2 Descripción del Problema **13**
1.3 Formulación del Problema **14**
1.3.1 Problema General 14
1.3.2 Problemas Específicos. 14
1.4 Antecedentes **14**
1.4.1 Antecedentes Nacionales 14
1.4.2 Antecedentes Internacionales 18
1.5 Justificación de la Investigación. **21**
1.6 Limitaciones de la Investigación. **23**
1.7 Objetivos. **23**
1.7.1 Objetivo General 23
1.7.2 Objetivos Específicos. 23
1.8 Hipótesis **24**
1.8.1 Hipótesis General 24
1.8.2 Hipótesis Especifica 24
II MARCO TEORICO **25**
2.1 Job Shop-Ensamble y Heurística **25**
2.1.1 Job Shop-Ensamble 25
2.1.2 Introducción a los Algoritmos. 31
2.1.3 Algoritmos y Problemas. 41
2.1.4 Algoritmos Heurísticos. 45
2.1.5 Problema de la programación en JobShop, motivo de la presente investigación. 47
2.2 Lenguajes de programación Java y Gestor de Base de Dato Oracle **62**
2.2.1 Java 62
2.2.2 Filosofía Orientada a Objetos 64
2.2.3 Oracle 66
2.3 Metodología de Desarrollo de Software. **66**
2.4 Marco Conceptual **67**
III MÉTODO **70**
3.1 Tipo de Investigación **70**
3.2 Población y Muestra **70**
3.3 Operacionalización de Variables **70**
3.3.1 Variables Independientes 70
3.3.2 Variables Dependientes 71
3.4 Instrumentos **71**
3.5 Procedimientos **71**

3.6 Análisis de Datos 72
3.7 Consideraciones Éticas 72
IV RESULTADOS 73
Propuesta de Sistema Informático de Programación de Ordenes 73
4.1 Enfoque 73
4.2 Comparación Planteamiento Clásico JobShop vs Propuesta 74
4.3 Resumen de la propuesta de Desarrollo 76
4.4 Modelamiento del Negocio actual 77
4.4.1 Identificación del Problema y Proceso involucrado 77
4.4.2 Diagramas DFD del Proceso Involucrado 82
4.4.3 Diagrama de Flujo del Proceso Involucrado 85
4.4.4 Descripción de Problemas hallados 86
4.4.5 Propuesta de Solución 87
4.4.6 Diagrama de Flujo Heurístico Propuesto 93
4.4.7 Objetivos del Nuevo Sistema 95
4.4.8 Requerimientos del Sistema de Información 95
4.5 Análisis del Sistema 96
4.5.1 Diagrama de Casos de Uso 96
4.5.2 Diagramas de Actividades 103
4.5.3 Diagrama de Clases 111
4.5.4 Diagrama de Estados 113
4.5.5 Diagrama de Secuencia. 114
4.6 Diseño del Sistema 116
4.6.1 Diagrama de Clases Físico. 116
4.6.2 Diagrama Entidad Relación 117
4.6.3 Patrón de Diseño 118
4.6.4 Diagrama de Clases Interface-Vistas 119
4.7 Implementación del Sistema. Base de Datos, Interface Grafica y Codificación. 120
4.7.1 Entornos de Desarrollo 120
4.7.2 Costo de la Implementación 121
4.7.3 Interfaces Graficas 123
4.7.4 Codificación 134
4.8 Resultados Descriptivos 139
4.9 Resultados Diferenciales 146
V DISCUSION DE RESULTADOS 154
VI CONCLUSIONES 156
VII RECOMENDACIONES 160
VIII REFERENCIAS 162
IX ANEXOS 165
9.1 ANEXO A Matriz de Consistencia 165
9.2 ANEXO B Codificación Clases 166
9.3 ANEXO C Codificación Conexión con la Base de datos: 177
9.4 ANEXO D Codificacion Interfaces y sus implementaciones 178
9.5 ANEXO E Codificación Recursivo 214
9.6 ANEXO F Codificación Vistas 217

ÍNDICE DE TABLAS

Tabla 1 Tabla de Tareas 15
Tabla 2 Tabla de Trabajos 16
Tabla 3 Secuencia de tareas 17
Tabla 4 Tabla de Tiempos por Maquina 17
Tabla 5 Ruta de Trabajos y Tiempos por Maquina 19
Tabla 6 Tiempo de Ciclo de Maquinas 20
Tabla 7 Simbologia de Diagramas de Flujo 33
Tabla 8 Tareas y sus Secuencias 48
Tabla 9 Máquina 1. Posibles secuencias de trabajo 49
Tabla 10 Fechas de Entrega por Orden 53
Tabla 11 Comparacion de Parametros Aleatorio vr Regla Despacho 61
Tabla 12 Variable Independiente 70
Tabla 13 Variable Dependiente 71
Tabla 14 Parametros JobShop: Planteamiento Clasico vr Planteamiento Investigacion 75
Tabla 15 Esquema General de la Investigacion 77
Tabla 16 Cuandro de Tiempos de los Procesos de Planeamiento 86
Tabla 17 Problemas Actuales y la Solucion Planteada 88
Tabla 18 Cuadro Entradas y Salidas del Sistema Propuesto 89
Tabla 19 Ejemplo de Disponibilidad de Maquinas 91
Tabla 20 Lista de Ordenes de Fabricacion 91
Tabla 21 Ejemplo de Gantt 91
Tabla 22 Lista de Actores del Sistema 96
Tabla 23 Caso de Uso Ingresar Producto 98
Tabla 24 Caso de Uso Ingresar Lista de Materiales 99
Tabla 25 Caso de Uso Ingresar Ruta del Producto 99
Tabla 26 Caso de Uso Ingresar Maquina 100
Tabla 27 Caso de Uso Ingresar Orden de Fabricacion 100
Tabla 28 Caso de Uso Consulta Gantt de Ordenes 101
Tabla 29 Caso de Uso Consulta Consumos 102
Tabla 30 Caso de Uso Consulta Ruta del Producto 103
Tabla 31 Estructura Lista de Materiales a Ingresar al Sistema 127
Tabla 32 Cuadro de Tomas de Tiempo Pre y Post Sistema de Programacion 139

ÍNDICE DE FIGURAS

Figura 1 Sistema de Produccion 26
Figura 2 Esquema JobShop 29
Figura 3 Esquema General JobShop-Ensamble 31
Figura 4 Ejemplo de secuencia de Produccion 33
Figura 5 Flujograma de Ejemplo 34
Figura 6 Estructura While 36
Figura 7 Estructura DoWhile 36
Figura 8 Estructura For 37
Figura 9 Secuencia de una Funcion 38
Figura 10 Secuencia de un Procedimiento 38
Figura 11 Recursividad 39
Figura 12 Maneras de Secuenciar por Maquina 49
Figura 13 Diagrama Gantt 51
Figura 14 Diagrama Gantt Software LEKIN 60
Figura 15 Ambiente Java 62
Figura 16 JobShop: Planteamiento Clasico vr Planteamiento Propuesta 74
Figura 17 Esquema Lista de Materiales 76
Figura 18 Orgranigrama de una Fabrica 79
Figura 19 Cadena de Valor de una Fabrica 80
Figura 20 Funciones del Area de Planeamiento y Control de la Produccion 80
Figura 21 Diagrama de Contexto Actual 82
Figura 22 Explosion Sistema Actual de Programacion de Ordenes 83
Figura 23 Proceso Elaborar Horario de Maquinas 83
Figura 24 Proceso Elaborar Lista de Materiales y Ruta del Producto 84
Figura 25 Proceso Elaborar Programa de Produccion 84
Figura 26 Flujograma Actual del Proceso de Planeamiento 85
Figura 27 Lista de Materiales 90
Figura 28 Ruta del Producto B 90
Figura 29 Diagrama de Flujo Heuristico Propuesto 94
Figura 30 Diagrama de Caso de Uso del Sistema 98
Figura 31 Diagrama de Actividad Ingresar Maquina 104
Figura 32 Diagrama de Actividad Ingresar Producto 105
Figura 33 Diagrama de Actividad Ingresar Lista de Materiales 106
Figura 34 Diagrama de Actividades Ruta del Producto 107
Figura 35 Diagrama de Actividades Ingresar Orden de Fabricacion 107
Figura 36 Diagrama de Actividades Consulta Consumo 108
Figura 37 Diagrama de Actividades Consulta Ruta Producto 109
Figura 38 Diagrama de Actividades Consulta Gantt Ordenes 110
Figura 39 Diagrama de Clases del Sistema Propuesto 112
Figura 40 Diagrama de Estados de la Clase Gantt 113
Figura 41 Diagrama de Secuencia de una instancia del Caso de Uso Consulta Gantt Ordenes. 115
Figura 42 Diagrama de Clases Fisico 116
Figura 43 Diagrama Entidad Relacion 117
Figura 44 Patron de Diseño 118
Figura 45 Diagrama de Clases con Interfaces y Clase Vista 119
Figura 46 Base de Datos del Sistema en Oracle Express 120
Figura 47 NetBeans 121

Figura 48 Ventana Inicial del Sistema 124
Figura 49 Ventana Ingreso de Maquinas 125
Figura 50 Ventana Ingreso del Producto 126
Figura 51 Ventana Ingreso de Orden de Produccion 126
Figura 52 Ventana Ingreso Lista de Materiales 127
Figura 53 Ventana Ingreso Ruta del Producto 128
Figura 54 Ventana Consulta Consumo de Materiales 129
Figura 55 Ventana Consulta Ruta del Producto 130
Figura 56 Ventana Inicial de Configuracion para el Diagrama Gantt 132
Figura 57 Ventana Resultado Final Diagrama Gantt del Sistema de Ordenes de Produccion por Maquina 133
Figura 58 Codificacion Clase Entidad Producto 134
Figura 59 Codificacion Clase Entidad Lista de Materiales 135
Figura 60 Codificacion Clase Recursivo 136
Figura 61 Codificacion Clase Interface 136
Figura 62 Codificacion Clase Implementacion Producto 137
Figura 63 Codificacion Clase Vista Producto 137
Figura 64 Codificacion Clase Conexion a Base de Datos 138
Figura 65 Estadistica Descriptiva Variable Gestion Industrial -Dimension: Tiempo Programa de Produccion 140
Figura 66 Grafica Comparacion de Medias Variable Gestion Industrial -Dimension: Tiempo Programa de Produccion Pre-Post 141
Figura 67 Estadistica Descriptiva Variable Gestion Industrial -Dimension: Tiempo Gestion Lista de Materiales 142
Figura 68 Grafica Comparacion de Medias Variable Gestion Industrial -Dimension: Gestion Lista de Materiales Pre-Post 143
Figura 69 Estadistica Descriptiva Variable Gestion Industrial -Dimension: Tiempo Ruta de Producto Pre-Post 144
Figura 70 Grafica Comparacion de Medias Variable Gestion Industrial -Dimension: Gestion Ruta del Producto Pre-Post 145
Figura 71 Prueba de Normalidad de los Datos 147
Figura 72 Prueba de Hipotesis Especifica 1 149
Figura 73 Prueba de Hipotesis Especifica 2 151
Figura 74 Prueba de Hipotesis Especifica 2 152

RESUMEN

El objetivo general fue desarrollar un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble que utiliza algoritmos heurísticos y recursivos para mejorar la Gestión Industrial. Se uso la siguiente metodología: el tipo de investigación es Aplicada, desde un alcance correlacional, con diseño Cuasi-Experimental debido a que la Muestra no es al azar, sino controlada y con enfoque cuantitativo. Se uso la técnica de la Observación y los instrumentos fueron la Computadora, libreta de apuntes y cronometro. La población fue todo el proceso de Gestionar la Empresa, y la Muestra es en el Área de Planeamiento. Esta fue no probabilística e intencional con la cantidad de 15 muestras de trabajo. La conclusión principal fue que se logró el objetivo de desarrollar un Sistema de Programación de Ordenes para empresas de fabricación Tipo Job Shop-Ensamble con algoritmos heurísticos y recursivos que mejoro la Gestión Industrial en el área de Planeamiento de una fábrica, mostrando disminuciones del 99.03%, 59.69% y 49.29% en el tiempo de procesamiento de programación de la producción, Gestión de Lista de Materiales y Gestión de Ruta de Productos.

Palabras Clave: programación de órdenes, job shop-ensamble, heurísticos, recursivo

ABSTRACT

The general objective was to develop an Order Scheduling System for Job Shop-Assembly type manufacturing companies that uses heuristic and recursive algorithms to improve Industrial Management. The following methodology was used: the type of research is Applied, from a correlational scope, with a Quasi-Experimental design because the Sample is not random, but controlled and with a quantitative approach. The observation technique was used and the instruments were the computer, notebook and stopwatch. The population was the entire process of Managing the Company, and the Sample is in the Planning Area. This was non-probabilistic and intentional with the number of 15 work samples. The main conclusion was that the objective of developing an Order Scheduling System for Job Shop-Assembly type manufacturing companies with heuristic and recursive algorithms that improved Industrial Management in the Planning area of a factory was achieved, showing decreases of 99.03%. , 59.69% and 49.29% in the processing time of production programming, Bill of Materials management and Product Route Management.

Keywords: order programming, job shop-assembly, heuristics, recursive.

I INTRODUCCIÓN

Hay varias investigaciones que abordan el tema de programación de órdenes desde diferentes enfoques. El tema de esta investigación abarca varios puntos a considerar: Programación de Ordenes, Ambiente Job Shop, Ensambles, Algoritmos Heurísticos y Recursivos, Ruta del Producto, Maquinas, Prioridad de Órdenes para la secuencia y Listas de Materiales. En la literatura se aborda el tema con distintos nombres como: Programación de Tareas, Task Scheduling, Calendarización de Trabajos, Programación Finita de la Producción, Gantt de Trabajos. Todas ellas aplicadas a ambientes industriales.

La programación de Ordenes es una actividad importante en el proceso de producción, forma parte del Sistema de Planeamiento y Control de la Producción de una empresa industrial. Toda empresa industrial compra materiales, las procesa y las convierte en productos terminados que van a los almacenes para despachar a los clientes. Cada producto final tiene una secuencia de producción y una lista de materiales que componen dicho producto. En la secuencia de producción, también llamada ruta de producción, se indica los procesos o máquinas por donde va a pasar la materia prima para ser procesada, los ensambles que tiene con otros subproductos, las denominaciones que obtiene en cada paso, los tiempos estimados que demora (lead times).

La decisión de cuales productos finales fabricar y en que cantidad se decide en una coordinación de diferentes áreas como: almacén de productos terminados, pedidos de clientes, gerencias de ventas, producción y logística. Cada producto destinado a fabricación se lanza a planta mediante una orden de producción, donde se indica el producto, la cantidad a producir y la fecha de emisión de la orden. Es aquí donde se obtiene el escenario de tener muchas órdenes de producción en planta, con diferentes rutas de producción en diferentes máquinas y

ensambles, y con la urgencia de distribuir esas órdenes con una secuencia de prioridad, es decir, indicar las órdenes a empezar en cada máquina y coordinar los futuros ensambles que se van a dar. En otras palabras, establecer un Gantt por máquina, indicando las materias primas, subproductos, ensamble y productos terminados de cada orden a procesar, calendarizados por un tiempo inicio y final aproximado, todo esto configurado con las prioridades que se establece para cada orden.

Este proceso es dinámico, ya que se considera que las órdenes entran de manera continua, además de hacer simulaciones con diferentes criterios de prioridad para las órdenes, máquinas ocupadas y cambios en la lista de materiales y rutas de producción. Todo esto debe ser resuelto de manera rápida y efectiva, centralizada e integral, por lo que una solución de Sistema informático que considere todas estas variables es necesaria para resolverlo. En el modelado de la Lista de Materiales y Rutas de Producción, se vislumbra el tema de Padre-Hijo y estructura de árboles, por lo que un algoritmo recursivo es el mas adecuado, ya que así solo se requeriría una sola tabla para representar la estructura del producto, considerando los ensambles.

En el caso del modelado del Gantt, la cantidad de variables y opciones de prioridad impiden una solución matemática exacta óptima para establecer la secuencia final Gantt en las máquinas, por lo que un algoritmo heurístico es una aplicación adecuada para este caso. Este problema ha sido tratado en varios trabajos que se van a explicar, con soluciones de Software, pero que no consideran todos los aspectos que se mencionó anteriormente.

1.1 Planteamiento del Problema

El sector manufacturero del Perú representa el 13% del PBI global, y depende del crecimiento de ese valor para lograr el salto al desarrollo que aspiramos como país. Un indicativo de un país industrializado y plenamente desarrollado es el porcentaje que el sector manufactura aporta al PBI, ya que genera pleno empleo, desarrolla la tecnología y eleva el crecimiento. Uno de los factores para poder elevar ese 13% es la productividad, es decir hacer mas con menos o iguales recursos. Entiéndase por recursos, todo lo que se consume como materiales, tiempo, maquinas, horas hombre y gestión. Y en la productividad en manufactura es donde el Perú tiene uno de los valores mas bajos en la región. Una de las causas, entre otras muchas, es el excesivo tiempo que se consume en la gestión de las órdenes de los clientes y/o almacén para ser trabajadas en las máquinas y así producir los skus de una planta de producción. Una fábrica trabaja en base a órdenes de producción, ya sean dictadas por un cliente o para abastecer el inventario.

En el caso particular del Perú, tenemos que el 96% de las empresas de manufactura es PYME, este indicador nos dice que el volumen de producción de la mayoría de las empresas peruanas es pequeño, pero con gran variedad de productos (SKUs). Esto genera un volumen importante de Ordenes de Producción para gestionar en Planta. Y estas necesitan ser generadas, asignadas y distribuidas, tomando en cuenta las máquinas disponibles, los turnos de producción, la estructura, ruta y ensamble de cada Skus, la disponibilidad de material y las prioridades de las órdenes. El objetivo siempre es tratar de asignar a cada máquina de la empresa, los trabajos (ordenes) que tiene que hacer, en productos intermedios o terminados, durante 1 semana de producción como mínimo, para así permitir un adecuado abastecimiento de materiales. Esto permite ahorrar tiempo en gestión de planta, evitar cambios inesperados en máquina por cambio de programación, previsión de materiales, aumento de la productividad en planta gracias a tener la lista de fabricación semanal de antemano y generación de fechas de

entrega fiables al cliente. Esta necesidad es evidente en la realidad de las empresas peruanas. El objetivo descrito es difícil de lograr con las herramientas tecnológicas actuales que poseen las PYMEs, como el EXCEL o algún Sistema Interno.

En el caso del Excel, si bien es una herramienta poderosa de alta personalización, no permite la integridad sencilla de los datos, genera mucha redundancia, información desordenada en varias áreas y lentitud al procesar miles de registros. En el caso de los Sistemas Internos, estos no contemplan la gestión de Planta, limitándose solamente a control de inventarios, facturación, ventas y contabilidad. Si bien existen ERPs que ofrecen gestionar todas estas variables, la inversión necesaria esta fuera del presupuesto de la mayoría de las empresas peruanas, al margen de la conocida dificultad en la implementación del ERP en aquellas empresas que si tienen el presupuesto. Por lo que hace falta una solución practica que centralice la información de planta, la procese y genere el programa de trabajo por máquina diario o semanal de manera rápida y dinámica. Considerando que las órdenes de los clientes o de almacén entran diariamente a la fábrica, trabajándose por día muchas órdenes, de productos diferentes y cada uno de estos con rutas y prioridades diversas; un sistema de información que procese toda esa información se hace necesario.

En la presente investigación se propone un Sistema de Programación de Ordenes en ambiente Job Shop-Ensamble (esto es así porque la realidad de las empresas peruanas es la variedad de productos que se fabrica y que tienen rutas muy diversas con ensambles incluidos a lo que se denomina Job Shop-Ensamble) basado en Java y Oracle 11g Express Edition como manejador de Base de Datos. Además, la complejidad de las rutas y ensamble de los productos, así como el número alto de combinaciones posibles en la prioridad y secuencia de trabajos hace necesario el uso de algoritmos recursivos y heurísticos para hallar soluciones de programas de trabajo, que sean satisfactorios para el objetivo de la presente investigación. En resumen, dados los siguientes datos: Listado de Ordenes, Prioridad de las Ordenes (varios criterios), Estructuras

de los Productos (Lista de Materiales con Ensamble), Ruta de Producción (Maquinas y Procesos por donde van a pasar los productos), Maquinas disponibles, Turnos y Horarios de Producción, se propone generar un Listado de Trabajos (Productos Terminados, en proceso y Materia Prima) por Maquina calendarizados con una fecha/hora inicio y fin en un periodo determinado con resúmenes para la toma de decisiones. Cabe resaltar que no estamos buscando la secuencia mas optima, la secuencia ya está definida como un parámetro de entrada a elección del gestor.

1.2 Descripción del Problema

El trabajo de investigación esta centrada en la realidad de las fábricas PYMES en el Perú, donde el proceso de fabricación Job Shop-Ensamble es la norma

Se refiere a las fábricas en el contexto actual, que es donde se presenta la variedad de productos y cantidad de PYMES.

La investigación se centra en desarrollar un Sistema de programación de órdenes para lograr rapidez en la gestión de una fábrica genérica tipo Job Shop-Ensamble, donde existen procesos intermitentes y de ensamble. Para ello se aplicará algoritmos heurísticos y recursivos en un contexto de Software.

La unidad de Análisis es la utilidad de los algoritmos heurísticos y recursivos aplicados en un Software para obtener una programación de órdenes efectiva en los ambientes industriales PYMEs de Job Shop-Ensamble y de esa manera mejorar la Gestión Industrial.

1.3 Formulación del Problema

1.3.1 Problema General.

¿Cómo el Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble utilizando algoritmos heurísticos y recursivos permitirá mejorar la Gestión Industrial?

1.3.2 Problemas Específicos.

- ¿Cómo el Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble utilizando algoritmos heurísticos y recursivos permitirá obtener rapidez en la Programación de la Producción?
- ¿Cómo el Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble utilizando algoritmos heurísticos y recursivos permitirá obtener eficiencia en la Gestión de Lista de Materiales?
- ¿Cómo el Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble utilizando algoritmos heurísticos y recursivos permitirá obtener eficiencia en la Gestión de la Ruta del Producto?

1.4 Antecedentes

1.4.1 Antecedentes Nacionales

Ramirez (2006) en su tesis sobre un algoritmo Grasp con doble Relajación para resolver el problema de Flow Shop Scheduling propone una solución para un problema de secuenciación de tipo Flow Shop, con el objetivo de reducir el tiempo de procesamiento total de todos los trabajos. El tipo Flow Shop es un sistema de producción donde la secuencia de máquinas es la misma para todos los productos. SI bien este caso es una particularidad del Sistema Job Shop, es una muestra de la aplicación de Algoritmos Heurísticos para resolver problemas de Scheduling. El Algoritmo Grasp se basa en construir soluciones en base a varias

alternativas escogidas al azar, pero antes estas alternativas han sido evaluadas como las mejores dentro de la vecindad de la solución inicial. Además, el autor implementó un servidor Web basado en PHP, Linux y Apache para la aplicación práctica del Algoritmo.

El autor usa la siguiente tabla como una de las variables de entrada para el Sistema.

Tabla 1

Tabla de Tareas

Tareas	F1	F2	F3	…	… Fk
J1	Xx	xx	xx	xx	Xx
J2	Xx	Xx	Xx	Xx	Xx
..	Xx	xx	Xx	Xx	xx
Jn	Xx	xx	xx	xx	Xx

Fuente: (Ramirez, 2006)

Donde:

F1 es la familia de máquinas 1

J1 es la Tarea

Xx es el tiempo de cada tarea en una máquina de la familia F1.

Aquí el autor, como habíamos indicado, considera un sistema Flow Shop, es decir todos los trabajos tienen la misma secuencia. La novedad es el concepto de Familia de Productos en vez de máquinas, el cual es un buen aporte a la investigación. En este trabajo tampoco se considera el ensamble de productos.

El autor prueba exitosamente el sistema creado en PhP para un escenario de 200 tareas y 40 máquinas distribuidas en 3 familias, obteniendo buenos resultados. Asimismo, considera que el Sistema es perfectamente aplicable a las industrias peruanas por su bajo costo de implementación.

Tello (2014) en su tesis sobre Algoritmos Evolutivos propone usar algoritmos Metaheurístico Evolutivo para resolver el problema de secuenciación en ambiente Job Shop con el objetivo de reducir el MaskesPan (Tiempo de terminación de todas las tareas). El metaheurístico Evolutivo simula el proceso de evolución, donde el método encuentra una solución completa inicial factible y esta solución inicial la va mejorando en cada paso, mediante herencia de padres a hijos, a semejanza del proceso evolutivo donde la descendencia puede heredar lo mejor de los padres. La autora elabora un Software para probar el Método, elaborado en Visual Basic.

Usa la siguiente tabla teórica como ingreso al Sistema.

Tabla 2

Tabla de Trabajos

	TRABAJOS				
Maquinas	T0	T1	T2	T3	T4
M1	X	X	X	X	X
M2	X	X	X	X	X
M3	X	X	X	X	X

Fuente: (Tello, 2014)

El formato de entrada tiene la misma distribución habitual para los problemas del Sistema JobShop. La autora desarrolla su propuesta de Método para el ejemplo de la tabla: 3 máquinas y 4 trabajos, logrando desarrollar una solución en formato Gantt.

En este trabajo de investigación no se considera el caso de Ensambles de Productos. La autora muestra resultados que son satisfactorios para un ambiente teórico de 15 trabajos y 15 máquinas. La autora menciona que el algoritmo puede resolver casos cuando los trabajos no pasan por todas las máquinas, parámetro que se muestra a menudo en la realidad.

Además, la autora ejecuta la aplicación en un contexto real de fábrica, específicamente para una realidad de 14 máquinas y 8 trabajos por realizar. Para este caso usa el siguiente formato de entrada para la secuencia:

Tabla 3

Secuencia de tareas

Trabajos	Secuencia
1	M1-M3
2	M7-M13-M5-M1-M3
3	M7-M14-M5-M9
4	M6
5	M10
6	M1-M3
7	M6-M12-M11
8	M11

Fuente: (Tello, 2014)

Para la tabla de tiempos usa el siguiente formato:

Tabla 4

Tabla de Tiempos por Maquina

	TRABAJOS							
MAQUINAS	1	2	3	4	5	6	7	8
M1	30	120	0	0	0	120	0	0
M2	0	0	0	0	0	0	0	0
M3	10	30	0	0	0	30	0	0
M4	0	0	0	0	0	0	0	0
M5	0	20	20	0	0	0	0	0
M6	0	0	0	10	0	0	20	0
M7	0	30	30	0	0	0	0	0
M8	0	0	0	0	0	0	0	0
M9	0	0	90	0	0	0	0	0
M10	0	0	0	0	20	0	0	0
M11	0	0	0	0	0	0	20	20
M12	0	0	0	0	0	0	20	0
M13	0	30	0	0	0	0	0	0
M14	0	0	30	0	0	0	0	0

Fuente: (Tello, 2014)

Como se muestra, el formato de entrada para la secuencia y los tiempos tiene un estilo tabular, adecuado para mostrar claramente la distribución de los datos. Luego la autora ejecuta el método y muestra un Gantt con los trabajos procesados en cada máquina a una escala de tiempo, este resultado los compara con el Software WinQSB arrojando resultados similares.

Finalmente, la autora recomienda mejorar el algoritmo para la aplicación en máquinas paralelas para si aumentar el campo de acción del Método creado.

1.4.2 Antecedentes Internacionales

Sippper y Bulfin (1998) en su libro sobre Planeación y Control de la Producción, en el cap. 8.6.2 Programación de Operaciones/Producción Intermitente/Despacho pp.458-461, los autores muestran un heurístico para programar las tareas en las máquinas en un sistema de Producción Intermitente (Job Shop). La propuesta de los autores considera las siguientes suposiciones iniciales:

1. Todos los trabajos tienen rutas diferentes (Job Shop o Intermitente).
2. El tiempo 0, todas las máquinas están disponibles.
3. Cada máquina solo puede procesar un trabajo a la vez.
4. Cuando un trabajo empieza en una máquina, se termina completamente.
5. Cualquier número de máquinas, y cualquier número de trabajos.
6. Se programa un trabajo en una máquina tan pronto como se pueda.
7. Si hay varios trabajos esperando una máquina, se programa el trabajo con la mejor prioridad.
8. Las prioridades pueden ser diversas:
    - TPC: Tiempo de procesado mas corto.
    - PEPS: Primera Orden en llegar.
    - FEC: Fecha de Entrega mas cercana.
    - Etc.
9. Para la ruta de producción usan esta tabla como ejemplo:

Tabla 5

Ruta de Trabajos y Tiempos por Maquina

	Operación			
Trabajo	1	2	3	4
1	6/1	8/2	13/3	5/4
2	4/1	1/2	4/3	3/4
3	3/4	8/2	6/1	4/3
4	5/2	10/1	15/3	4/4
5	3/1	4/2	6/4	4/3
6	4/3	2/1	4/2	5/4

Fuente: (Sippper y Bulfin, 1998)

Aquí 6/1 indica que 6 es el tiempo de procesamiento y 1 es la máquina de la primera operación del trabajo 1. Como se aprecia, esta tabla es un ejemplo del trabajo intermitente, ya que el trabajo 1 tiene la secuencia de máquinas 1-2-3-4, pero el trabajo 3 tiene la secuencia de máquinas 4-2-1-3.

De las suposiciones consideradas en el libro, el modelo propuesto en la investigación considera los puntos 1,3,4,5,6 y 7, pero el libro no considera el ensamble de productos, y además considera que entre los trabajos no hay subproductos que coinciden (subproductos comunes), aspectos que si considera el modelo propuesto. Estas dos consideraciones adicionales (ensambles y subproductos comunes, que a la vez son muy recurrentes en una fábrica real) aumentan la dificultad del modelo propuesto. El libro no presenta alguna solución de Software aplicado.

Además, el algoritmo del libro considera que la tabla del punto 8 ya esta dada de antemano y no muestra como se obtiene la secuencia de máquinas en formato tabular a partir de una lista de materiales, aspecto que si considera el modelo propuesto, a partir de algoritmos recursivos aplicados a la tabla de materiales.

En el punto 2, mientras el libro considera que al tiempo 0, todas las máquinas están disponibles, la propuesta considera que la programación es dinámica, y que al tiempo de corrida del programa, puede haber máquinas ocupadas o no disponibles en ese momento.

Al final el libro presenta un diagrama Gantt presentando la secuencia en cada máquina de los trabajos programados.

Valle (2013) en su tesis sobre Programación de Rutas de Materiales propone elaborar una secuencia de programación de trabajos en un ambiente Job-Shop de una empresa real usando un Metaheurístico Genético, esto es, simulando el proceso natural de la evolución genética. La investigación presenta un Software elaborado para tal fin, llamado BAPS. Al igual que en la investigación anterior, el autor omite en sus considerandos el Ensamble de Productos, se dirige exclusivamente al área de maquinado, donde las secuencias son lineales. Pero establece un Objetivo de investigación, que es minimizar el tiempo de terminación de todos los trabajos, sin considerar la prioridad diferente de cada trabajo. Esto es muy diferente al objetivo propuesto por la investigación propia, que es programar los trabajos en base a prioridades. A continuación, se muestra la tabla que el autor considera como entrada al Sistema BAPS:

Tabla 6

Tiempo de Ciclo de Maquinas

				TIEMPO DE CICLO DE MAQUINAS			
Componente	Nombre	Precedencia de Procesos	LOTE	TO	CN	CM	BA
COMP1	COMP1	TO-CN-BA	12	XX	XX	XX	XX
COMP2	COMP2	CN	24	XX	XX	XX	XX
COMP3	COMP3	CN-BA	12	XX	XX	XX	XX
COMP4	COMP4	BA-TO	24	XX	XX	XX	XX

Fuente: (Valle, 2013)

Como se aprecia en la Columna Precedencia de Procesos, la tabla muestra correctamente un ambiente Job Shop, pero no muestra algún tipo de ensamble, ya que la

precedencia es lineal (TO-CN-BA), esto es porque esta orientado solamente al área de maquinado y no de ensamble.

Al final el autor presenta el diagrama Gantt con la secuencia de trabajos por máquina, al cual le llama Plan de Secuenciación de Trabajos.

El autor concluye que la implantación del Software BAPS con base en algoritmos genéticos en la empresa donde trabaja es un éxito, que permite programar rápidamente el área de maquinado. También muestra los aportes a la investigación sobre el tema de optimización combinatoria, mostrando que su trabajo puede trabajar exitosamente con problemas de 29 máquinas y 295 órdenes, aspectos muy reales en el ambiente industrial. Además, indica que puede ampliarse la investigación para ambientes JIT.

Todo esto no hace mas que confirmar la importancia de la investigación propuesta en el presente trabajo, ya que el uso de un Software para programar trabajos en Ambiente Job Shop es de utilidad práctica.

1.5 Justificación de la Investigación.

El presente trabajo permite aportar en la investigación de la programación de órdenes en ambiente Job Shop-Ensamble. La demora en la generación de un calendario de trabajo por órdenes para las máquinas dentro de la planta industrial y tener repartida la información de producción en diferentes repositorios de datos, no permite una eficiente gestión en planta. El presente trabajo propone una solución acorde con la necesidad de las empresas peruanas. La industria puede beneficiarse de esta investigación, ya que presenta una solución practica a los problemas de gestión de órdenes. La investigación aborda un tema recurrente en las industrias peruanas, donde el ambiente Job Shop-Ensamble es el proceso mas común. La dificultad radica en la consideración de procesos intermitentes (los productos pasan por diversas máquinas en secuencias diferentes) y de ensamble (unión de semiproductos elaborados previamente). Ello

implica diseñar una base de datos eficiente para modelar la lista de materiales y el algoritmo adecuado para elaborar el calendario (programación) de órdenes.

La importancia de la investigación se puede resumir en los siguientes puntos:

- Aporta a la investigación de Programación de Ordenes en ambiente Job-Shop Ensamble, modo de producción recurrente en las PYMES en el Perú (96% de las empresas).
- Centralización de la información en una sola Base de Datos, logrando integridad de datos (Órdenes de Venta, Lista de Materiales, Ruta del Producto, Maquinas, Horarios, Prioridad).
- Permitirá agilizar el proceso de programación de órdenes ganando productividad en la gestión de planta en concordancia con el dinámico trabajo en las PYMES con muchas ordenes de lotes pequeños, diversos productos, diferentes rutas y listas de materiales.
- Sencillez en el ingreso de Datos de Lista de Materiales, Ruta del Producto y Ordenes de Producción.
- Costo del Sistema S/. 15,000 (como se indica posteriormente en Costo de Implementación) accesible para unas ventas PYME entre 150 UIT-2300 UIT anuales (S/. 742,500- S/. 11’385,000). Comparado al Costo de instalación de un SAP.
- Comunicación de las Fechas de Entrega de las órdenes, obteniendo satisfacción del Cliente.

1.6 Limitaciones de la Investigación.

- Ingreso lento de datos reales a la Lista de materiales cuando la empresa produce millones de artículos. Si bien el sistema puede soportar el procesamiento de la información, haría falta cambiar el manejador de base de datos para hacer viable el aplicativo y contratación de personal para el llenado de las rutas, productos y ensambles a la base de datos.
- Entorno grafico básico de los resultados de la investigación en el sistema, ya que el presente trabajo se ha concentrado en la aplicación de los algoritmos heurísticos y recursivos en la programación de órdenes. Pero cumple con el objetivo de informar de manera detallada para la toma de decisiones.

1.7 Objetivos.

1.7.1 Objetivo General

Desarrollar un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble que utiliza algoritmos heurísticos y recursivos para mejorar la Gestión Industrial.

1.7.2 Objetivos Específicos.

- Desarrollar un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble que utiliza algoritmos heurísticos y recursivos para obtener rapidez en la Programación de la Producción.
- Desarrollar un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble que utiliza algoritmos heurísticos y recursivos para mejorar la eficiencia en la Gestión de la Lista de Materiales.
- Desarrollar un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble que utiliza algoritmos heurísticos y recursivos para mejorar la eficiencia en la Gestión de la Ruta del Producto.

1.8 Hipótesis

1.8.1 Hipótesis General

Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces se mejora la Gestión Industrial.

1.8.2 Hipótesis Especifica

1. Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces se obtiene rapidez en la Programación de la Producción.

2. Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces se mejora la eficiencia en la Gestión de la Lista de Materiales.

3. Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces se mejora la eficiencia en la Gestión de la Ruta del Producto.

II MARCO TEÓRICO

2.1 Job Shop-Ensamble y Heurística.

La propuesta de la presente investigación: Sistema de Programación de Ordenes en ambiente industrial Job shop-Ensamble utilizando Algoritmos Heurísticos y Recursivos, abarca varios temas, pero todos interrelacionados. A manera de resumen se puede adelantar que en el tipo de ambiente industrial JobShop se trabaja mediante órdenes de producción (necesarias para organizar la producción), y estas necesitan ser programadas en las máquinas JobShop en una secuenciación dictada por algún método. Estos métodos pueden ser intuitivos o seguir alguna regla formal.

Dentro de los varios métodos formales, podemos usar los llamados Algoritmos Heurísticos y Recursivos que tienen la ventaja, sobre los métodos intuitivos, de la rapidez y de encontrar una secuencia que minimice alguna variable como, por ejemplo, la fecha final de todas las tareas. De esta manera se grafica someramente la relación entre los conceptos de la propuesta a manera de introducción. Pero a continuación detallaremos cada uno de los conceptos, así como sus relaciones.

2.1.1 Job Shop-Ensamble.

Comenzaremos con el tipo de ambiente industrial JobShop-Ensamble, para ello explicaremos algunos conceptos necesarios para entender la procedencia de la definición.

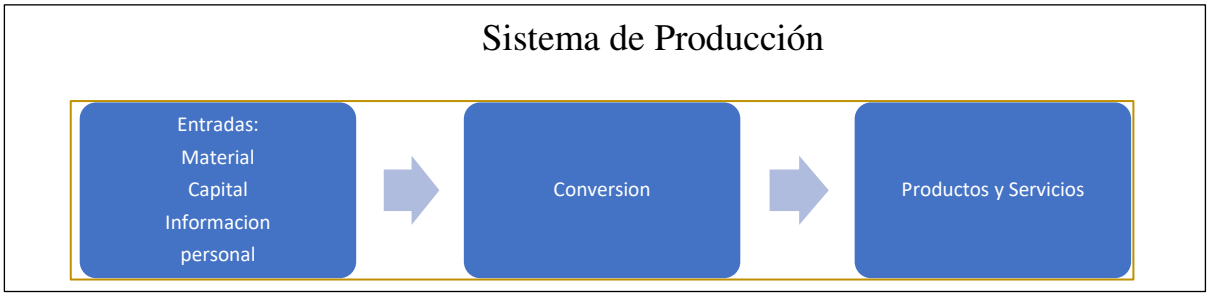
2.1.1.1 Sistema De Producción

El mundo tal como lo conocemos se mueve mediante el consumo de algún tipo de producto o servicio, y estos son determinados por las diferentes necesidades del ser humano. Siempre alguien necesita algún producto o servicio y existen personas que ofrecen tales productos. Pero un producto necesita elaborarse, fabricarse y para ello es que se creó el concepto de Producción. Gaither y Frazier (2000) lo define así: “Un

Sistema de Producción recibe insumos en forma de materiales, personal, capital, servicios e información. Estos insumos son transformados en un subsistema de conversión en los productos y servicios deseados, que se conocen como productos." (p. 15). En otras palabras, hay una entrada de entidades y un procesamiento de estas para convertirlas en salidas.

Figura 1

Sistema de Producción



Este procesamiento es el llamado subsistema de conversión, que mas adelante los autores Gaither y Frazier (2000) explican:

> El núcleo central de un sistema de producción es su subsistema de conversión, mediante el cual los trabajadores, materiales y máquinas se utilizan para convertir los insumos en productos y servicios. El proceso de conversión esta en el centro de la administración de la producción y de las operaciones y de alguna manera está presente en toda la organización. (p. 16)

Esto significa que la conversión de los insumos es vital para cualquier organización.

2.1.1.2 Estrategia De Operaciones.

Ahora bien, para poner en marcha el Sistema de Producción indicado en la figura, necesitamos de una estrategia de Operaciones, esto es, Gaither y Frazier (2000) "un plan de acción a largo plazo para la elaboración de productos/servicios de una

empresa y nos aporta un mapa de lo que debe hacer la función de producción si se han de lograr las estrategias empresariales" (p. 42). Como bien indican los autores, la estrategia de Operaciones nos indica la serie de acciones que debemos seguir para poner en actividad la función(sistema) de producción y poder así cumplir la estrategia empresarial. Las acciones básicas necesarias para poner en curso una estrategia básica de operaciones son las siguientes:

Diseño del Producto.

Indica cuales con las características del producto que se desea fabricar (diseño del producto final y materiales). Hay 2 tipos de diseño del Producto:

Sobre Pedido.

Es cuando el producto es altamente personalizado, gran variedad y en pequeñas cantidades.

Estándar.

Pocos productos y en grandes cantidades.

Tipo de Procesamiento de la Producción.

Determina como se va a efectuar el proceso de conversión (parte del Sistema de Producción), pueden ser:

Enfocado al Producto.

Gaither y Frazier (2000) "describe una forma de organización de procesamiento de la producción en la cual los departamentos de producción están organizados de acuerdo con el tipo de producto/servicio que se está elaborando" (p. 120). Esto es, los procesos

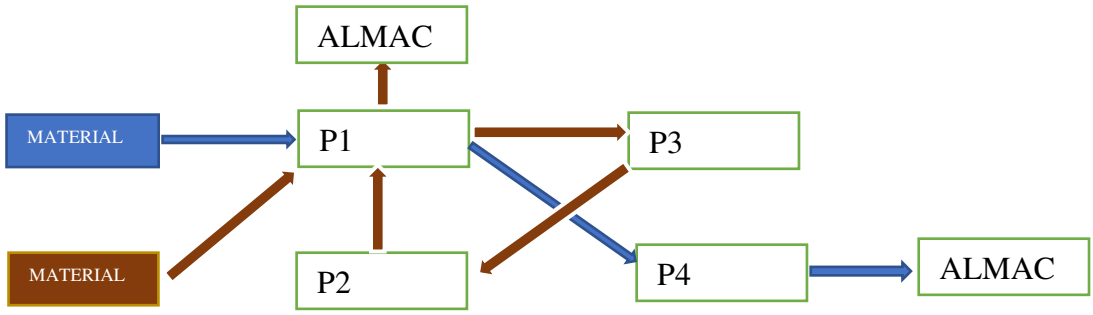
de producción están organizados de tal manera que el producto final siga una línea continua de producción. Por ejemplo, la fabricación de automóviles, donde el automóvil empieza como una carcasa y esta pasa por una línea, donde a cada cierta distancia se van ensamblando mas y mas componentes hasta completar el acabado del automóvil. Es llamada también producción Continua o Ensamble. Como vemos en el tema de estudio JobShop-Ensamble, acabamos de definir el termino ensamble.

Enfocado al Proceso.

Gaither y Frazier (2000) ”describe una forma de producción en la cual las operaciones se agrupan según los tipos de proceso” (p. 122). Esto es, los procesos de producción están organizados por tipos de operación. Por ejemplo, el área de corte, donde están todas las máquinas cortadoras, o el área de pintura, donde se pintan todos los productos de la fábrica. En este tipo de Procesamiento, un producto puede pasar por diferentes áreas y en diferente orden. Es decir, la secuencia puede ser primero se pinta y luego se corta, pero también puede cortarse y luego pintarse. También se le llama tipo de proceso Intermitente, JobShop o Talleres de producción. A continuación, un gráfico explicando este tipo de procesamiento.

Figura 2

Esquema JobShop



En el grafico se muestra las siguientes rutas:

- PRODUCTO 1: P1-P4-ALMACEN.
- PRODUCTO 2: P1-P3-P2-P1-ALMACEN.

Donde se nota las diferentes rutas para 2 productos diferentes. Con lo cual acabamos de explicar el Primer Término del tema de estudio: Job Shop.

2.1.1.3 Concepto Job shop Ensamble

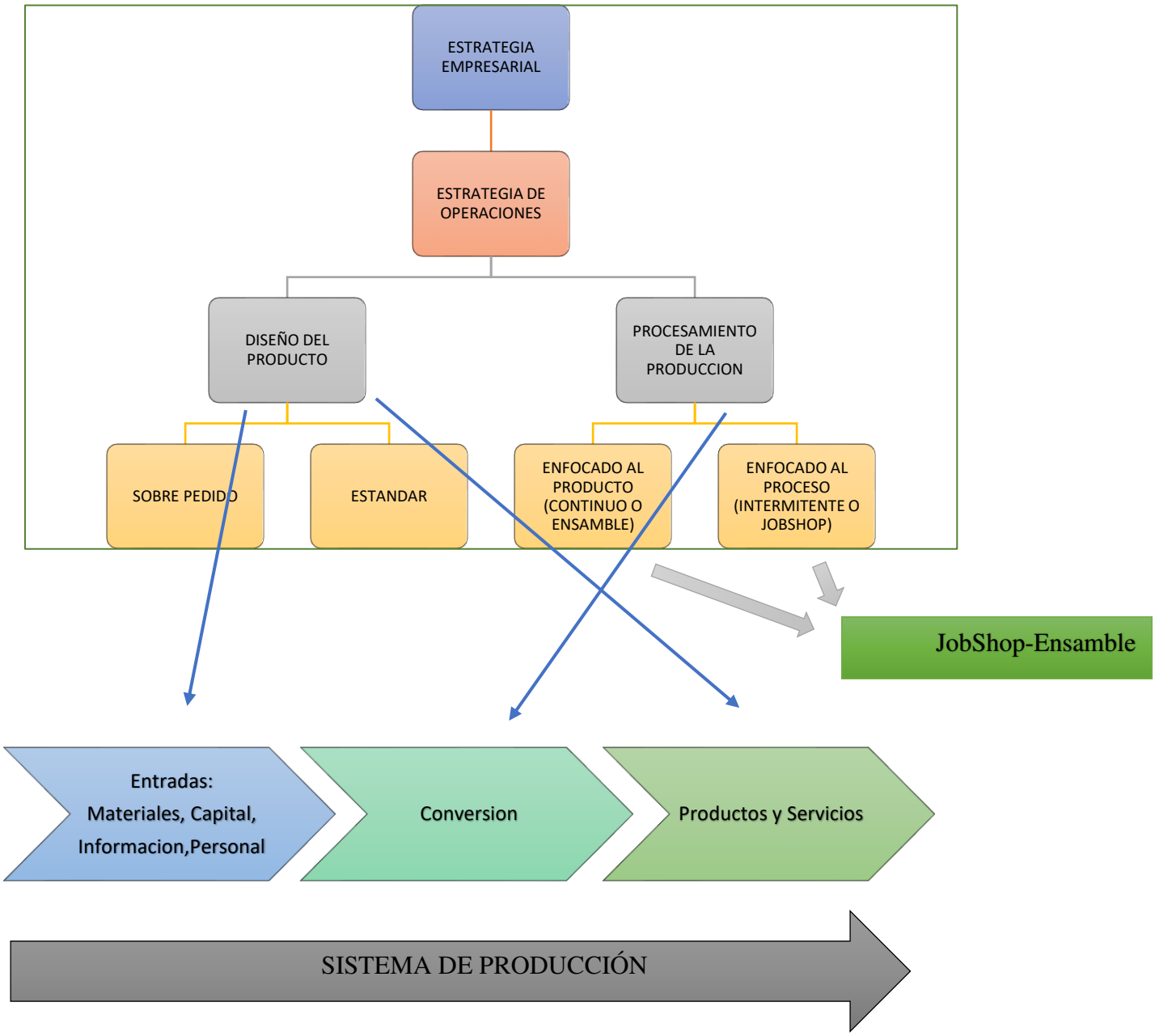
Diseño del Producto y Tipo de Procesamiento son los 2 elementos suficientes para poner en marcha el Sistema de Producción dentro de una Estrategia Básica de Operaciones para poder cumplir la Estrategia Empresarial. Ahora, dentro del Tipo de Procesamiento, se ha definido 2 tipos de procesos: enfocado al producto y enfocado al proceso, pero una empresa puede escoger un tipo u otro, pero también una mezcla de ambos, que es lo que sucede en la realidad de las empresas en el mundo, incluyendo la peruana. Es decir, se puede implementar procesos intermitentes (Job shop) y también de ensamble, a esa unión es la que llamamos Job Shop-Ensamble. Ejemplo: Para producir Filtros de Carros, primero tenemos que fabricar los componentes en un proceso Intermitente (procesos de Cortado, Prensado, Embutido) en la cual fabricamos todos

los elementos de un filtro, listos para ser ensamblados en una línea de ensamble, donde se coloca primero la carcasa del filtro y luego se van adicionado en una línea todos los elementos ya fabricados. Este tipo mixto de proceso de producción es la que gran parte de las empresas peruanas poseen en sus fábricas, por lo que el Sistema Propuesto tiene una plena aplicación en la realidad industrial.

Como explicábamos al principio, este sistema de producción JobShop Ensamble trabaja con órdenes que necesitan ser programadas según algún método o algoritmo. Mas adelante se detallará el concepto de programación de órdenes y como abordarlo para ser resuelto mediante un algoritmo. A manera de graficar se muestra el siguiente cuadro donde se muestra la interrelación de todo lo explicado.

Figura 3

Esquema General JobShop-Ensamble



2.1.2 Introducción a los Algoritmos.

Para empezar, definiremos el termino Algoritmo y luego a explicar cada una de sus clasificaciones.

2.1.2.1 Algoritmos

Joyanes (2003) indica:

> La palabra Algoritmo se deriva de la traducción al latín de la palabra Alkhowarizmi, nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX. Un algoritmo es un método para resolver un problema mediante una seria de pasos precisos, definidos y finitos. (p. 40)

Como indica el autor, los algoritmos fueron ideados para resolver problemas y encontrar una solución. Dado un problema se hace un análisis de este para encontrar el algoritmo adecuado. Una vez encontrado el algoritmo (cálculos en un papel o mentalmente), este debe ser representado de alguna manera para que el algoritmo sea definido formalmente y así pueda ser posteriormente traducido a algún programa específico como Visual Basic, Java o Python. Esta representación puede ser de 2 formas: Diagramas de Flujo y Pseudocodigo. Para ellos vamos a proponer un problema simple, analizarlo y proponer al algoritmo adecuado, para luego representarlo mediante las 2 formas:

Problema: Calcular el tiempo total de fabricación de 2 piezas, si las 2 piezas pasan por 3 procesos. Primero pasa por el proceso de cortado (tiempo=10 min x pieza), luego proceso de soldado (tiempo=15 min x pieza) y luego proceso de pintado (tiempo=20 min x pieza).

Analizando el problema, descubrimos que solo tenemos que multiplicar los tiempos por pieza de cada proceso y multiplicarlos por el número de piezas (2) y luego sumar los tiempos hallados.

Figura 4

Ejemplo de secuencia de Producción

Tiempo cortado = 2 piezas x 10 min / pieza=20 min

Tiempo soldado = 2 piezas x 15 min / pieza=30 min

Tiempo pintado = 2 piezas x 20 min / pieza=40 min

Tiempo total = 20+30+40=90 min.

Ahora vamos a representar el algoritmo hallado.

- Diagramas de Flujo. Joyanes (2003) "Es una representación gráfica de un Algoritmo" (p. 43).

Para representar un Algoritmo por un Diagrama de Flujo se usan los siguientes símbolos:

Tabla 7

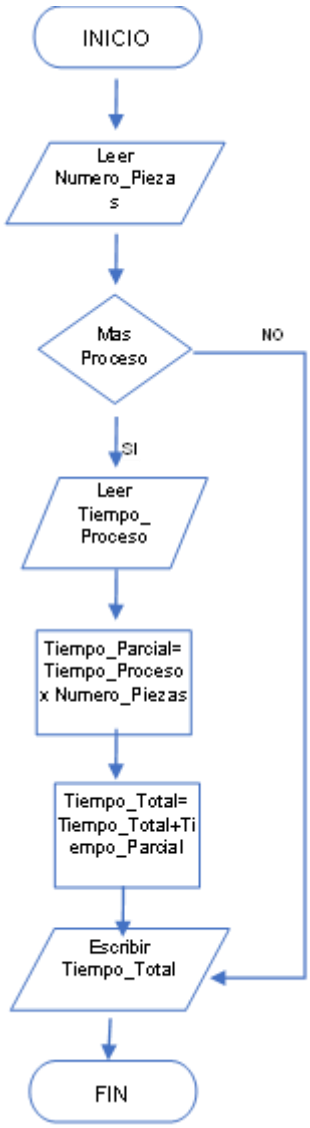
Simbología de Diagramas de Flujo

Entrada o Salida de Datos	
Decisión	
Proceso	
Terminal	

Para el problema:

Figura 5

Flujograma de Ejemplo



- Pseudocodigo. Joyanes (2003) “Es una herramienta de programación en la que las instrucciones se escriben en palabras similares al inglés o español, que facilitan tanto la escritura como la lectura de programas” (p. 43). Las palabras más comunes que se usan

para escribir un pseudocodigo son start, read, write, end, if, then, else, while, repeat y until. Por ejemplo, para el problema propuesto el Pseudocodigo seria:

```
start
        read Numero_piezas
        read Procesos
        while hay mas procesos do
                read tiempo_proceso
                tiempo_parcial=tiempo_proceso x Numero_piezas
                tiempo_total = tiempo_total + tiempo_parcial
        end while
write tiempo_total
end
```

En el Pseudocodigo mostrado se ha graficado un Algoritmo Repetitivo o Interactivo (while) a manera de introducción al siguiente tema.

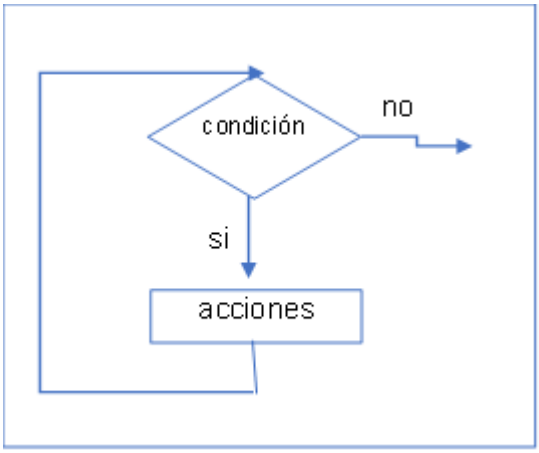
2.1.2.2 Algoritmos Repetitivos

Joyanes (2003) “Son aquellos Algoritmos qué repiten una o varias acciones un número determinado de veces. A la estructura que se repite se llama Bucle y se denomina interacción al hecho de repetir la ejecución varias veces” (p. 158). Este tipo de algoritmo es de amplio uso en los programas informáticos y vamos a listar someramente los diversos tipos:

- While.

Figura 6

Estructura While

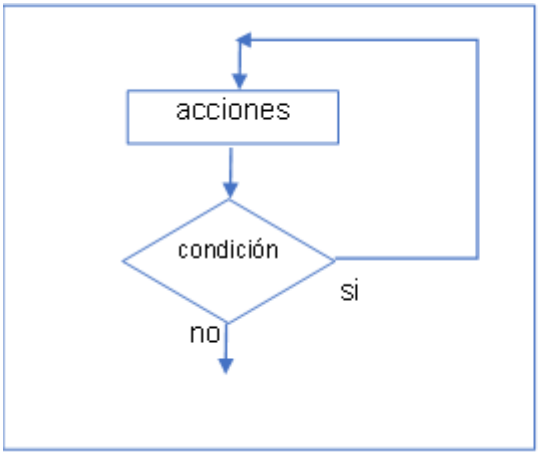


While condición **do**
Acción S1
Accion S2
.
.
End_while

- Do-while

Figura 7

Estructura DoWhile

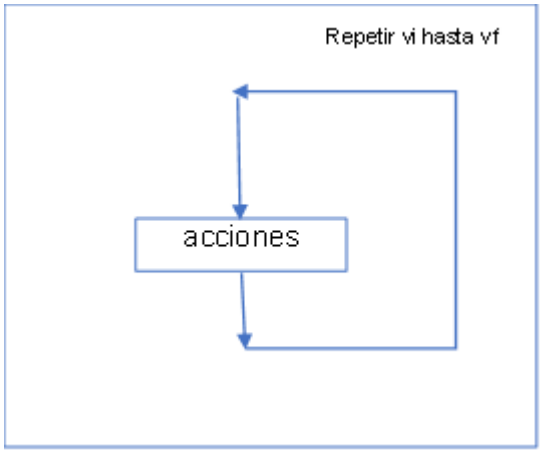


Do
Acción S1
Accion S2
.
.
While condición

- For

Figura 8

Estructura For



For v ← vi hasta vf (incremento incr) do
Acción S1
Accion S2
.
.
End_ For

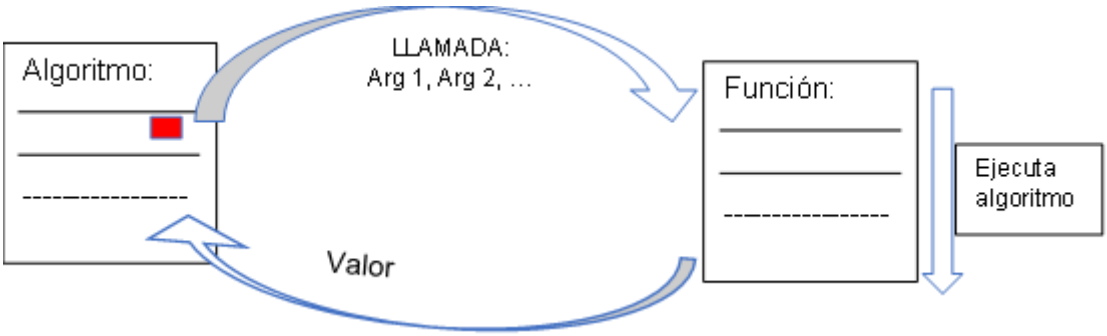
2.1.2.3 Funciones y Procedimientos

Como mencionamos anteriormente, los algoritmos nos sirven para resolver problemas. Pero en muchas ocasiones los problemas complejos pueden ser resueltos dividiéndolos en partes mas sencillas de resolver, estas partes pequeñas son resueltas por algoritmos mas pequeños, y luego estas soluciones son usadas para resolver el problema mayor. Estos algoritmos mas pequeños que luego son usados después son las Funciones y Procedimientos. A continuación, la definición de cada uno de ellos.

- Funciones. En la programación, Función es, Joyanes (2003) "Una operación que toma uno o mas valores llamados argumentos y produce un valor denominado resultado" (p. 207). Es decir, una función puede ser llamada por un algoritmo mediante unos argumentos y entregarle un valor calculado internamente.

Figura 9

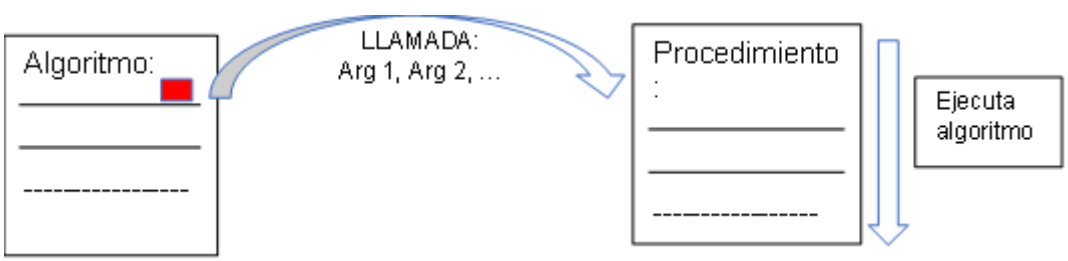
Secuencia de una Función



- Procedimientos. En la programación, Procedimiento es, Joyanes (2003) "Es un subprograma que ejecuta un proceso específico" (p. 215). Es diferente a una función, ya que este arroja un valor, en cambio el procedimiento ejecuta una serie de pasos, no devuelve un valor. El algoritmo llama al procedimiento en algún momento del código y se ejecuta el procedimiento y el algoritmo continua con su secuencia.

Figura 10

Secuencia de un Procedimiento

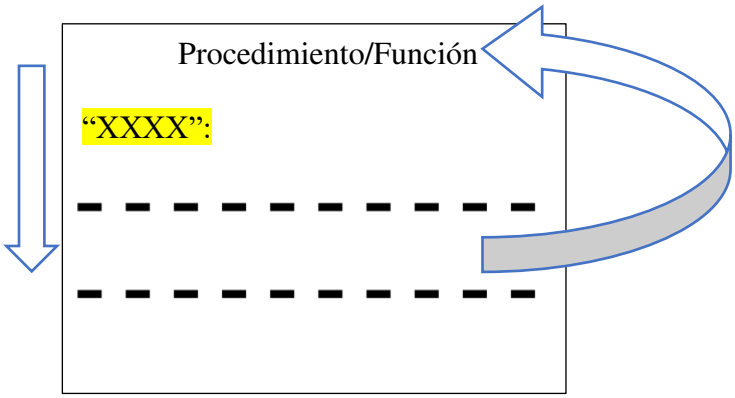


2.1.2.4 Algoritmos Recursivos

Como explicamos en las Funciones y Procedimientos, el algoritmo principal llama a las Funciones o procedimientos, recibe un valor o no (función o procedimiento), y luego continua con la secuencia de pasos del algoritmo. Ese es el proceso normal de funcionamiento de la programación llamada estructurada, pero existen funciones y procedimientos que se pueden llamar a si mismos. Estos son los llamados algoritmos recursivos. Joyanes (2003) lo define así: ”Un subprograma recursivo es un subprograma que se llama asi mismo” (p. 229). Se grafica de la siguiente manera:

Figura 11

Recursividad



Como vemos el subprograma se llama a si mismo en algún momento del código. Este procedimiento es muy intuitivo para implementar, es muy parecido a una estructura repetitiva, ya que cuando se llama a si mismo, lo que esta haciendo es volver a recorrer el código (como se indica en la figura). Es decir, un problema que necesita una estructura repetitiva de algoritmo para resolverse, también podemos usar recursividad

para resolverlo. Por ejemplo, en vez de usar un While, podemos usar Recursividad, claro que organizando el código para poder implementarlo. La elección entre estructura repetitiva o recursividad depende de la naturaleza del problema, ya que intuitivamente puede ser mas sencillo o efectivo resolverlo con recursividad o repetitivamente según sea el caso.

Un ejemplo clásico de aplicación recursiva es el algoritmo para los números factoriales (n!).

Por definición 1!=1, 2! = 1x2, 3! =1x2x3, 4!=1x2x3x4, 5!=1x2x3x4x5. De la última operación podemos deducir que 5!=4!x5. Luego, podemos generalizar n!= (n-1)!xn. En esta última expresión podemos leer así: El factorial de n es igual al factorial de n-1 por n. Aquí ya estamos llamando a la operación factorial asi mismo. Esto ya es recursividad. De esta manera estamos planteando el problema para ser resuelto de manera recursiva. A continuación, el Pseudocodigo:

```
integer function factorial (integer: n)
        begin
            if (n=0) then
                    return 1
            else
                    return (n*factorial(n-1))
            end if
        end_function
```

Integer function: indica que la función va a devolver un valor entero cuando la llaman.

Integer: n: indica que el argumento que recibe la función es de tipo entero.

Begin: comienzo del algoritmo de la función.

End_function: fin de la función

2.1.3 Algoritmos y Problemas.

Se describió anteriormente el concepto de algoritmo como un método para resolver problemas, pero hay una teoría que clasifica los problemas de acuerdo con la posibilidad de crear un algoritmo para resolverlos. Esta es la Teoría de la Computabilidad, que se explica a continuación:

2.1.3.1 Teoría de la Computabilidad.

Esta teoría clasifica los problemas en tres tipos:

- Problemas Computables. Son problemas donde, aparte si tengan solución o no, existe un algoritmo que es capaz de dar con la solución cuando existe, y también existe un algoritmo capaz de determinar que no hay solución cuando no la haya.
- Problemas Semicomputables. Son problemas donde, aparte si tengan solución o no, existe un algoritmo que es capaz de dar con la solución cuando existe, pero no existe un algoritmo capaz de determinar que no hay solución cuando no la haya.
- Problemas Incomputables. Son problemas donde, aparte si tengan solución o no, no existe algún algoritmo que las pueda resolver si hay solución, o determinar que no hay solución cuando no la haya.

Nos concentraremos en los problemas computables, ya que son los mas abundantes en la realidad. Cuando se modela una realidad, se procede de tal manera que exista algún algoritmo que pueda resolver el modelo creado. Un ejemplo de problema es justamente la secuenciación y programación de ordenes en un sistema industrial de JobShop Ensamble.

Los algoritmos pueden tener unos cuantos pasos, pero también pueden tener millones de pasos, resolverlos manualmente es materialmente imposible, por lo que los algoritmos han tenido la aplicación en los sistemas informáticos modernos. Estos han

hecho posible resolver problemas cuyo algoritmo de resolución podrían tener miles de pasos. Pero aun asi, la tecnología actual de los procesadores de computadora es finita, es decir, tienen un tope de procesamiento, por lo que también tenemos una limitación tecnológica que considerar. Por ejemplo, para un algoritmo con $1x10^{30}$ transacciones, la PC mas rápida del mundo (10^{15} operaciones x segundo) demoraría 31 millones de años en procesarla. Por este motivo surgió la Teoría de la Complejidad Computacional.

2.1.3.2 Teoría de la Complejidad Computacional.

Esta teoría clasifica los problemas computables de acuerdo con la posibilidad de que el algoritmo que los resuelva pueda ser físicamente factible con los recursos computacionales actuales (costo y tiempo).

- Clase de Complejidad P. Contiene aquellos problemas computables cuyo algoritmo de solución duran un tiempo polinómico (es decir que el tiempo de ejecución de un algoritmo es <= tiempo de una expresión polinómica dada en segundos) en una Maquina de Turing determinística (se refiere a las PC actuales). La mayoría de los problemas están en esta clasificación.

Ejm. Dado el problema de determinar el camino mas óptimo de un cartero por N ciudades, el algoritmo de solución sería de complejidad P, si el tiempo de ejecución del algoritmo creado para la solución es $<=50\ N^2 + N$ segundos.

- Clase de Complejidad NP. Es lo contrario a P, es decir, que el algoritmo de solución toma un tiempo que no es polinómico (mayor a $50\ N^2+N^2$ en el ejemplo anterior). No es rentable ni físicamente posible encontrar un algoritmo que lo pueda resolver en un tiempo razonable. Son los problemas mas interesantes para analizar. Además, hay tipo especial de problema Complejidad NP, que es el Tipo Complejidad NP-Completo.

- Tipo Complejidad NP-Completo. Este es un tipo de problema, donde, además de no hallar un algoritmo de resolución con tiempo razonable, la solución en si misma tampoco se puede encontrar en tiempo razonable. Es el tipo de problema mas complejo de resolver. Un problema clásico de este tipo son los problemas de Optimización Combinatoria.

2.1.3.3 Problemas de Optimización Combinatoria.

Empecemos por definir que es Optimización Matemática. Esta se define como la elección del mejor elemento dentro de un conjunto de elementos disponibles. Un caso particular es maximizar o minimizar una función real, eligiendo valores de entrada y calculando el valor de la función. Como se indica, en un problema de Optimización Matemática los valores pueden ser reales (enteros, fraccionarios, etc.). Un caso particular de Optimización es la Optimización Combinatoria. Yepes (2014) lo define asi "Son problemas de optimización en los que las variables de decisión son enteras, es decir, donde el espacio de soluciones esta formado por ordenaciones o subconjuntos de números naturales." Aquí explica una característica particular de los problemas de Optimización Combinatoria: la naturaleza discreta (números enteros) del planteamiento.

Los problemas de Optimización Combinatoria son recurrentes en la vida diaria, y para identificarlos hay una serie de parámetros que necesitan cumplir:

- Minimizar o maximizar una función en base a ciertas variables. Por ejemplo, si quiero ir caminando de Lince a Villa el Salvador, mi función es la distancia recorrida, y mi objetivo es minimizar esa función, es decir la distancia recorrida.

- Una lista de variables disponibles. Para ir de Lince a Villa el Salvador caminando, tengo muchas opciones, puedo escoger cualquiera. El problema es escoger el camino que me dé la menor distancia recorrida.
- Una vez encontrada la variable que minimiza la función, la variable es llamada el óptimo global. Es decir, el camino desde lince a Villa el Salvador que me produzca la menor distancia es el óptimo global.

Tomando en consideración estos 3 puntos, hay muchos problemas de la vida real que consideramos como optimización combinatoria, entre ellos tenemos:

- Viajante de Comercio. Como se mencionó, se trata de encontrar la ruta de una ciudad a otra que minimice la distancia recorrida.
- Problema de la Mochila. Consiste en escoger, de una serie de elementos posibles, aquellos objetos, tal que la sumatoria de pesos no exceda un valor máximo y que maximice la utilidad de llevar un objeto u otro.
- Problemas de Inventario. Decidir cuanto producir y cuanto almacenar de algunos productos, sujeto a restricciones.
- Distribución y programación de Ordenes en la Producción. Dado un conjunto de Ordenes, definir la secuencia de distribución de estas en un conjunto de máquinas, de manera de minimizar algún indicador. Este tipo de problema es justamente el que aborda la presente investigación. “Problema de Secuenciación de Ordenes en ambiente JobShop-Ensamble”.

Estos problemas pueden caer dentro de algunas de los tipos de complejidad que mencionamos anteriormente, por lo que en cada caso, los algoritmos que se hallan para resolverlos pueden no lograr encontrar la solución óptima en tiempo razonable, por lo que los algoritmos para tratar estos problemas se pueden clasificar en, Vidal (2013):

- Algoritmos Exactos: Son algoritmos que pueden resolver los problemas de optimización combinatoria y dar el resultado mas óptimo.
- Algoritmos Aproximados: Son algoritmos, que ante la imposibilidad de encontrar algoritmos que resuelvan al 100% un problema de optimización combinatoria, dan soluciones a un cierto % del óptimo.
- Algoritmos Heurísticos: Son algoritmos, que ante la imposibilidad de encontrar algoritmos que resuelvan al 100% un problema de optimización combinatoria, dan soluciones que no tratan de acercarse al optimo, sino tratan de buscar "buenas" soluciones. La presente investigación "Problema de Secuenciación de Ordenes en ambiente JobShop-Ensamble" es un problema que se trata con algoritmos heurísticos y se detallara a continuación.

2.1.4 Algoritmos Heurísticos.

Suarez (2011) lo define así:

> Se califica de Heurístico a un procedimiento para el que se tiene un alto grado de confianza en que encuentra soluciones de alta calidad con un costo computacional razonable, aunque no se garantice su optimalidad o su factibilidad, e incluso, en algunos casos, no se llegue a establecer lo cerca que se está de dicha situación. Se usa el calificativo de Heurístico en contraposición a exacto. (p. 45).

En esta definición el autor define eficazmente el concepto de Heurística relacionándolo con la utilidad de esta, es decir, dentro de los problemas computables, en la particularidad de los problemas NP-Complejos como son los de Optimización Combinatoria, el método mas eficaz de tratarlos es usando los algoritmos heurísticos, ya que da una buena solución dentro de los limites tecnológicos actuales, los cuales son impedimentos para llegar una solución óptima.

Hay varios intentos por clasificar los algoritmos heurísticos, en esta parte vamos a seguir la clasificación dada por Garcia (2006):

- Heurísticas Constructivas. El autor las define como: “aquellas que en cada paso añaden un elemento mas a una solución que no ha sido completamente construida”. Es decir, la solución se va construyendo paso por paso, interacción por interacción. Empieza con un % de la solución, y va aumentando progresivamente. Entre ellas tenemos:
    - Reglas de Prioridad. En cada interacción del proceso de solución, se escoge el camino por la opción que tenga la mejor prioridad, siendo esta fija en todo el proceso. Esta se da en base a un criterio preestablecido.
    - Heurística de Paso Múltiple. En cada interacción del proceso de solución, se escoge el camino por la opción que tenga la mejor prioridad, pero esta puede cambiar en el proceso.
    - Heurística BackTracking. Es un tipo de Heurística donde en cada interacción, se escoge una opción entre varias alternativas, pero puede regresar a la interacción anterior, si el escenario posterior no es satisfactorio.
- Heurísticas de Mejora Local. El autor las define como: “Estas parten de una solución cualquiera (no necesariamente aleatoria) y avanzan buscando el vecindario mas próximo, produciendo mejoras hasta que alcanzan un punto donde ningún elemento en el vecindario es mejor que la solución que ya se dispone”. Este tipo de heurística, al contrario de las constructivas, comienza la interacción con una solución inicial, y en cada paso va cambiando la solución hacia una mejor. Vecindario se llama a todas las opciones posibles que va encontrando en cada interacción. Un ejemplo de este tipo de Heurística es la Búsqueda Tabú.

- Metaheurísticas. Este tipo de heurística se caracterizan por estar inspiradas en fenómenos de la naturaleza como la física, la evolución y la biología. Este tipo de Heurística surgió debido a las limitaciones de las heurísticas constructivas y de mejora local, ya que estas, cuando al iterar encontraban un escenario menos optimo que el anterior, paraban el proceso, ya que había encontrado el "mas óptimo". Pero el problema estaba en que podría haber un mejor optimo varias interacciones mas adelante y que era descartado. Entre las Metaheurísticas mas conocidas tenemos a: Recocido Simulado, Algoritmos Genéticos y Colonización de Hormigas.

2.1.5 Problema de la programación en JobShop, motivo de la presente investigación.

Ahora vamos a tratar el tema de la presente investigación, que es Programación de Ordenes en Ambiente JobShop o también llamado JobShop Scheduling (Programación de Tareas). Vamos a plantear el problema tal como se le conoce en la literatura.

El problema del Job-Shop Scheduling es cuando tenemos un numero de Ordenes (n) por procesar en varias máquinas (m), donde cada orden consiste en un producto con varias operaciones a ejecutar en las máquinas según una ruta. Tenemos que encontrar la secuencia de órdenes en cada máquina de tal manera que minimice algún parámetro como puede ser: makespan (tiempo de terminación de todas las tareas), máxima tardanza, menor tiempo de paradas de máquina, etc. Exponemos un ejemplo.

Tabla 8

Tareas y sus Secuencias

Orden	Operación 1	Operación 2	Operación 3	Operación 4
A	2/6	3/5	1/3	4/9
B	3/2	1/2	4/3	2/3
C	1/2	2/5	3/4	4/5
D	4/2	1/2	2/1	3/2

Fuente: Elaboración Propia

Los valores en cada celda tienen el formato Maquina/Tiempo(h), ejm. 4/2 significa Maquina 4, tiempo 2 horas. El cuadro mostrado es un ejemplo clásico del problema JobShop.

Orden A. Ruta de Operaciones: Maquinas 2-3-1-4

Orden B. Ruta de Operaciones: Maquinas 3-1-4-2

Orden C. Ruta de Operaciones: Maquinas 1-2-3-4

Orden D. Ruta de Operaciones: Maquinas 4-1-2-3

Todas las ordenes tienen diferentes rutas y tiempos diferentes de ejecución. Tenemos 4 máquinas y 4 órdenes.

Ahora tomemos como ejemplo la Maquina 1, en esta máquina se van a procesar la tercera operación de la Orden A, la segunda operación de la Orden B, la Primera Operación de la Orden C y la segunda Operaciones de la Orden D, según el cuadro mostrado. Estas 4 operaciones pueden ser trabajadas de 4!=24 maneras distintas. Estas son: (Cada celda tiene el formato Operación-Orden, es decir 3A indica tercera operación de la Orden A)

Tabla 9

Máquina 1. Posibles secuencias de trabajo

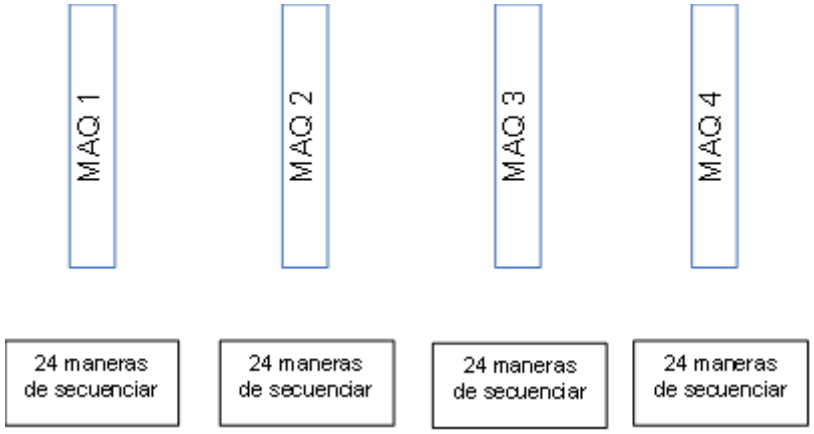
Sec	OPCIONES															
1	3A	3A	3A	3A	3A	..	..	..	..	..	..	..	..	..	..	
2	2B	2B	2D	2D	1C											
3	1C	2D	2B	1C	2B											
4	2D	1C	1C	2B	2D											
Opción	1	2	3	4	5	.	.	.	.	.	.	..	..	..	23	24

Fuente: Elaboración Propia

Estas son las 24 maneras que podemos secuenciar los trabajos en la máquina 1, y asi sucede con las otras 3 máquinas. Para cada una de ellas vamos a tener 24 maneras de secuenciar los trabajos. Entonces vamos a tener la siguiente situación:

Figura 12

Maneras de Secuenciar por Maquina



De cuantas maneras diferentes podemos secuenciar en las 4 máquinas. Por ejemplo, por cada secuencia que escojamos en la máquina 1, tenemos 24 secuencias posibles en la máquina 2 y asi para las 4 máquinas. En total, el número de secuencias posibles para las 4 máquinas seria 24x24x24x24=331776 maneras. Esto significa que, para el ejemplo de 4 órdenes para procesar en 4 máquinas, dadas la ruta indicada, tenemos 331776 maneras de programar los

trabajos en las 4 máquinas (4!x4!x4!x4!=$(4!)^4$). Esto podemos generalizar para N Ordenes y M máquinas. El número posible de secuencias posibles seria $(N!)^M$. Por ejemplo, si escogemos que el objetivo es encontrar la secuencia que minimice el Makespan (máxima tiempo de finalización de todos los trabajos), tendríamos que probar cada una de las 331776 secuencias, sumar los tiempos y escoger la secuencia que de menor tiempo de finalización. Se puede crear un algoritmo que pruebe estas 331776 secuencias rápidamente y nos muestre la secuencia con menor MakesPan.

Pero ahora hagamos la prueba con 20 órdenes y 10 máquinas, que es el número que podemos encontrar en la realidad industrial. Entonces el número de posibles secuencias seria = $(20!)^{10}$= 7.26 x 10^{183}. Si calculáramos cuanto demoraría la computadora actual mas veloz (10^{15} operaciones por segundo) en procesar el algoritmo para todas las posibles secuencias y mostrar la que tiene menor MakesPan, tendríamos como resultado 7.26 x 10^{183} / 10^{15} = 7.26 x 10^{168} segundos = 2.3 x 10^{155} millones de años. Cantidades de tiempo difíciles de imaginar, consideremos que la edad del universo se estima en 10^4 millones años.

Como se ve, el problema de JobShop Scheduling tiene un algoritmo que lo puede resolver para cantidades pequeñas, es computable, pero ese mismo algoritmo cuando se usa para cantidades mayores es computacionalmente imposible de resolver con la tecnología actual, por lo que a ese tipo de problema se le llama NP-Completo, es decir se sabe que existe un algoritmo que lo puede resolver exactamente, pero es físicamente imposible llevarlo a cabo. Esto dio origen, como ya se explicó, a la búsqueda de algoritmos heurísticos, que si bien no dan la solución exacta, pueden dar buenas soluciones satisfactorias.

Este problema es de real actualidad en las empresas de fabricación, ya que la continua competencia en los mercados hace necesaria mas diversificación y personalización de la producción, con cambios inesperados de las ordenes de los clientes , tanto en cantidad como

en fecha de entrega. Por lo que la programación cientos de las ordenes en un ambiente altamente dinámico y con decenas de máquinas a programar genera una necesidad real de mostrar soluciones.

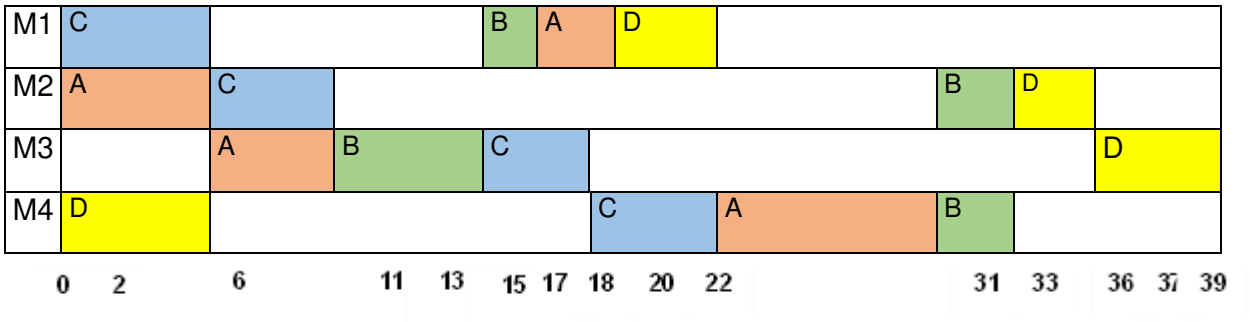
2.1.5.1 Ejemplo de un Problema de JobShop. A manera de ejemplo, se va a graficar en un diagrama Gantt una de aquellas secuencias, de manera aleatoria, para describir los elementos que se considera en un problema JobShop Ensamble. Se tienen las siguientes secuencias por máquina (al azar):

- Maq 1 --------- CBAD
- Maq 2 --------- ACBD
- Maq 3 --------- ABCD
- Maq 4 ----------DCAB

Esta es la secuencia que se escogió al azar para cada máquina, ejm. La maq 1 va a efectuar las ordenes CBAD en ese orden. Ahora lo graficamos en un Diagrama Gantt respetando la información del Cuadro de Precedencias.

Figura 13

Diagrama Gantt



Este es el resultado de escoger una secuencia al azar, sin ningún algoritmo en particular. De esta manera el MakesPan o Tiempo máximo de terminación es 39 horas,

que corresponde a la Orden D, asi en el tiempo 39 están terminados todos los trabajos. Mediante la aplicación de algún algoritmo Heurístico, se va a buscar disminuir el valor de 39, ya que este parámetro se usa para medir la eficiencia del Heurístico que se va a aplicar. Pero este parámetro (Makespan) no es el único que se usa para medir la validez de un Heurístico. Aparicio y Gonzales (2016) indica los siguientes parámetros:

- Tiempo máximo de terminación (Makespan) C_{max}= max{C_j}, donde j denota trabajo. Es el tiempo máximo de terminación de entre todos los trabajos.

MakesPan=39

- Tiempo total de terminación. $\sum Cj = \sum_{j=1}^{n} Cj$. Es decir la sumatoria de los tiempos de terminación de cada orden en su última operación.

En el ejemplo =Ca+Cb+Cc+Cd=31+36+22+39=128.

- Tiempo total de Terminación Ponderado. $\sum wjCj = \sum_{j=1}^{n} wjCj$; wj : importancia relativa de cada trabajo. Es una variante del anterior, donde a cada orden se le da un peso específico de acuerdo con su importancia. Ejm. Asignamos un peso a cada orden: A (2), B(5), C(2), D(1). El Tiempo total de terminación ponderado seria: 31*2+36*5+22*2+39*1=325.
- Carga total de las máquinas. $\sum_{max}^{k} C = \sum_{j=1}^{m} C$. Es decir, la sumatoria de los tiempos de terminación máximo en cada máquina.

En el ejemplo: M1= 20, M2=37, M3=39, M4=33. Total =20+37+39+33= 129.

- Tardanza Máxima. Lmax= max {Lj}= max {Cj-dj}; dj: fecha de entrega del trabajo j.

Para este indicador vamos a considerar fechas de entrega (horas) para cada Orden.

Tabla 10

Fechas de Entrega por Orden

Orden	Fecha Entrega(Horas)
A	20
B	40
C	18
D	45

Comparando las fechas de terminación de cada orden con sus fechas de entrega:

A: 31-20=11

B: 36-40= -4

C: 22-18=4

D: 39-45= -6

Donde vemos que las Ordenes A y C están con retraso. Las ordenes B y D están a tiempo. La tardanza máxima = max{11,-4,4,-6}= 11.

- Número de trabajos tardíos. $\sum Uj = \sum_{j=1}^{n} Uj$, donde Uj=0 si Lj<=0 y Uj=1 si Lj>0. Es decir el numero de trabajos que se terminaron después de la fecha de entrega prometida. Usando los datos del indicador anterior. El número de Trabajadores tardíos seria: 1+0+1+0= 2.

- Costo máximo. Fmax= max{fj(Cj)}, donde f1,f2,f3,…,fn son funciones de costo dadas. En este caso se crea una función de costos en base al tiempo de terminación de cada orden, para encontrar cuanto cuesta el uso de las máquinas para el esquema dado. Para el ejemplo podemos considerar la siguiente función de costos : fj(Cj) = S/10 x Cj. Es decir cada hora cuesta 10 soles. En el ejemplo:

Ca=31, entonces F=S/10x31=S/310.

Cb=36, entonces F=S/10x36=S/360.

Cc=22, entonces F=S/10x22=S/220.

Cd=39, entonces F=S/10x39=S/390.

El costo máximo seria de S/390.

- Costo total. $\sum Fj = \sum_{j=1}^{n} Fj(Cj)$. Es una variante del anterior, donde se suman todos los costos del tiempo de terminación. Para el Ejemplo seria: 310+360+220+390=S/1280.

Estos indicadores mostrados no son los únicos que se puede considerar para evaluar el resultado del Heurístico. Existe total libertad para formular los indicadores que la industria requiera. Los indicadores pueden ser creados usando diversos criterios de acuerdo con los objetivos de la empresa. Esto permite tener la capacidad de tomar buenas decisiones en programación.

Medrano (2007) muestra algunas aplicaciones en la industria del problema de JobShop Scheduling:

- Planificación de Proyectos. En este campo es vital la precedencia de los proyectos y minimizar el máximo tiempo de terminación de estos.
- Programación de Horarios y Sistema de Reservaciones. De Aplicación en los Servicios Educativos y Aeropuertos. En este caso el objetivo es normalmente es planificar la mayor cantidad de tareas en un tiempo determinado.
- Programación de Fuerza de Trabajo. Aplicable en los hospitales y talleres. Se planifica siempre con restricciones de horario.
- Secuenciación de tareas en talleres. De amplia aplicación en la Industria, tema de la presente investigación.

Aunque ya hemos planteado el problema del JobShop Scheduling de manera práctica, hay un planteamiento de manera formal para el problema: Vicentini y Puddu (2003):

> Sean dados M un conjunto de máquinas, J un conjunto de trabajos y O un conjunto de operaciones. Para cada operación i ε O tenemos ligadas un trabajo j_i ε O al que pertenece y una máquina m_i ε M en la que debe realizarse consumiendo un tiempo p_i ε R(reales) ininterrumpido y positivo. Además, es dada una relación de precedencia binaria ς que descompone O en cadenas, una para cada trabajo. Encontrar un tiempo de 0comienzo s_j para cada operación i ε O tal que se minimiza el Makespan, definido como max $_{i \varepsilon o}(s_i + p_i)$, sujeto a las siguientes restricciones
>
> 1. $s_i >= 0$ para todo i ε O
> 2. sj >= si+ pi para todo i,j ε O con i ς j
> 3. sj >= si+ pi o si>= sj+ pj para todo i,j ε O con mi=mj . (p. 7)

2.1.5.2 Tipos de Algoritmos Heurísticos para el Problema de JobShop. Como se mencionó, existen Algoritmos Heurísticos que pueden mostrar buenas soluciones para el problema del JobShop, en vista de la imposibilidad de brindar soluciones exactas por la limitante tecnológica. Los Heurísticos pueden estar destinados a minimizar el MakesPan, el tiempo total de terminación, Tardanza Máxima, Costo máximo u algún otro indicador que se crea conveniente. Por eso hay bastante campo de acción para la investigación en el campo del problema JobShop, ya que existen muchos parámetros para minimizar en la elaboración del Algoritmo Heurístico. Algunos Heurísticos se muestran a continuación:

1. Heurísticas de Mejora Local

- Algoritmo Grasp. Viene de las iniciales Greddy Randomized Adaptive Search Procedures, es un procedimiento aleatorio de búsqueda. Parte de una solución inicial y va construyendo mejores soluciones en base a un criterio definido. Gonzales (2015) dicta los pasos básicos: “
    - Construcción de una solución inicial.
    - Aplicación de las modificaciones básicas aleatorias para construir nuevas soluciones.
    - En el momento que se encuentre una solución de mejora, se toma como mejor solución y se vuelve al paso anterior. “ (p. 47)
- Búsqueda Tabú. En el proceso anterior esta involucrado un proceso aleatorio, por lo que hay un riesgo que en cada interacción se vuelva a tomar una solución ya evaluada. En el algoritmo Búsqueda Tabú, no existe ese problema, porque se crea un almacén virtual de opciones ya escogidas, esto es para que ya no vuelva a ser tomadas en una nueva interacción. De esta manera se evita que se caiga en un bucle o que se encuentre un óptimo local pensando equivocadamente que es el mejor, ignorando posibles otros óptimos locales mejores.

2. Metaheurísticas.

- Recocido Simulado

Este algoritmo simula el proceso de enfriamiento térmico de un material. En el enfriamiento térmico se parte de una situación de alta temperatura, donde las moléculas están vibrantes y pasa a una situación de baja temperatura lentamente con el fin de que las moléculas adquieran una mínima energía. En

la simulación las posiciones de las moléculas son representadas por la secuencia de tareas y la energía es la variable que se quiere minimizar.

- Algoritmos Genéticos

Es un método que usa la teoría de la evolución para simularla en un entorno Job Shop. Parte del concepto de que la herencia produce sujetos de mejores características por medio de la mutación. En la simulación los sujetos (padres) son las diferentes secuencias de fabricación en un JobShop, y a estos padres se les asigna un cierto indicador que sirve como determinante si un sujeto o sus hijos pasan a la siguiente generación (interacción). La detención siempre se determina por un máximo número de interacciones definido.

3. Algoritmos Constructivos:

- *Shifting Bottleneck Procedure (SBP) o Cuello de Botella.*

En este algoritmo, el problema del JobShop Scheduling es tratado como un problema de cuello de botella. Sigue un procedimiento de plasmar el planteamiento en un gráfico Pert. Dividir el problema en subproblemas por máquina, identifica el cuello de botella, encuentra secuencias para estas máquinas cuello de botella, revisa si esta secuencia se puede mejorar o no, luego procesa las máquinas que faltan secuenciar. Si las nuevas máquinas ya no se pueden mejorar, entonces el algoritmo finaliza y se obtiene la secuencia para todas las máquinas. Romano (2016) explica claramente el procedimiento SBP para un ejemplo de JobShop Scheduling.

- Reglas de Despacho: Es una Heurística constructiva, es decir que va construyendo la solución paso por paso. En cada paso escoge una opción en base a algún criterio de optimización. Este criterio puede ser, Sippper y Bulfin (1998):

- TPC (Tiempo de procesado mas corto). Se escoge la tarea que, sumados los tiempos de todas sus operaciones, es la de menor valor.
- PEPS (primero en entrar, primera en procesar). Se procesa la operación que llego primero.
- FEC (Fecha de entrega mas cercana). Se procesa el trabajo con la fecha mas urgente.
- MTR (mayor trabajo restante). Este es un indicador, al contrario de los 3 anteriores que son fijos al inicio del heurístico, es dinámico. Ya que en cualquier momento del algoritmo, se escoge la tarea que tiene la mayor suma de tiempos de proceso que aun faltan programar.
- HLG (Holgura). Se programa la tarea con menos holgura. Holgura = fecha de entrega de la tarea – (tiempo de proceso que falta programar de la tarea + la fecha actual).
- RC (Razón critica). Se programa la tarea con la menor razón=Holgura /(Fecha de Entrega de la Tarea-Fecha Actual).
- HLG/OP (Operación con Holgura). Se programa la tarea con la menor razón=Holgura/(número de operaciones que quedan a la tarea).

Como se aprecia, los criterios de optimización son variados, y pueden idearse muchos otros en línea con el objetivo de la programación. Estos criterios de optimización tienen una relación con los parámetros para validar un algoritmo como se vio anteriormente (Makespan, Tardanza máxima, etc.). La ventaja del Heurístico es que se puede simular varios criterios de optimización y medir varios parámetros de validación y tomar la decisión en consonancia con los objetivos empresariales. Vamos a profundizar en este tipo de Heurística ya que es el tipo que vamos a usar en la presente investigación.

Sippper y Bulfin (1998) muestran el algoritmo para resolver un problema de JobShop:

"Define:

A=conjunto de máquinas ociosas.

J_k =índice del último trabajo programado en la máquina k

U_k =conjunto de trabajos

H_k = tiempo de terminación del trabajo que se esta procesando en la máquina k

u_{it} = urgencia o prioridad del trabajo i en el tiempo t. Mientras mas pequeña mejor.

s_{ij} = tiempo de inicio de la operación j del trabajo i

c_{ij} = tiempo de inicio de la operación j del trabajo i

Paso 0. Inicialización: t=0; H_k=0,k=1,2,….m; A={1,2,….m}; U_k={i|operación 1 del trabajo i esta en la máquina k, i={1,2,….n}; s_{ij}=c_{ij}=0,i=1,2,….,n; j=1,2,..m. Se va al paso 4.

Paso 1. Se incrementa t; sea t= $\min_{k=1,m,k\,\varepsilon\,A} H_t$ y K={k|H_k = t}.

Paso 2. Se encuentra el trabajo o trabajos que terminan en el tiempo t y las máquinas que quedan libres. Se hace i^t={i|J_k=i,kεK} y A=A ∪ K.

Paso 3. Se determinan los trabajos listos para programarse en cada máquina; sea U_k = {i|el trabajo i usa la máquina k y todas las operaciones de i antes de k están terminadas}, k=1,2,…,m. Si U_k =ϕ para k=1,2,….,m, se detiene; el programa está completo. Si U_k =ϕ para todo k ε A, ninguno de los trabajos que esperan tiene una máquina libre, de manera que no se pueden programar trabajos en este momento. Se va al paso 1.

Paso 4. Para cada máquina ociosa, se intenta programar un trabajo; para cada k ε A con $U_k \neq \phi$, sea i^* el trabajo con la mejor prioridad, $u_{i^*t} = \min_{i\varepsilon Uk} u_{it}$. Se programa el trabajo i* en la máquina k; se hace J_k=i*, s_{i^*k} = t, c_{i^*k} =t+$p_{i^*j(k)}$,

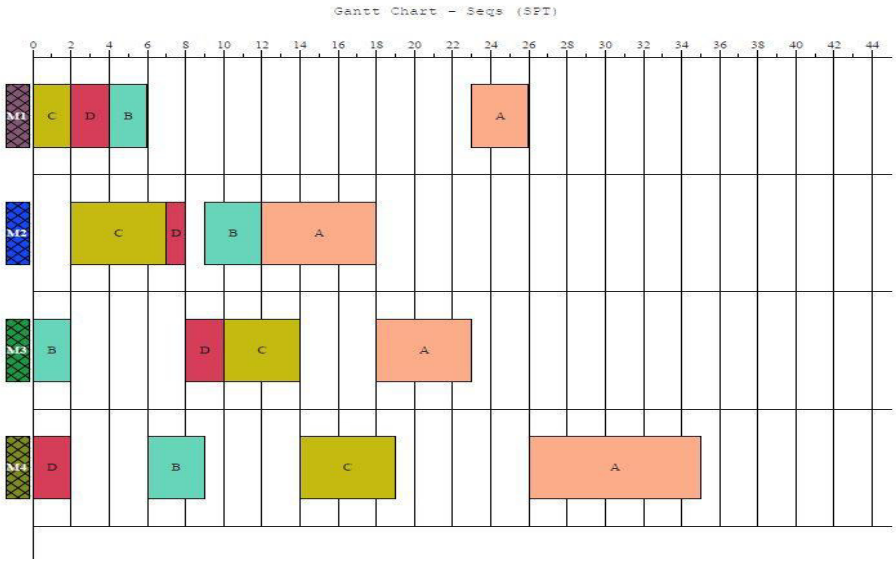
$H_k=c_{i*k}$. Se elimina el trabajo programado de U_k, $U_k \leftarrow U_k - \{i^*\}$ y la máquina de A, $A \leftarrow A-\{k\}$. Una vez que se ha examinado toda $k \varepsilon A$ con $U_k \neq \phi$, se va al paso 1.

Finalizado el algoritmo, lo tiempos de inicio y terminación en cada máquina se usa para construir el Diagrama Gantt" (pp. 458-459)

2.1.5.3 Aplicación práctica de un Heurístico en un Problema de Job Shop. Para efectos de comparación, vamos a aplicar el Algoritmo de Despacho en el caso ejemplo que pusimos. Para ellos vamos a usar el Software LEKIN (Stern School of Business, NYU) con regla de despacho usamos el TPC (Tiempo de procesado mas corto).

Figura 14

Diagrama Gantt Software LEKIN



Ahora vamos a calcular los mismos parámetros que hicimos para la secuenciación aleatoria y compararlos.

Tabla 11

Comparación de Parámetros Aleatorio vr Regla Despacho

Parámetro	Secuencia Aleatoria	Heurística regla de Despacho. Regla=TPC. Software LEKIN
Secuencia de los trabajos por Maquina:	Maq 1 = CBAD Maq 2 = ACBD Maq 3 = ABCD Maq 4 = DCAB	Maq 1 = CDBA Maq 2 = CDBA Maq 3 = BDCA Maq 4 = DBCA
MakesPan(Tiempo máximo de terminación)	39	35
Tiempo Total de Terminación	128	76
Tiempo Total de terminación Ponderado	325	178
Carga Total de Maquinas	129	102
Tardanza Máxima	11	15
Número de Trabajos Tardíos	2	2
Costo Máximo	390	350
Costo Total	1280	760

Como vemos, el algoritmo regla de despacho ha producido secuencias diferentes para cada máquina que el método aleatorio y tiene mejores resultados en todos los parámetros de medición excepto en la tardanza máxima. Por lo que el rendimiento y eficacia de este algoritmo es un hecho factico. El peor resultado en el indicador de tardanza máxima se explica por lo siguiente: el algoritmo regla de despacho ha producido las siguientes fechas finales de terminación para los trabajos {A=35, B=12, C=19, D=10} con secuencias diferentes para cada máquina en comparación con el procedimiento aleatorio. Las fechas de entrega para cada trabajo son: { A=20, B=40, C=18, D=45} y los tiempos totales de fabricación por trabajo son: {A=23,B=10,C=16,D=7}. EL algoritmo regla de despacho con regla TPC, da prioridad a los trabajos con menor tiempo de fabricación, por lo que el algoritmo va a tomar a las

órdenes B,C y D como prioridad en detrimento de la orden A. Por lo que la orden A esta al final de todas las secuencias generando la mayor tardanza (A: 35-20=15).

Usar otra regla de despacho en este algoritmo generará otros valores para los parámetros mostrados, por lo que la evaluación de estos determinará la secuencia mas adecuada de acuerdo con los objetivos empresariales.

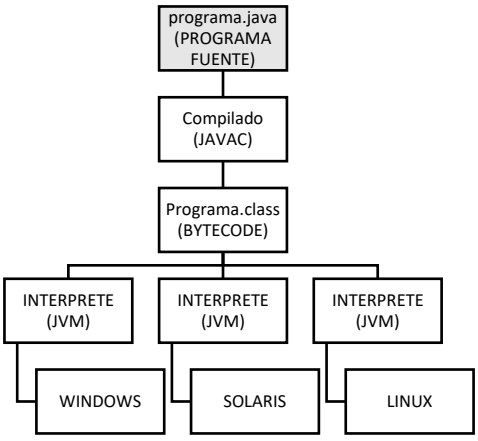
2.2 Lenguajes de programación Java y Gestor de Base de Dato Oracle

2.2.1 Java

Es un lenguaje de Programación de propósito general (para todo tipo de uso), concurrente (múltiples tareas al mismo tiempo) y orientado a objetos. Fue creado originalmente para ambientes de desarrollo web pero su uso se ha extendido a ambientes Cliente Servidor Empresariales. La novedad de este lenguaje con respecto a otros, es que cuando se crea el código, este no necesita ser compilado en cada máquina para que pueda correr, siempre y cuando, cada maquina tenga instalada el Java Virtual Machine (JVM). Es decir, existe un JVM diferente para Windows, Linux o Solaris.

Figura 15

Ambiente Java



Java tiene varios niveles en su arquitectura. Por ejemplo, para que una máquina puede correr un programa Java (código compilado extensión .class) necesita: el JVM y algunos componentes que permitan la ejecución en la máquina, a todo esto se denomina JRE (Java Runtime Enviroment). Y para crear un programa Java (código con extensión .class), necesitamos el JDK (Java Development Kit), que provee herramientas de desarrollo para programas java. En el JDK esta el compilador, que permite convertir un archivo .txt a un archivo .class. Si bien con el JDK se puede crear y ejecutar programas java, es muy limitado para poder crear aplicaciones empresariales por lo que se creó el JSE (Java Standard Edition) con mas atribuciones como aplicaciones de escritorio. Pero como el mundo empresarial trabaja en red y necesita aplicaciones cliente/servidor, se creó el Java JEE destinado a aplicaciones grandes y a servidores. La versión java JEE 7 es la que se usa en la presente investigación. Esta versión de Java no necesita licencia para el uso en programación de aplicaciones. Este lenguaje soporta la recursividad y es escalable desde aplicaciones cliente-servidor hacia aplicaciones web. Estas fueron las razones principales para escoger Java como lenguaje de la presente investigación.

Java es uno de los lenguajes de programación mas populares, tal como lo muestra la página de Oracle: https://www.java.com/es/about/.

- El 97% de los escritorios empresariales usa Java.
- 9 millones de desarrolladores de Java en todo el mundo.
- 3 mil millones de teléfonos móviles ejecutan Java.
- 125 millones de dispositivos de televisión ejecutan Java.

Como Java es un software orientado a Objetos, haremos una introducción a la filosofía orientada a objetos.

2.2.2 Filosofía Orientada a Objetos

Esta filosofía busca orientar la programación al mundo real, es decir, al contrario de la programación estructurada, secuencial y con módulos, la programación orientada a objetos modela el desarrollo de sistemas, tal como es el mundo real. Kendall y Kendall (2011) menciona: " Lo que distingue a la programación orientada a objetos (y por ende al análisis y diseño orientado a objetos) de la programación clásica es la técnica de colocar todos los atributos y métodos de un objeto dentro de una estructura autocontenida, la clase en sí. Éste es un acontecimiento familiar en el mundo físico." (p. 282). Esto es una base primordial de esta filosofía, ya que justamente Java al crear las clases, al mismo tiempo establece los atributos y los métodos. Todo en un paquete completo, al contrario de la estructurada que los métodos se programan aparte y se llaman después.

Objeto. Son objetos relevantes para el Sistema, como clientes, productos, etc.

Clase: Los objetos comunes o parecidos forman parte de otro mas general, eso es una clase. Ejm. Juan y Pedro son Objetos, pero los 2 son personas, así que existen una clase llamada Persona, donde Juan y Pedro serian las instancias de la clase Persona.

Como mencionó Kendall y Kendall (2011), cuando se declara la clase, al mismo tiempo se declara los atributos y los métodos. Los atributos son las características que todos los objetos de esa clase poseen y los métodos son las operaciones que esa clase puede ejecutar. Ejm. Clase Carros, 2 atributos son Color y Marca. 1 método seria Arrancar.

Herencia:

Es un concepto también tomado de la realidad, del Padre a Hijo. Una clase hereda de otra cuando es una réplica que hereda los mismos atributos y métodos, pero que le puede

adicionar su “toque” personal. Ejm- La clase Auto hereda de una Clase mas general Vehículo. La clase Auto, además de los atributos que heredó de su clase padre Vehículo, tiene el atributo adicional Estilo. Este concepto es el usado para la reusabilidad de componentes en Java.

Polimorfismo.

Como ya se vio en los métodos, estos son funciones de las clases. Pero aquí se introduce el concepto que es polimorfismo, que significa que un método puede tener diferentes operaciones dependiendo de la clase en la cual esta. Ejem. En la vida real En la vida real, el método arrancar, tiene el mismo nombre para un avión, que para un carro, pero ambos ejecutan el método de modo diferente.

Sobrecarga de métodos

Es un caso especial de polimorfismo, donde el mismo método con el mismo nombre responde diferente, debido a que los parámetros de entrada son diferentes, ejm. Si se hace un depósito al banco, puede resultar 2 acciones, que depositen todo el dinero, o que entregan vuelto y depositen. Los dos son el mismo método, depositar al banco, pero dependiendo de la cantidad que se ingrese, ejecuta diferentes operaciones. Es como un condicional pero englobado en un solo método.

Encapsulamiento.

Ya se vio el concepto de atributos y métodos, el encapsulamiento significa que las los atributos de las clases no pueden ser accesados desde afuera a no ser que sea por algún método sobre la clase. Están encerrados y solo se accesan con una llave que son los propios métodos de la clase. Ejm. Si quiero modificar el atributo color de la clase Vehículo, no podría hacerlo directamente y pintarlo, si no tendría que apelar al método pintar de la clase, y este método sería el encargado de pintarlo.

2.2.3 Oracle

Es un Sistema de Gestión de Base de Datos de tipo relacional. Esta presente en 98 de las 100 empresas mas ricas del mundo. Es un Sistema de licencia, pero tiene una versión gratuita: Oracle Express Edition. (Oracle XE), la cual tiene las siguientes limitaciones: Tamaño máximo de la Base de datos: 4 GB, 1 instalación por ordenador y 1 GB de RAM como máximo. Se escogió este Sistema Gestor por su gran potencia y seguridad a pesar de ser gratuita. En el caso de necesidad de mas tamaño en la base de datos, se puede optar otros Sistemas Gestores libres como MySQL y PostgreSQL. Se puede hacer la migración de la base de Datos en Oracle XE hacia otros Sistemas Gestores usando el Oracle SQL Developer

2.3 Metodología de Desarrollo de Software.

Se usará el modelo en Cascada para el desarrollo del Software, haciendo uso de Diagramas UML propios de la metodología de programación orientado a objetos combinado con Diagramas propios de la metodología Estructurada. por lo que los pasos serán los siguientes:

1. Análisis y Definición de Requerimientos.

   Se define el análisis de requerimientos como el proceso para determinar la necesidad del cliente, en este caso el ambiente industrial de Job Shop. También establecer el dominio de la aplicación (sector de trabajo) incluyendo el modelo de negocio (Programación de Ordenes en ambiente Job Shop), y para ello se aplica técnicas de casos de uso en el ámbito de la industria.

2. Diseño de la Aplicación.

   En esta etapa se diseña la Arquitectura del Sistema tomando en consideración los requerimientos de Software y Hardware. Se usará el modelo Vista Controlador (MVC). Se hará uso de múltiples diagramas para graficar el Modelo del Sistema y el Flujo de

Datos, describiendo la manera en que el Sistema va a lograr los requerimientos. Se diseñará la base de datos, dentro el modelo MVC. Aquí es donde se aplica los algoritmos recursivos y heurísticos a nivel conceptual. El algoritmo recursivo se usará para tratar la Lista de Materiales-Ensamble y el Algoritmo Heurístico para la Programación de Ordenes. Se establece además el diseño de las Ventanas y Menús de la Aplicación.

3. Implementación del Sistema.

Se procede a desarrollar el sistema mediante la codificación del aplicativo en Java y la base de datos en Oracle, tomando como base el Diseño de la Aplicación previamente elaborado. Se usará el entorno de Desarrollo NetBeans 8.0.2.

4. Pruebas de la Aplicación.

Se prueba el Sistema con variados escenarios de lista de materiales, productos, maquinas, horarios y órdenes para validar el correcto cumplimiento de los requerimientos.

2.4 Marco Conceptual

Programación de Ordenes

Se refiere a la actividad de colocar las Ordenes de los Clientes, disgregados en productos terminados y componentes, en las Maquinas o personas para su transformación en Productos elaborados. Es una actividad neurálgica dentro del proceso de Planeamiento y Control de la Producción, necesaria para cumplir con los pedidos del cliente. Gaither y Frazier (2000) menciona con respecto a esta actividad: “ Los trabajadores y las maquinas son tan flexibles que es posible asignarlos y reasignarlo a muchas ordenes diferentes. En un entorno

tan flexible, variable y cambiante como este, los programas para cada centro de trabajo deben ser específicos y detallados para poner orden" (p. 440). Lo que indica el autor guarda certera relación con la realidad peruana de los procesos industriales, reafirmando mas el sentido práctico de la presente investigación.

JobShop.

Es termino ingles para la Producción Intermitente, que significa un tipo de producción donde el material pasa por diferentes maquinas en cualquier orden de acuerdo al producto requerido. Es decir los productos tienen diferentes rutas de producción. Sippper y Bulfin (1998) dice al respecto: "Es difícil programar la producción intermitente. Existen (n!)m programas posibles para una planta con n trabajos y m maquinas. Aun para valores pequeños de n y m, este número es enorme. Incluso el problema mencionado de tres trabajos y cuatro maquinas tiene más de 1000 programas; un problema de 10 trabajos tendría más de 10^{26}." (p. 456). Como explica el autor, la programación de ordenes es de naturaleza combinatoria, reforzando el necesario tratamiento con la Heurística.

Algoritmos Heurísticos.

Joyanes (2003) describe a los algoritmos Heurísticos como métodos de resolución de problemas mediante pasos definidos, precisos y finitos pero que implican algún juicio o interpretación. En contraposición a los algoritmos normales que no tienen ese componente juicioso. Reafirmado, este juicio o interpretación son, en el caso particular de la presente investigación, los criterios de prioridad de las órdenes. Esos criterios son los que van a determinar la toma de decisiones del algoritmo Heurístico llegando no a una solución exacta al problema, sino a una buena solución.

Algoritmos Recursivos:

Son algoritmos que se llaman a si mismos, si bien su interpretación es básica de entender su aplicación en términos computacionales consume bastante tiempo, pero las desventajas son mayores que las ventajas ganadas en algoritmo eficiente y claro. Joyanes (2003) menciona: " se puede usar la recursividad como una alternativa a la iteración. Una solución recursiva es normalmente menos eficiente en términos de tiempos de computadora que una solución iterativa……. En muchas circunstancias el uso de la recursión permite a los programadores especificar soluciones naturales, sencillas, que serian, en caso contrario, difíciles de resolver" (p. 537). El autor reafirma la gran utilidad de la recursividad. Por lo que esta técnica es la que se usa en la presente investigación para resolver el problema de la lista de materiales (Árbol Padre-Hijo).

Gestión Industrial

También llamado Administración de la Producción y de las Operaciones. Son todos los procesos y actividades que se llevan a cabo para fabricar productos y entregarlos al cliente. Forma todo lo relacionado a la producción de lo que denominamos Empresa. Gaither y Frazier (2000) los describe así: " es la administración del sistema de producción de una organización, que convierte insumos en productos y servicios" (p. 5). Además el autor considera la producción como un Sistema, donde todas sus partes se interrelacionan entre si con el objetivo de Producir. Mas adelante describe el concepto de efectividad operacional como el acto de llevar a cabo operaciones mejor que los competidores. Y justamente el trabajo de investigación guarda consonancia con este concepto, al agilizar el proceso de programación se está ganando efectividad operacional logrando asi competitividad.

III MÉTODO

3.1 Tipo de Investigación

El Tipo de investigación será de tipo Aplicada por el desarrollo del Sistema Informático para mejorar la Gestión Industrial, con diseño Cuasi-Experimental debido a que la Muestra no es al azar, sino controlada y con enfoque cuantitativo.

3.2 Población y Muestra

La población se idéntica como todo el proceso de Gestionar la Empresa de Fabricación, y la Muestra se identifica dentro del proceso de Gestión Industrial en el Área de Planeamiento. Y para tal efecto la muestra es no probabilística e intencional considerando una cantidad adecuada de 15 muestras de trabajo. Es decir, se procederá a elaborar el programa de producción para 15 casos, antes y después del uso del Sistema Informático.

3.3 Operacionalización de Variables

3.3.1 Variables Independientes

Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos.

Tabla 12

Variable Independiente

Variable	Definición Conceptual	Dimensiones	Indicadores	Unidad de Medida
Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos.	Sistema Informático elaborado en base a Algoritmos Heurísticos y recursivos considerando ambientes Job Shop Ensamble	Sistema Informático	Activo	SI-NO

Fuente: Elaboración Propia

3.3.2 Variables Dependientes

Gestión Industrial.

Tabla 13

Variable Dependiente

Variable	Definición Conceptual	Dimensiones	Indicadores	Unidad de Medida
Gestión Industrial	Todos los procesos necesarios para la programación en una fábrica industrial	Programación de la Producción	Tiempo de Programar la Producción	Minutos
		Gestión de Lista de Materiales	Tiempo de Registrar Lista de Materiales y Visualizar	Minutos
		Gestión de Rutas del Producto	Tiempo de Registrar Rutas de Producción y Visualizar	Minutos

Fuente: Elaboración Propia

3.4 Instrumentos

Para la recolección de información se usará la técnica de la Observación, y se usara como instrumentos la Computadora, libreta de apuntes y cronometro, estos elementos serán usados para recolectar el nivel de Mejora logrado por la implementación del Sistema Informático mediante la toma de tiempos de los indicadores.

3.5 Procedimientos

Para los análisis estadísticos, inferenciales y resultados se usará Excel y el Programa SPSS.

3.6 Análisis de Datos

Se seguirá la siguiente secuencia para la contrastación de Hipótesis:. Se tomará la data mediante los instrumentos antes de la aplicación del Sistema Informático (pretest)y se volverá a tomar la data luego de la aplicación del Sistema Informático(pos-test). Luego se determinará si las datas recogidas por el instrumento son normales o no, mediante la prueba de Normalidad de Shapiro-Wilk (ya que tenemos menos de 50 datos). En el caso de salir distribución Normal se aplicará la prueba paramétrica t de Student, en caso contrario se usará la prueba No Paramétrica de Wilcoxon para la Hipótesis. En las pruebas de Hipótesis para cada caso, se demostrará si el Sistema Informático es causa o no de la Mejora en la Gestión Industrial.

3.7 Consideraciones Éticas

La investigación, al abordar el tema de la programación de órdenes en un ambiente industrial JobShop Ensamble, permite dinamizar la gestión de las órdenes de producción, esto genera rapidez en la respuesta al cliente y despacho oportuno de los pedidos. Al tener un sistema informático, la información en tiempo real permite una rápida toma de decisiones en ventas. El dar un buen servicio al cliente, redundara en un mayor beneficio para la empresa productora, ya que se mantiene la cartera de ventas y la vigencia empresarial. El trabajo de los colaboradores estará asegurado impulsando el consumo de la sociedad. También el programa permite el uso efectivo de recursos, ya que el adelanto en la programación permite abastecerse de insumos anticipadamente generando menos mermas por cambios de programación y reprocesos. Esto influye en una menor solicitud de recursos al medio ambiente en periodos posteriores.

IV RESULTADOS

Se procede a desarrollar la metodología de desarrollo del Sistema Informático.

Propuesta de Sistema Informático de Programación de Ordenes

4.1 Enfoque

En el marco teórico y conceptual se ha repasado los conceptos de JobShop Scheduling, así como los Algoritmos Heurísticos y Recursivos. Como se explicó, el problema del JobShop Scheduling tiene una solución satisfactoria con la aplicación de los Algoritmos Heurísticos. El problema del JobShop Scheduling es una modelación del conocido caso de la programación de órdenes en un ambiente industrial, donde tenemos rutas en máquinas, tiempos de procesado y órdenes de producción. Estas variables son modeladas en el problema del JobShop Scheduling según lo mostrado en la teoría. Se ha revisado varios casos de investigaciones abordando este problema, así como la teoría pertinente del mismo. Pero el problema del JobShop Scheduling, tal como ha sido planteado en la teoría, no toma en cuenta una variable de la realidad que es el Ensamble. Esto es, la unión de 2 subproductos, para formar otro subproducto u producto final. Este concepto es de plena aplicación en las empresas de fabricación en todos los campos de la industria.

4.2 Comparación Planteamiento Clásico JobShop vs Propuesta

A continuación se muestra un cuadro que compara el planteamiento clásico con la propuesta de la investigación, para el caso del Job Shop Ensamble.

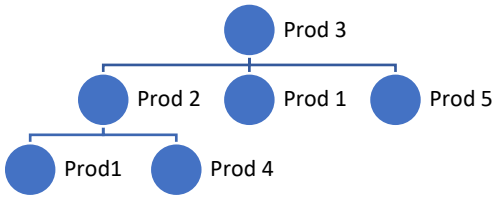
Figura 16

JobShop: Planteamiento Clásico vr Planteamiento Propuesta

Planteamiento Clásico Job Shop (A)

		Op1	Op2	Op3
Productos Independientes	Prod 1	M1	M3	M2
	Prod 2	M3	M2	M1
	Prod 3	M2	M3	M1
	Prod 4	M1	M2	M3
	Prod 5	M2	M3	M1

Propuesta JobShop Ensamble (B)



		Op1	Op2	Op3
Productos Relacionados	Prod 1	M1	M3	M2
	Prod 2	M3	M2	M1
	Prod 3	M2	M3	M2
	Prod 4	M1	M2	M3
	Prod 5	M2	M3	M1

Fuente: Elaboración Propia

En lado B de cuadro se muestra la propuesta de la investigación, que consiste en resolver el problema de JobShop cuando hay un producto como P3, que depende de la fabricación previa de los productos P2, P1 y P5, para poder ser ensamblados y formar el producto P3. De igual manera para el producto P2, que necesita del abastecimiento previo de los productos P1 y P4.

Con estas nuevas condiciones reales que se presenta en la industria, se propondrá un algoritmo Heurístico de Despacho para encontrar una solución adecuada al caso JobShop

Ensamble. Considerando además, de que un producto puede volver a pasar por una máquina mas de 2 veces (Prod3=M2-M3-M2), aspecto que el planteamiento clásico no permite.

Además, la propuesta clásica propone que las máquinas están disponibles al inicio del programa, mientras que la propuesta de investigación considera que no siempre se encuentran disponibles las máquinas al inicio del programa.

A continuación un cuadro que resume las diferencias entre el Planteamiento Clásico del JobShob Scheduling y la Propuesta, que ya fueron comentadas en los antecedentes de la investigación.

Tabla 14

Parámetros JobShop: Planteamiento Clásico vr Planteamiento Investigación

PARÁMETROS	Planteamiento Clásico	Propuesta de Investigación
Trabajos con Rutas Diferentes.	SI	SI
Ensamble de Productos.	NO	SI
Productos procesados en una misma máquina mas de 1 vez.	NO	SI
Máquinas disponibles al inicio.	SI	SI
Máquinas ocupadas al inicio.	NO	SI
Cada Máquina solo puede procesar un trabajo a la vez.	SI	SI
Cualquier número de máquinas y de trabajos.	SI	SI
Se programa el trabajo con la mejor prioridad.	SI	SI
Método Push (se programa cada trabajo tan rápido como se pueda).	SI	SI

Fuente: Elaboración Propia

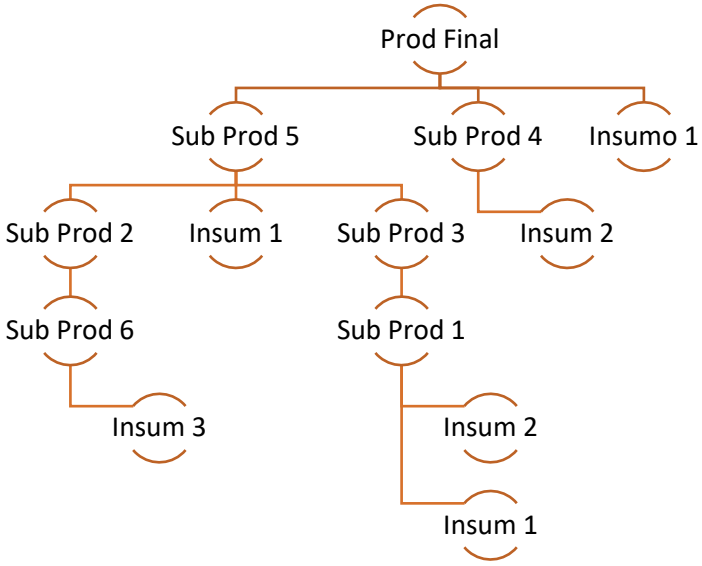
Como se muestra en la tabla, los parámetros: Ensamble de Productos, Productos procesados en una máquina mas de una vez y máquinas ocupadas al inicio son aspectos de palpable realidad en las industrias, por lo que la investigación contribuirá en forma práctica al desempeño industrial.

4.3 Resumen de la propuesta de Desarrollo.

El planteamiento del Algoritmo Heurístico propuesto parte de una lista de materiales previa, que debe estar elaborada. Esta lista de materiales (estructura del producto) es de propósito general, con todas las opciones posibles que se dan en la industria. Se propondrá un algoritmo recursivo que pueda tratar esta lista de materiales y poder utilizarse en el heurístico del JobShop. Tal como se índice en los antecedentes de la investigación, el algoritmo recursivo es el mas adecuado para tratar configuraciones tipo Árbol o Padre-Hijo, que es el caso de la Lista de Materiales y Ruta de Productos. A continuación, se grafica el Esquema de la Lista de Materiales.

Figura 17

Esquema Lista de Materiales



Esta lista de materiales es el input para el Algoritmo heurístico que dará una solución para el JobShop-Ensamble. Como se mencionó en el Marco Conceptual, el algoritmo Heurístico es el mas adecuado, considerando las limitaciones computacionales actuales, para

tratar los problemas de Complejidad NP-Completo como es el caso del Problema de JobShop Ensamble.

A continuación un esquema muy general de la Metodología de Trabajo:

Tabla 15

Esquema General de la Investigación

<table>
<tr><th>Entradas</th><th colspan="2">Desarrollo del Software</th><th>Resultado</th></tr>
<tr><td>-Lista de Materiales (Ensambles)
-Ruta de Productos (Maquinas y Operaciones por donde pasa cada producto)</td><td>Normalización y Programación de Algoritmo Recursivo</td><td></td><td rowspan="2">Programación de Ordenes en ambiente Job Shop (Gantt de Ordenes por Maquina Calendarizados)</td></tr>
<tr><td>-Lista de Maquinas (Disponibles o no)
-Lista de Productos
-Características de Productos (Formatos, tiempos, etc.)
-Lista de Operaciones
-Orden de Producción (Estados)
-Horarios de Producción</td><td>Normalización</td><td>Programación de Algoritmo Heurístico</td></tr>
</table>

Fuente: Elaboración Propia

En el cuadro se muestra que las informaciones de Lista de Materiales y Ruta de Productos serán tratadas con el algoritmo recursivo de manera previa para poder usadas como input del algoritmo Heurístico que genera el Gantt de Ordenes. Las informaciones de Máquinas, Productos, Operaciones, Horarios y Ordenes de producción son las entradas, junto a la lista de materiales y Ruta de Productos, de Algoritmo Heurístico que generará el Gantt de Programación de Ordenes.

4.4 Modelamiento del Negocio actual

4.4.1 Identificación del Problema y Proceso involucrado

Como se detalló en la justificación de la Investigación, el problema inicial que se vislumbra es la lentitud actual para elaborar los programas de producción que se envían a las

áreas productivas. Se necesita dinamismo para responder rápido a las necesidades del cliente por consiguiente el Sistema informático de Programación de Ordenes esta plenamente sustentado para así poder iniciar el desarrollo de todo el Software. Este sistema, como se mencionó, esta suscrito a un ambiente industrial de Job Shop-Ensamble, que es el ambiente preponderante en la realidad industrial peruana. Así mismo, las industrias en general son muy similares en el funcionamiento a nivel macro, donde las diversas áreas actúan como un Sistema Interrelacionado con el objetivo de fabricar productos para el Cliente

El programa de producción es generado por el Área de Planeamiento y Control de la Producción, por lo que a continuación se muestra un alcance de la posición de esta área en la Empresa y entender el funcionamiento macro de una empresa industrial objeto de la presente Tesis. También se mostrará un organigrama genérico de una empresa de fabricación y también la cadena de valor de la misma, mostrando los procesos comunes en todas las empresas de fabricación, a modo de ubicar el proceso y el área donde el Sistema Informático tendrá aplicación.

Figura 18

Organigrama de una Fabrica

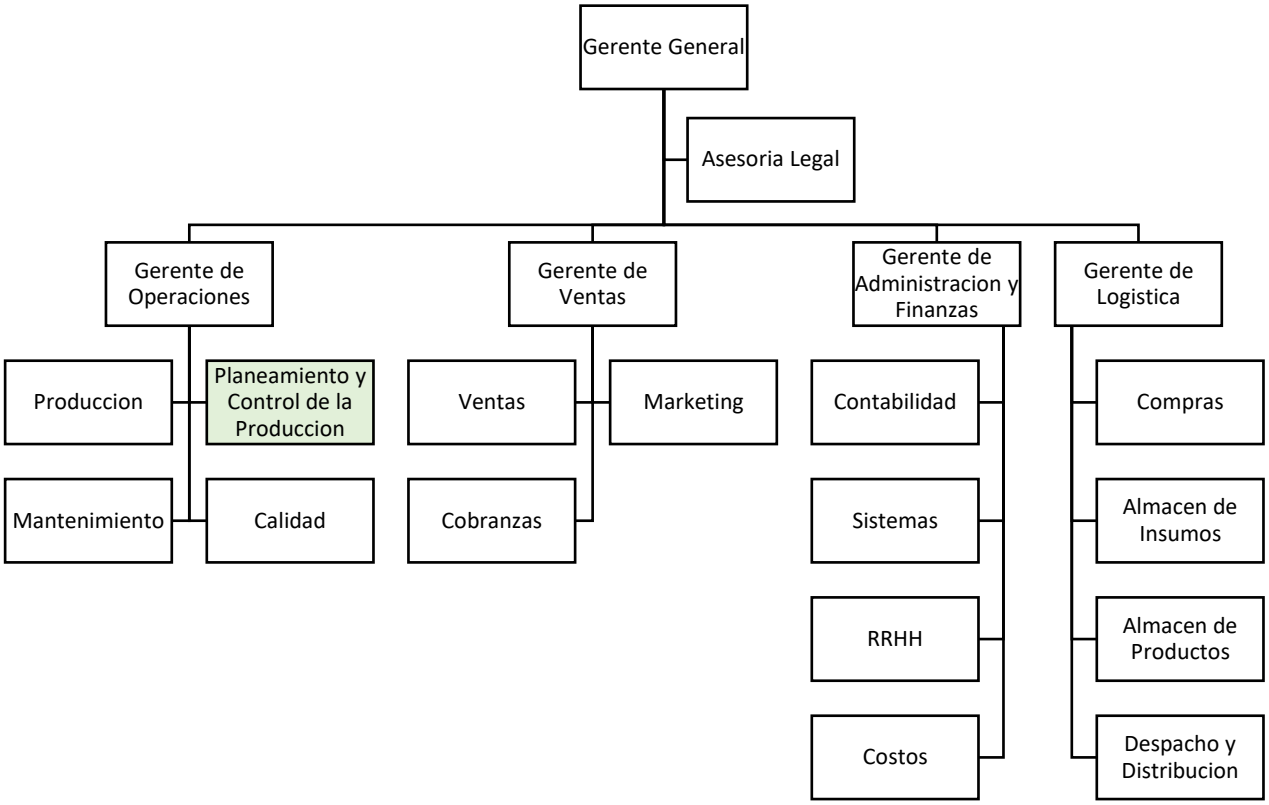


Figura 19

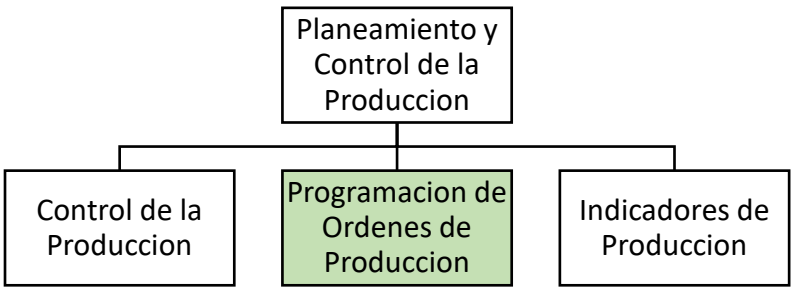
Cadena de Valor de una Fabrica

Funciones y Actividades del Área de Planeamiento y Control de la Producción

El área de Planeamiento y Control de la Producción tiene varias funciones asociadas, entre las cuales esta generar el programa de producción para las áreas de fabricación.

Figura 20

Funciones del Área de Planeamiento y Control de la Producción



Se muestra una de las funciones es Programación de la Producción, que es el proceso de estudio donde el problema ha sido detectado: lentitud en generación del programa de producción.

Describimos el proceso de Actividades para la Programación de la Producción.

1. El proceso inicia con el envío de Ordenes de Venta de parte de los vendedores hacia el área de Planeamiento de la Producción por correo. Esto se da todos los días a cualquier hora del día.

2. PCP revisa si el producto de la Orden de Venta es nuevo, si es así, solicita a Ingeniería la lista de materiales y la ruta del Producto.

3. PCP genera un Numero de fabricación correlativo con los productos para cada Orden de Venta en una Hoja Excel, indicando además la prioridad de la Orden de Fabricación. Normalmente es un trabajo diario de actualización.

4. Mientras tanto PCP mantiene actualizado constantemente la lista de materiales y la ruta de producto que Ingeniería les entrega en formato Excel. También solicita el horario de máquinas disponibles semanalmente a Mantenimiento. Para ello PCP mantiene archivos Excel actualizados todos los días.

5. Elabora un cuadro Excel con los productos de la orden de fabricación donde esta la información de los procesos y maquinas por donde pasan los productos. Esto se actualiza todos los días.

6. PCP en forma manual en Hoja Excel asigna los productos de las OF a las maquinas disponibles respetando la información de Lista de Materiales, Ruta de Productos y Prioridades de las Ordenes. Se elabora un Programa semanal por máquina, donde se por Maquina todo el listado de Productos y Componentes que pasaran por esa máquina en una

Semana de Producción. Esto se hace el último día de la semana para programar la semana entrante. Normalmente el trabajo de hacer la programación dura todo el día. Se entrega el programa a los supervisores de Producción.

7. En el transcurso de la semana ya programada, ingresan varias órdenes urgentes todos los días. Entonces en vista de la urgencia y para no esperar el tiempo de actualización manual de la programación, se envían directamente a Producción y se empieza a fabricar.

8. También a mitad de semana se establece 1 día de actualización para el Programa de Producción, actualizando las cantidades faltantes de las Ordenes, donde se vuelven a hacer los pasos descritos.

9. Se lleva un control de la programación para elaborar indicadores de cumplimiento.

4.4.2 Diagramas DFD del Proceso Involucrado

Vamos a usar los Diagramas de Flujos de Datos para ilustrar el proceso actual de Elaboración del Programa de Producción para evidenciar las causas y soluciones para el problema descrito: Lentitud en la elaboración de los programas de producción.

Figura 21

Diagrama de Contexto Actual

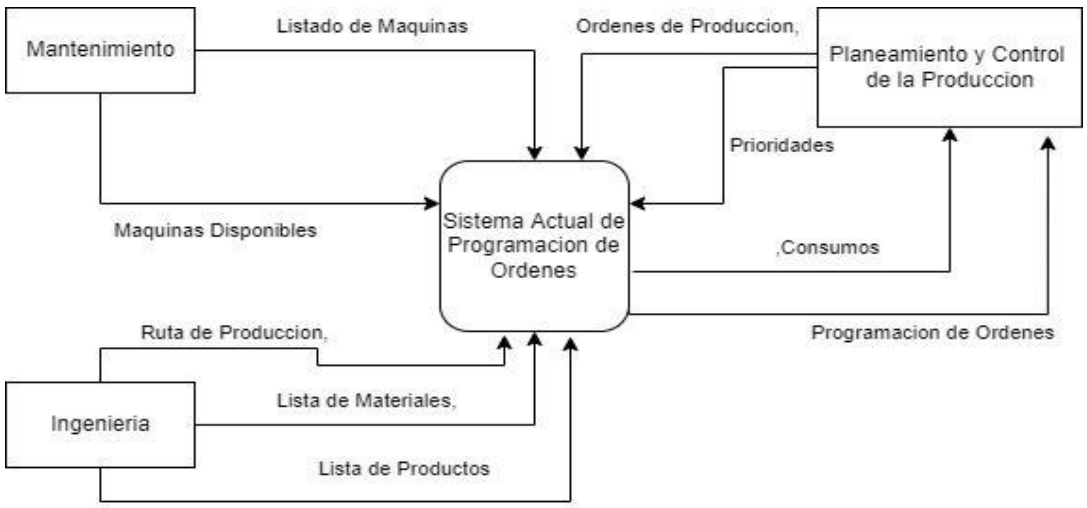


Figura 22

Explosión Sistema Actual de Programación de Ordenes

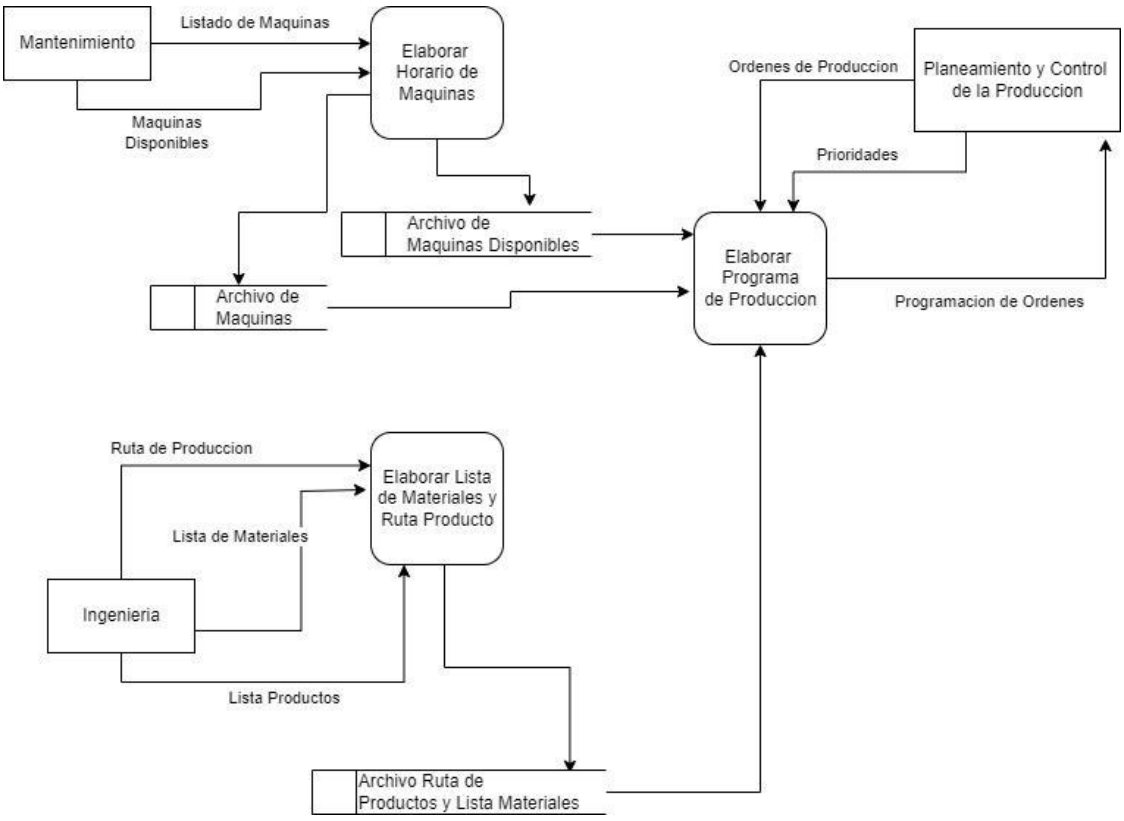


Figura 23

Proceso Elaborar Horario de Maquinas

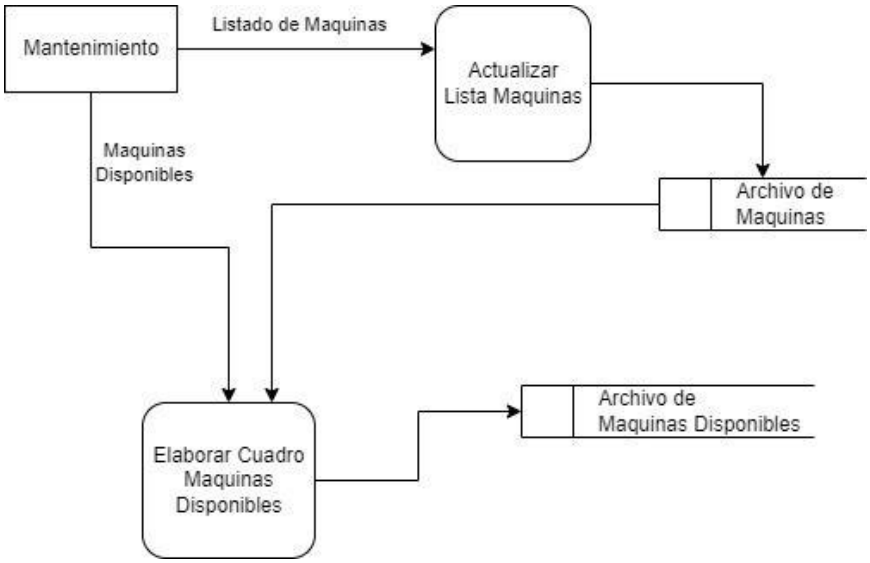


Figura 24

Proceso Elaborar Lista de Materiales y Ruta del Producto

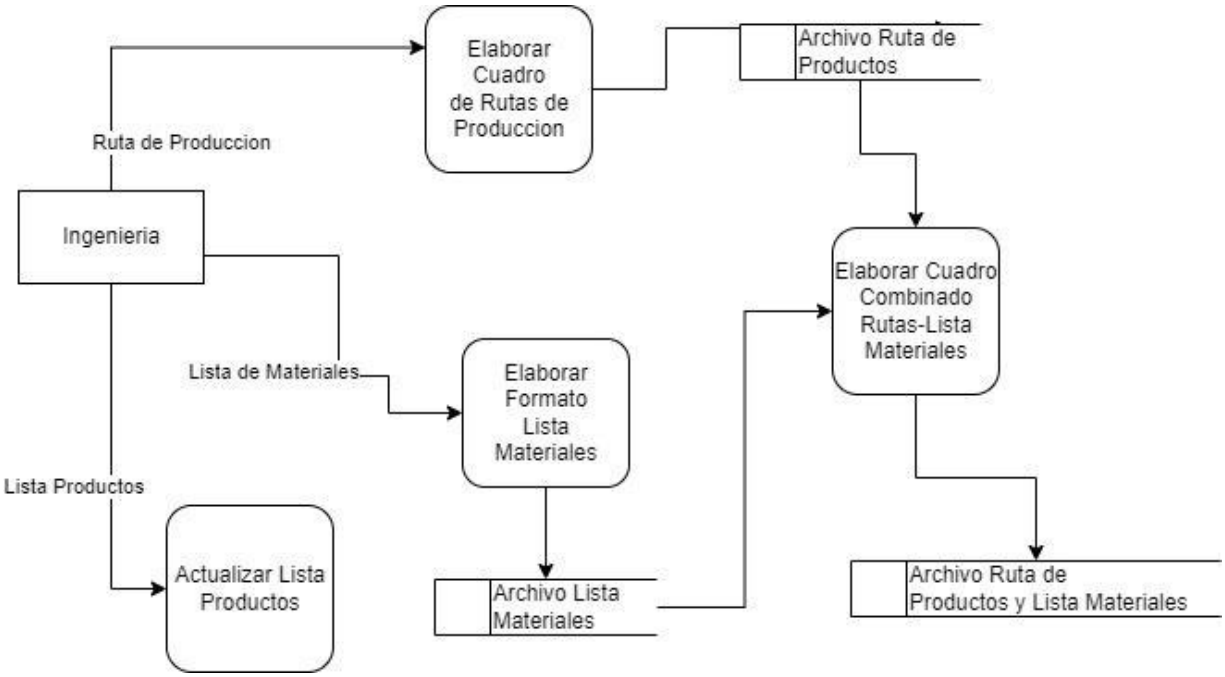
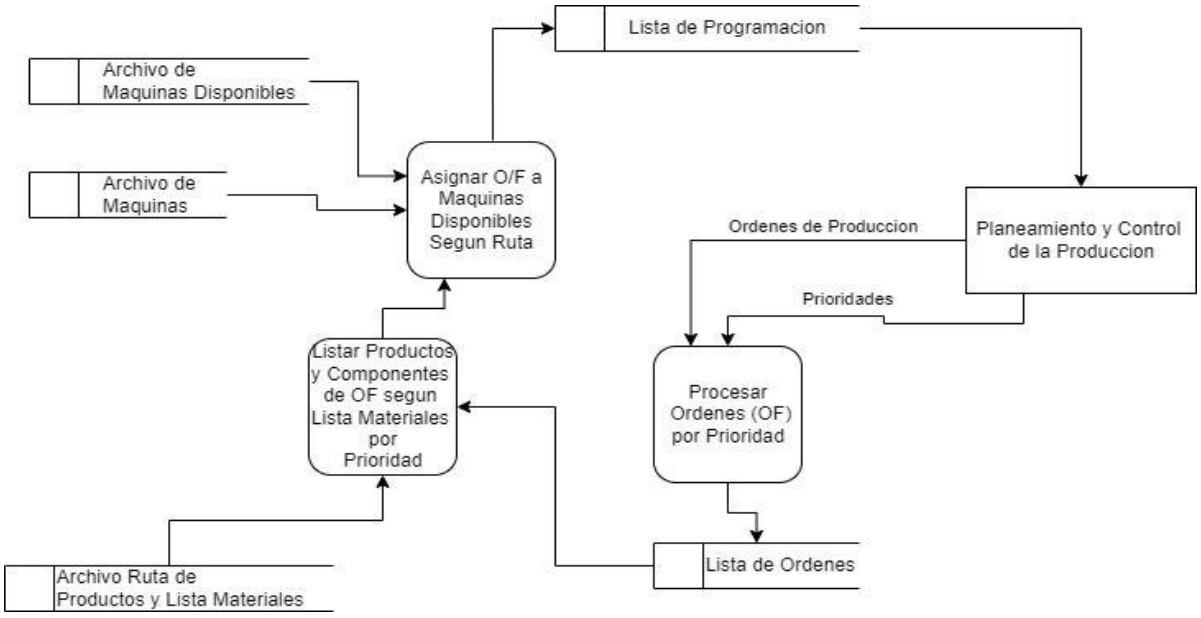


Figura 25

Proceso Elaborar Programa de Producción

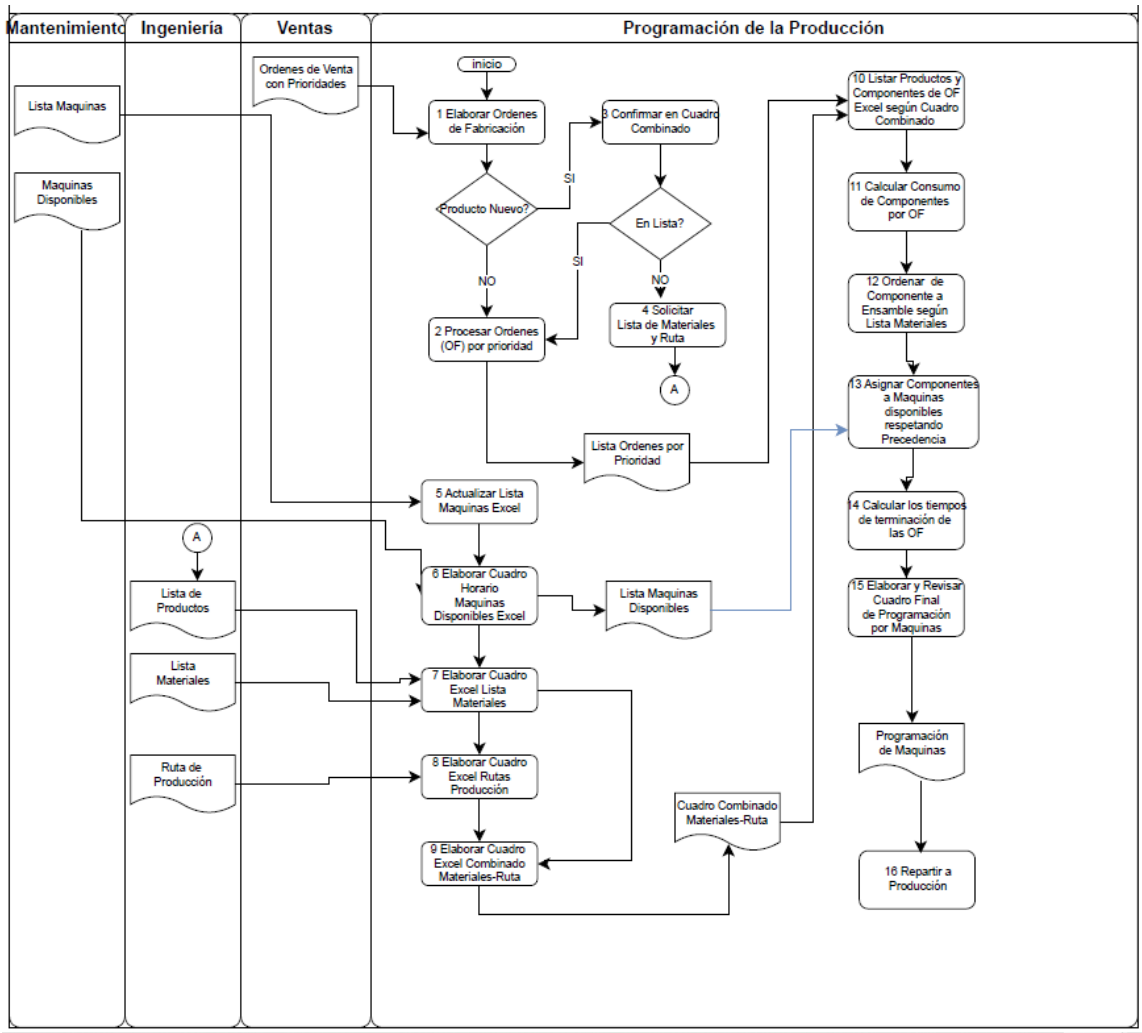


4.4.3 Diagrama de Flujo del Proceso Involucrado

En este último DFD se muestra los procesos donde mas tiempo de procesamiento manual se necesita: Listar Productos y Componentes de OF según Lista Materiales por prioridad y el proceso Asignar O/F a Maquinas Disponibles según Ruta. Ya que son procesos ejecutados en Excel de manera manual. Si bien se hace uso de fórmulas, el proceso de asignación y prioridades se hace de manera prueba y error, debido a la complejidad de la asignación, donde se toma varias variables a considerar, como prioridad, precedencia, componentes, maquinas disponibles. A continuación se muestra un Diagrama de Flujo donde se detalla los procesos seguidos para los DFD mostrados anteriormente.

Figura 26

Flujograma Actual del Proceso de Planeamiento



4.4.4 Descripción de Problemas hallados

Se muestra a continuación un cuadro con la valoración de tiempo que el área de planeamiento usa para ejecutar cada uno de los procesos, así como observaciones del proceso.

Tabla 16

Cuadro de Tiempos de los Procesos de Planeamiento

Proceso	Ejecutante	Tiempo	Frecuencia	Observación
1 Elaborar Ordenes de Fabricación	PCP	0.25 hr	2 veces x semana	Las Ordenes de venta se reciben por correo.
2 Procesar Ordenes (OF) por prioridad	PCP	0.25 hr	2 veces x semana	Ventas indica la prioridad que desea, ya sea por Fecha de entrega o por Fecha mas antigua de entrada.
3 Confirmar en Cuadro Combinado	PCP	0.1 hr	2 veces x semana	Se confirma si el producto esta en la base de datos de PCP
4 Solicitar Lista de Materiales y Ruta	PCP	0.1 hr	2 veces x semana	En el caso del producto de la OF sea nuevo.
5 Actualizar Lista Maquinas Excel	PCP	0.25 hr	Diario	PCP recibe un Excel con el formato de Mantenimiento y PCP lo transforma en el formato interno de Excel de PCP
6 Elaborar Cuadro Horario Maquinas Disponibles Excel	PCP	0.25 hr	Diario	PCP recibe un Excel con el formato de Mantenimiento y PCP lo transforma en el formato interno de Excel de PCP
7 Elaborar Cuadro Excel Lista Materiales	PCP	0.5 hr	Diario	PCP recibe un Excel en diagrama de árbol por parte de Ingeniería y lo transforma en filas y columnas para PCP
8 Elaborar Cuadro Excel Rutas Producción	PCP	0.5 hr	Diario	PCP recibe un Excel en diagrama de árbol por parte de Ingeniería y lo transforma en filas y columnas para PCP
9 Elaborar Cuadro Excel Combinado Materiales-Ruta	PCP	1 hr	Diario	PCP hace un consolidado en Excel, donde para cada producto muestra los componentes, tiempos y las maquinas por donde pasa.
10 Listar Productos y Componentes de OF Excel según Cuadro Combinado	PCP	1.5 hr	2 veces x semana	PCP busca los productos de la OF en el cuadro combinado y los lista en excel, y así para cada uno de los productos de todas las OF para programar.
11 Calcular Consumo de Componentes por OF	PCP	1 hr	2 veces x semana	PCP calcula en el cuadro anterior los consumos de componentes según formulas. Respetando la lista de materiales.
12 Ordenar de Componente a Ensamble según Lista Materiales	PCP	1.5 hr	2 veces x semana	PCP acomoda el cuadro de productos OF (Cuadro combinado) para poner en primer lugar los componentes que se van a fabricar primero para luego los ensambles.
13 Asignar Componentes a Maquinas disponibles respetando Precedencia	PCP	2 hr	2 veces x semana	PCP en base al cuadro ordenado anteriormente asigna manualmente en otra Hoja Excel para cada máquina, los componentes a fabricar, poniendo hora inicio y hora fin, respetando la precedencia y la lista de materiales. Empieza por los componentes para luego los ensambles hasta terminar todos los productos de todas las OFs.
14 Calcular los tiempos de terminación de las OF	PCP	1.5 hr	2 veces x semana	PCP en base al cuadro anterior calcula los tiempos estimados de finalización de las OFs para enviárselas después al área de Ventas.

15 Elaborar y Revisar Cuadro Final de Programación por Maquinas	PCP	1.5 hr	2 veces x semana	Consolida y revisa el Cuadro final de Programación de Ordenes.
16 Repartir a Producción	PCP	0.25 hr	2 veces x semana	Reparte la programación a las áreas de producción.

Fuente: Elaboración Propia

Como se muestra en la tabla, los procesos 10,11,12,13,14,15 que son los que se ejecutan para hacer la programación de ordenes semanal, duran en total 9 horas de trabajo en Hojas Excel. Esta demora de 9 horas (1 turno de trabajo) genera que las actualizaciones u ordenes urgente se obvie su paso normal por PCP y se envíe directamente a planta generando desorden y desinformación, muy frecuente en la industria nacional. Además los trabajos diarios 7,8,9 duran 2 horas que son los referentes a la elaboración de Lista de Materiales y Hojas de Ruta. Se indica diario porque siempre hay cambios en la ruta de producción o en los tiempos de fabricación. La actualización y mejora de tiempos es una constante en las fábricas de producción. Los trabajos 7,8,9 y los trabajos 5,6 se elaboran con informaciones proporcionadas por las áreas de ingeniería y mantenimiento respectivamente. Es un trabajo doble, ya que las hojas de Excel que esas áreas envían a PCP, son transformadas redundantemente al formato de PCP Excel. Una realidad muy palpante en las industrias peruanas.

4.4.5 Propuesta de Solución

Como se mencionó en la justificación de la investigación, el proceso de candelarizar los trabajos por maquina usa mucho tiempo de preparación, como se muestra en al cuadro anterior (9 horas) por lo que la implementación de un sistema que haga el proceso de manera automática se hace muy necesario. A continuación se enumera los problemas y solución del método actual:

Tabla 17

Problemas Actuales y la Solución Planteada

Problema	Propuesta	Involucrados	Objetivos del Sistema
Tiempo excesivo de procesamiento para la Programación de Trabajos por Maquina (procesos 10,11,12,13,14,15)	Un sistema informático que procese automáticamente los inputs necesarios. Se usaría un algoritmo Heurístico tal como se describió en la justificación.	PCP	Reducir el tiempo de programación de los trabajados desde 9 horas a 2 seg.
Redundancia en el tratamiento de la información de Maquinas (listado y disponibilidad)	En el Sistema informático se ingresaría el Listado de Máquinas y la disponibilidad.	Mantenimiento	Mantenimiento ingresaría directamente al Sistema, sin tratamiento doble de la información y en una base de datos de acceso instantáneo
Redundancia en el tratamiento de la información de Lista de Materiales y Ruta del Producto	En el Sistema informático se ingresaría la Lista de Materiales, Ruta del Producto y el mismo Sistema generaría el Cuadro Combinado de Lista de Materiales y Ruta del Producto usando algoritmos recursivos como se describió en la justificación.	Ingeniería.	Ingeniería ingresaría directamente al Sistema la Lista de Materiales y Rutas del Producto, con procesamiento interno del Sistema de Cuadro Combinado para uso por la Programación de Ordenes.

Fuente: Elaboración Propia

En esta etapa se va vislumbrando los actores que van a interactuar con el Sistema, que serian PCP, Mantenimiento e Ingeniera. Los involucrados directamente en el ingreso de la información.

En resumen se requiere que el Sistema procese los datos de Maquinas (ingresados y actualizados por Mantenimiento), de Lista de Materiales y Ruta del Producto (ingresados y actualizados por Ingeniería), Ordenes de fabricación (ingresados y actualizados por

Planeamiento) y genere un listado de Programación de Ordenes de Fabricación por Maquina y Calendarizados de forma automática (Lista Gantt). Todo esto de fácil ingreso y aceptable visualmente.

Tabla 18

Cuadro Entradas y Salidas del Sistema Propuesto

Entradas		Resultado
-Lista de Materiales (con Ensambles) -Ruta de Productos (Maquinas y Operaciones por donde pasa cada producto) -Lista de Maquinas (Disponibles o no) -Lista de Productos -Características de Productos (Formatos, tiempos, etc.) -Lista de Operaciones -Orden de Producción -Horarios de Producción	SISTEMA INFORMÁTICO DE PROGRAMACIÓN DE ORDENES	-Programación de Ordenes en ambiente Job Shop (Gantt de Ordenes por Maquina Calendarizados) -Consumo de Componentes

Fuente: Elaboración Propia

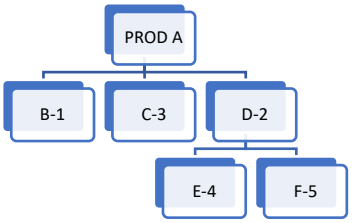
En esta etapa se mostrará la propuesta de Diagrama de Flujo para el Sistema Informático, detallando además los Requerimientos del Sistema, consolidando lo indicado en los párrafos anteriores.

La propuesta de sistemas tiene que considerar que se cuenta con la siguiente información:

Lista de Materiales: Es una lista donde se indica los componentes para un producto determinado, indicando también las cantidades de cada uno. Esto se explicó en el fundamento teórico y a continuación se muestra un ejemplo:

Figura 27

Lista de Materiales

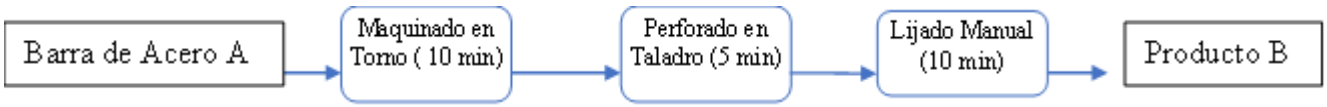


En el cuadro se muestra el que para fabricar el Producto A se necesita 1 unidad de B, 3 unidades de C y 2 unidades de D, y para hacer 1 unidad de D, se necesita 4 unidades de E y 5 unidades de F. A manera de ejemplo, para hacer una unidad de A, necesito 2x5=10 unidades de F.

Ruta de Productos. Indica las maquinas por donde pasa un producto para su transformación y llegar a ser el producto requerido. Ejemplo para lograr un producto B (barra de acero con 3 huecos), se ha tenido que comprar la una barra de acero como materia prima, hacerle un maquinado en el torno, hacerle huecos con el taladro y darle un lijado manual. La ruta del Producto B seria:

Figura 28

Ruta del Producto B



Lista de Maquinas Disponibles. Es una lista donde se indica el momento que las maquinas estén listas para su uso. Ejemplo, el torno N°2 esta disponible desde hoy, pero la fresadora N°2 mañana esta disponible. Ejemplo:

Tabla 19

Ejemplo de Disponibilidad de Maquinas

Maquina	Disponibilidad
torno N°2	Hoy
fresadora N°2	Mañana
Prensa N°1	Hoy

Lista de Ordenes de Fabricación: Es una Lista con las Ordenes de Fabricación con sus Fechas de Entrega y Prioridades. Ejm:

Tabla 20

Lista de Ordenes de Fabricación

Orden	Producto	Cantidad	Fecha de Entrega
1	A	1000	24/09/2022
2	A-1	2000	21/09/2022
3	A-2	1500	18/08/2022

Con estas 4 listas principales el Sistema tiene que calcular una Programación de Ordenes Con el siguiente formato por ejemplo:

Tabla 21

Ejemplo de Gantt

Maquina	OP	Producto	Dia	Inicio	Fin
Torno N”2	1	A	14/09/2022	08:30	14:30
Torno N”2	2	B	14/09/2022	14:30	18:00
fresadora N°2	1	C	16/09/2022	10:00	12:00
fresadora N°2	2	B-1	16/09/2022	15:00	18:00

Para este objetivo se propone lo siguiente: usar repositorios de información de dos tipos, repositorios estables y temporales. Repositorios se llaman estables porque almacenan

información que no cambia a lo largo de la ejecución del sistema o que se generan en la ejecución y se almacenan, por ejemplo Lista de Materiales, Lista de Maquinas, Rutas de los Productos, Ordenes de Producción, Gantt, Asignación de Maquinas. Los llamados temporales se denominan así porque se guarda información de manera temporal durante la ejecución del Sistema, pero una vez terminada la ejecución los datos se borran. Por ejemplo: Maquinas Libres, Trabajos disponibles.

En la parte de fundamento teórico se explicó el algoritmo heurístico de la Programación de Ordenes y también el algoritmo recursivo para el tratamiento de la Lista de Materiales, pero ahora se explicará dando mayores alcances y luego se graficará en un diagrama de flujo:

En el tiempo 0 de la programación, hay maquinas libres dispuestas a recibir trabajos, estos trabajos se extraen de la lista de materiales y ruta del producto, que se va a llamar repositorio “combinado de lista materiales y ruta del producto” y deben ser los primeros trabajos en la línea de precedencia, es decir si no se ejecutan estos, no se puede seguir al siguiente paso en la línea de precedencia hasta llegar al producto final. Estos trabajos se colocan el repositorio “trabajos listos para ingresar a máquina” .Si hay 2 trabajos que pueden entrar al mismo tiempo a una maquina libre, se escoge la que tiene mayor prioridad. Una vez asignado el trabajo, se asigna para todas las demás maquinas, hasta terminarlas todas. Y se va a registrando el Gantt parcial. Luego se hace correr el cronometro virtual hasta el mínimo tiempo final programado, es decir, el mínimo tiempo final donde terminan los trabajos en las maquinas ya programadas. En ese tiempo, se liberan maquinas, se vuelven libres e ingresan al repositorio “maquinas libres”, y se vuelve a revisar en la lista de materiales y ruta de productos si hay componentes que pueden entrar a esas máquinas libres. El componente que se ha terminado en ese tiempo, lanza la alerta para la lista de materiales, que el siguiente proceso de la lista de precedencia ya puede ser ejecutado siempre y cuando no tengo componentes “hermanos” parte de un ensamble, en este caso, cuando se termina un trabajo que es parte de un ensamble, el

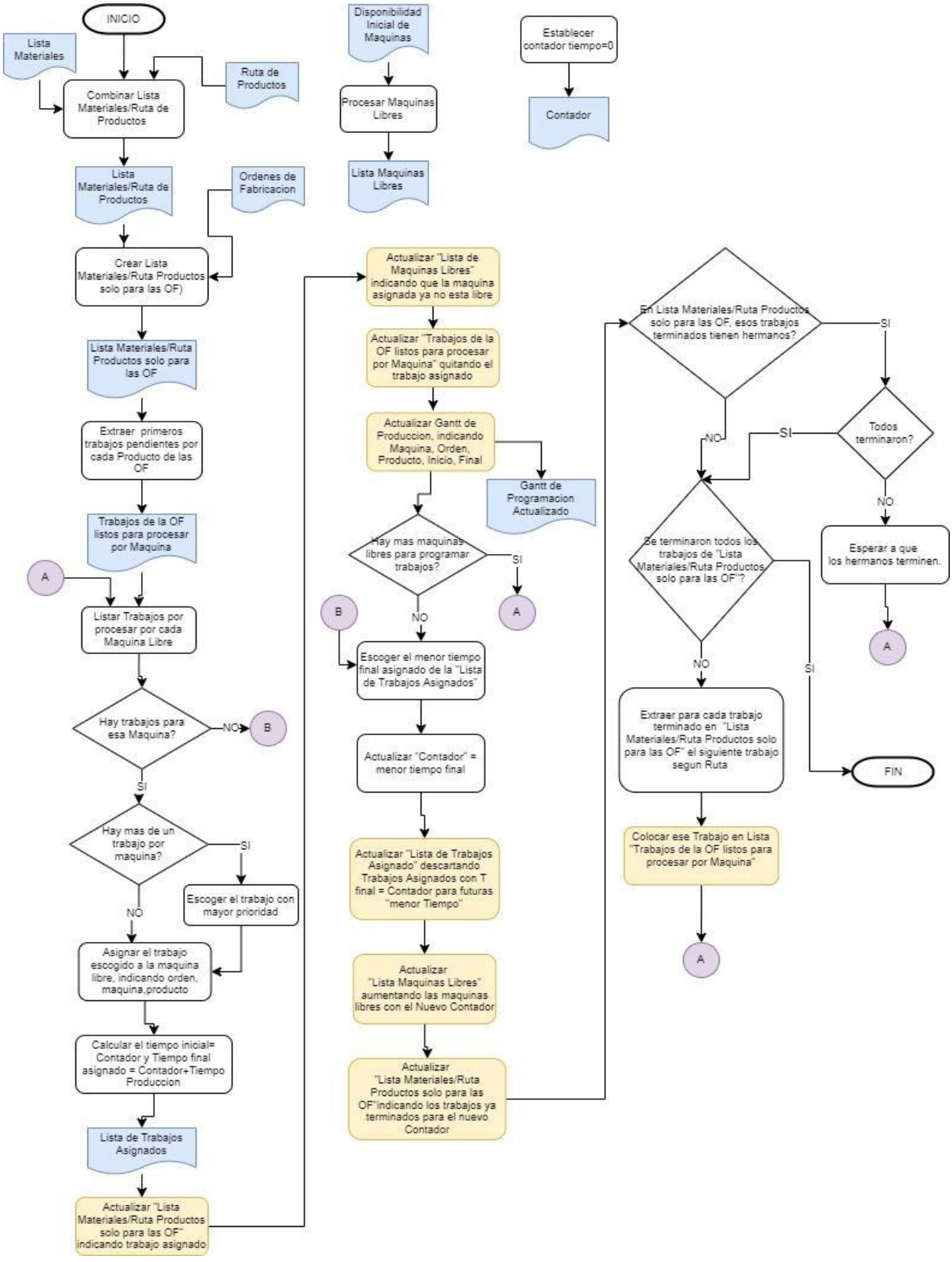
siguiente proceso(ensamble) tiene que esperar a que terminen todos sus “hijos” para que pueda estar disponible como un trabajo listo para ingresar a máquina, si no fuera el caso, esos trabajos ingresan al repositorio “trabajos listos para ingresar a máquina”. Y se repite el ciclo hasta que todos los componentes del repositorio “combinado de lista materiales y ruta del producto” ya ha sido programado. En resumen es un flujo que va desde los repositorios “trabajos listos para ingresar a máquina”, “combinado de lista materiales y ruta del producto”(solo para los productos que se va a programar), “maquinas libres”, a través del tiempo. Este tiempo siempre es el mínimo tiempo de terminación entre varios trabajos programados en diferentes maquinas. Se ejecuta hasta que en el repositorio “trabajos listos para ingresar a máquina” no haya ningún trabajo pendiente y ya se programó todos los trabajos para los productos de las OF que se desea programar.

4.4.6 Diagrama de Flujo Heurístico Propuesto

Se muestra el diagrama de Flujo de la Propuesta:

Figura 29

Diagrama de Flujo Heurístico Propuesto



4.4.7 Objetivos del Nuevo Sistema

En este punto se debe trazar unos objetivos reales a cumplir, que justifiquen la implementación del sistema y toda la inversión que ello genere.

Se puede resumir los objetivos de la siguiente manera:

- **Alcanzar rapidez en la generación del Gantt de Programación de Ordenes**. Meta: Obtener esta información de manera instantánea.
- **Manejar un mismo lenguaje en todo el proceso**, evitando errores de trascripción y duplicidad de documentos. La información de las Maquinas, Lista de Materiales y Productos estaría unida en un solo Sistema. Meta: 0 % errores por mes.

4.4.8 Requerimientos del Sistema de Información

De todo lo expuesto se puede detectar los principales requerimientos del Sistema Informático, exigidos tanto por los usuarios como por el Sistema mismo.

4.4.8.1 Requerimientos Funcionales:

- Ingresar Productos Nuevos de forma rápida e intuitiva
- Ingresar Lista de Materiales de forma rápida e intuitiva
- Ingresar Rutas del Producto de forma rápida e intuitiva
- Ingresar Disponibilidad de Maquina de forma rápida e intuitiva
- Ingresar Maquinas Nuevas de forma rápida e intuitiva.
- Ingresar Ordenes de Producción de forma rápida.
- Calcular y Mostrar Gantt de Ordenes programadas, indicando la máquina, las horas de inicio y final, y los componentes que se trabajaran.
- Calcular el consumo de materiales por cada Producto solicitado y mostrarlo.
- Mostrar la ruta de los productos cada vez que se requiera.

4.4.8.2 Requerimientos No Funcionales. Son aquellos requerimientos no obvios para el usuario, están asociados indirectamente con el trabajo funcional diario. Entre ellos tenemos:

- Facilidad en el diseño de las ventanas para un registro fácil de la información
- La base de datos debe soportar miles de datos en el transcurso del periodo de trabajo.
- Lugar físico del Terminal del sistema (La PC), debe ser un lugar idóneo para el registro de la información.
- Velocidad en la transacción del sistema, debe ser lo suficientemente rápida, para no generar demoras.

4.5 Análisis del Sistema

4.5.1 Diagrama de Casos de Uso

De acuerdo a todo lo descrito ya se puede vislumbrar los actores que van a interactuar con el Sistema y casos de uso que se pueden inferir de lo expuesto.

Tabla 22

Lista de Actores del Sistema

Actores	Casos de Uso que Interviene
Planner	Ingresar Ordenes de Producción, Consulta Gantt de Ordenes, Consulta Consumos, Consulta Ruta de Producción, Ingresar Disponibilidad de Maquinas
Ingeniero	Ingresar Producto, Ingresar Lista de Materiales, Ingresar Ruta del Producto.
Asistente de Mantenimiento	Ingresar Maquina.

Actores:

- **Planner:**

  Es el encargado de la programación de Ordenes en la Planta de Producción y controlador de la producción. Debido a la demora que tiene el Planner para la programación de ordenes es que se ha propuesto el presente Sistema Informático.

- **Ingeniero:**

  Es parte del Área de Ingeniería, y se encarga de alimentar al Sistema proyectado en lo referente a Productos, Lista de Materiales y Rutas de fabricación del Producto. En la manera anterior de trabajar, el ingeniero enviaba formatos de Lista de Materiales y Rutas y el Planner tenia que transformarlo para serle de utilidad. Ahora se propone que el Ingeniero directamente ingrese la información al Sistema Propuesto.

- **Asistente de Mantenimiento:**

  Es en el encargado de Ingresar al Sistema la información de Maquinas, para que el Sistema considere esa información en los Programas de órdenes.

Diagramas de Caso de Uso del Sistema:

Figura 30

Diagrama de Caso de Uso del Sistema

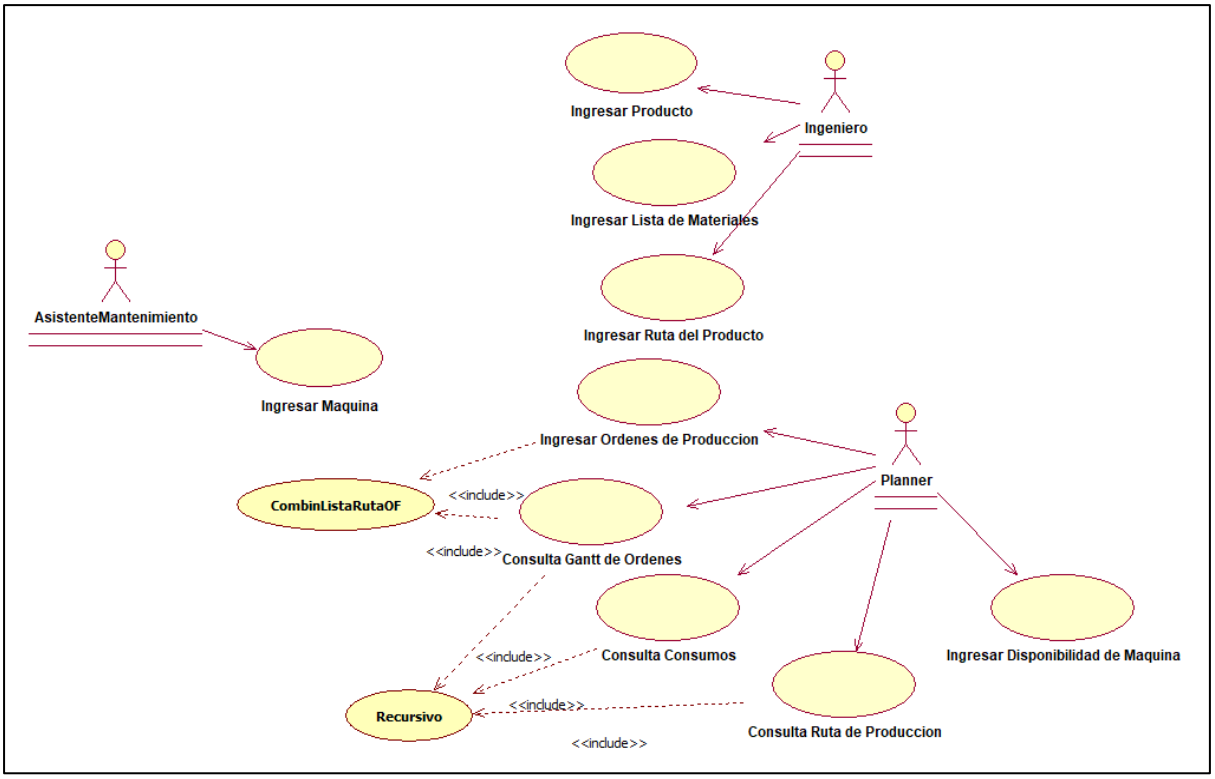


Tabla 23

Caso de Uso Ingresar Producto

Caso de Uso	Ingresar Producto
Descripción del Caso de Uso: El actor Ingeniero ingresa al sistema informático los datos del producto: Nombre del producto (iniciales), Nombre extendido(completo) Tipo de producto (una variable para diferenciar un producto) Clase de producto(la clasificación usual Materia prima, intermedio o producto final) Unidad de medida. El ID del producto lo genera el Sistema Informático de manera correlativa y automática. El sistema informático ingresa el producto usando los comandos usuales para registro en Base de Datos mostrando en una pantalla el ingreso del producto.	

Tabla 24

Caso de Uso Ingresar Lista de Materiales

Caso de Uso	Ingresar Lista de Materiales
Descripción del Caso de Uso: El actor Ingeniero ingresa al sistema informático los datos de la lista de Materiales Componente Hijo: Es el componente que se va a unir a otro para formar un tercero (Padre) Cantidad del Componente Hijo: La cantidad de componente necesario para una unidad del Padre. Componente Padre: Es el componente que en esta transacción es el Padre, es decir el resultado del ensamble de otros componentes. El sistema informático inserta la lista de materiales creada mostrando en una pantalla el ingreso. El Sistema registra la transacción usando los comandos usuales para registro en base de datos, pero en un mismo repositorio tanto los padres como los hijos. Así se podrá usar algoritmos recursivos para calcular los Consumos y el Gantt de Producción.	

Tabla 25

Caso de Uso Ingresar Ruta del Producto

Caso de Uso	Ingresar Ruta del Producto
Descripción del Caso de Uso: El actor Ingeniero ingresa al sistema informático los datos de la Ruta del Producto: Producto: El producto al cual se le va a hacer su hoja de ruta. Operación: La operación que se va a ejecutar en el producto para esa Hoja de Ruta. Maquina: La máquina donde se va a ejecutar la operación. Tiempo Configuración: El tiempo de Preparación del Producto Tiempo Pieza: EL tiempo de producción por una pieza. Numero Trabajadores: El número de trabajadores necesario para esa operación El sistema informático ingresa la Ruta del Producto usando los comandos usuales para registro en Base de Datos mostrando en una pantalla el ingreso de la Ruta.	

Tabla 26

Caso de Uso Ingresar Maquina

Caso de Uso	Ingresar Maquina
Descripción del Caso de Uso: El actor Asistente de Mantenimiento ingresa al sistema informático los datos de las Maquinas. Nombre: El nombre corto asignado a la maquina Nombre Ext: El nombre largo de la Maquina ID: El sistema informático genera un correlativo automáticamente El sistema informático ingresa la Maquina usando los comandos usuales para registro en Base de Datos mostrando en una pantalla el ingreso de la Maquina.	

Tabla 27

Caso de Uso Ingresar Orden de Fabricación

Caso de Uso	Ingresar Orden de Fabricación
Descripción del Caso de Uso: El actor Planner ingresa al sistema informático los datos de las Ordenes. Producto: El producto que están pidiendo fabricar. Lote: La cantidad a fabricar. Fecha: Fecha en la que ingreso la orden al área de Planeamiento Tipo: Si la Orden es interna (generada por la empresa) o externa (del cliente) Fecha de Entrega: La fecha que el cliente esta pidiendo para su entrega. IdOP: Generada automáticamente por el Sistema Informático. El sistema informático ingresa la Orden usando los comandos usuales para registro en Base de Datos mostrando en una pantalla el ingreso de la Orden.	

Tabla 28

Caso de Uso Consulta Gantt de Ordenes

Caso de Uso	Consulta Gantt de Ordenes
Descripción del Caso de Uso: El actor Planner ingresa al sistema informático para visualizar el Gantt de las ordenes de Producción que desee programar. El sistema informático mostrara todos los datos de las ordenes que están pendientes de fabricar para que el actor Planner escoja cuales ordenes desea programar. También el actor podrá escoger cual método de priorización trabajará el sistema informático. La priorización es el método que usara el sistema informático cuando haya 2 o mas trabajos disputando una maquina libre para asignar (como se explicó anteriormente). Se mostrará 4 reglas de prioridades: Fecha de emisión: La orden que llego primero tiene la mejor prioridad Fecha de Entrega: La orden con fecha de entrega mas urgente tiene la prioridad. Suma de tiempo de operación: La orden que tenga la menor suma de tiempos de producción de todas sus operaciones es la que tiene la mejor prioridad. Número de Operaciones faltantes: La orden que tenga el menor número de operaciones es la que tiene la mejor prioridad. En cualquier caso, si hay varias órdenes que coinciden en los parámetros el sistema escogerá la primera ocurrencia. Ejm. 2 órdenes con la misma fecha de entrega (si es que se escoge el método de fecha de entrega para la prioridad) el sistema escoge la primera orden evaluada. Una vez escogido el método de priorización el Sistema usara el algoritmo heurístico explicado anteriormente para mostrar la programación de Ordenes. Esta mostrara los siguientes campos: Máquina, Orden de fabricación, producto programado, producto padre(referencial), Hora inicio, Hora final. Se deberá mostrar todos los productos de la orden de fabricación y todos sus componentes programados en las maquinas correspondientes, con hora de inicio y hora final. Respetando la precedencia, prioridades y ensambles. Es decir si para hacer un producto C, se tiene que hacer primero el producto A, también el producto B y luego ensamblarlos, en el programa el producto C debe estar posterior en el tiempo a cualquiera de los productos A y B. También en general los productos con mejor prioridad se deben	

terminar mas rápido que los otros. Para estos cálculos el Sistema informático usara los algoritmos heurísticos y recursivos mencionados anteriormente.

También el sistema mostrara el % de utilización por cada maquina como indicador de ocupación. Esta se calculará dividiendo las horas ocupadas de la maquina en el periodo de programación entre las horas de programación. Las horas de programación se entiende como el tiempo de terminación mas tarde en la programación menos el tiempo de inicio de la programación para esa máquina en particular.

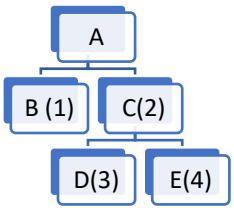
Tabla 29

Caso de Uso Consulta Consumos

Caso de Uso	Consulta Consumos

Descripción del Caso de Uso:

El actor Planner ingresa al sistema informático para visualizar los consumos por cada producto de la base de datos, para ello el actor Planner escoge el producto que desea ver el consumo de materiales. El sistema informático usando algoritmos recursivos mostrara todos los componentes necesarios con sus cantidades para fabricar ese producto. Ejm. Si para el producto A, la lista de materiales indica:

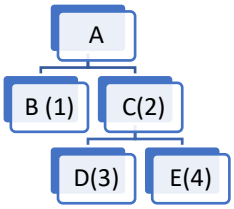


Entonces el Sistema informático deberá mostrar la información:

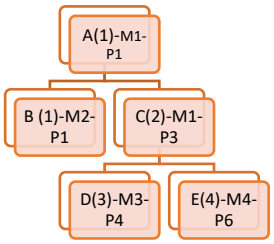
Para 1 unidad de A	
B	1 un
C	2 un
D	6 un
E	8 un

Tabla 30

Caso de Uso Consulta Ruta del Producto

<table>
<tr><td>Caso de Uso</td><td>Consulta Ruta del Producto</td></tr>
<tr><td colspan="2">Descripción del Caso de Uso:
El actor Planner ingresa al sistema informático para visualizar la Ruta por cada producto de la base de datos, para ello el actor Planner escoge el producto que desea ver la Ruta. El sistema informático usando algoritmos recursivos mostrara todos los componentes, máquinas y cantidades necesarias para fabricar ese producto. Ejm. Si para el producto A, la lista de materiales indica:
 

Y la ruta de los productos es la siguiente:
<table>
<tr><td>Producto</td><td>Maquina</td><td>Proceso</td></tr>
<tr><td>A</td><td>M1</td><td>P1</td></tr>
<tr><td>B</td><td>M2</td><td>P1</td></tr>
<tr><td>C</td><td>M1</td><td>P3</td></tr>
<tr><td>D</td><td>M3</td><td>P4</td></tr>
<tr><td>E</td><td>M4</td><td>P6</td></tr>
</table>

Entonces el Sistema informático mostrara la información:

</td></tr>
</table>

4.5.2 Diagramas de Actividades

Como ya tenemos identificados los Casos de Uso del Sistema, procedemos a mostrar el diagrama de actividades de cada uno de ellos.

Figura 31

Diagrama de Actividad Ingresar Maquina

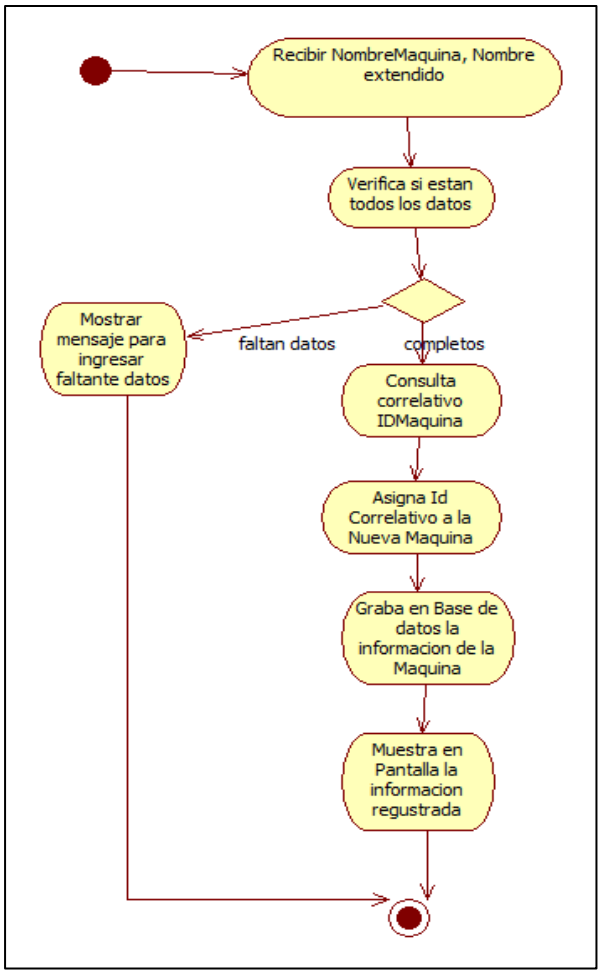


Figura 32

Diagrama de Actividad Ingresar Producto

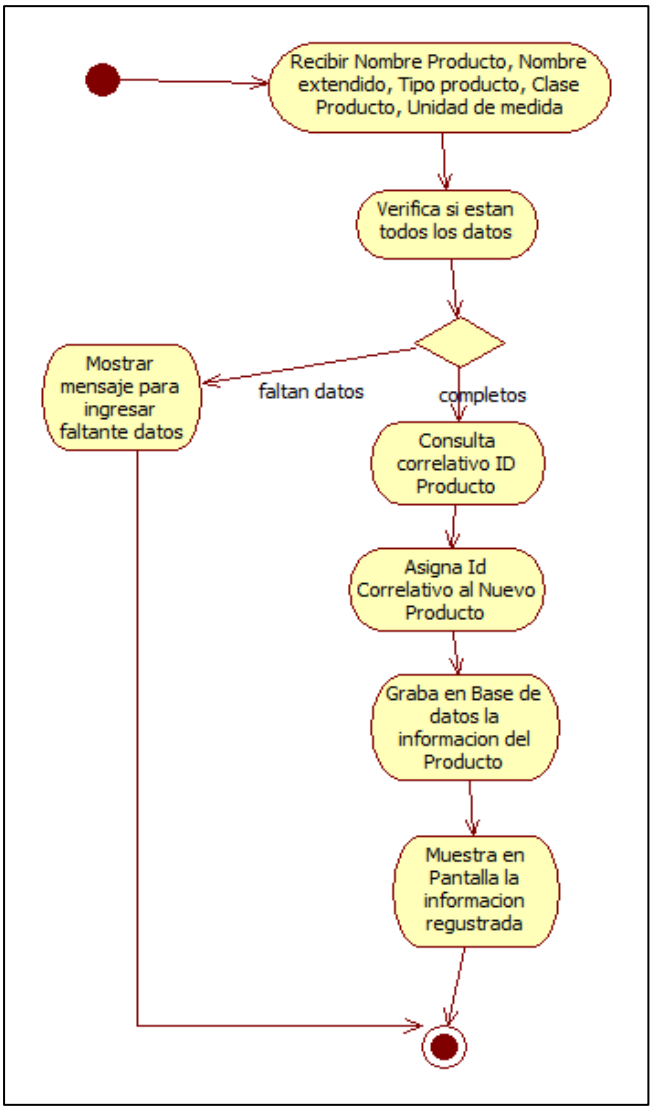


Figura 33

Diagrama de Actividad Ingresar Lista de Materiales

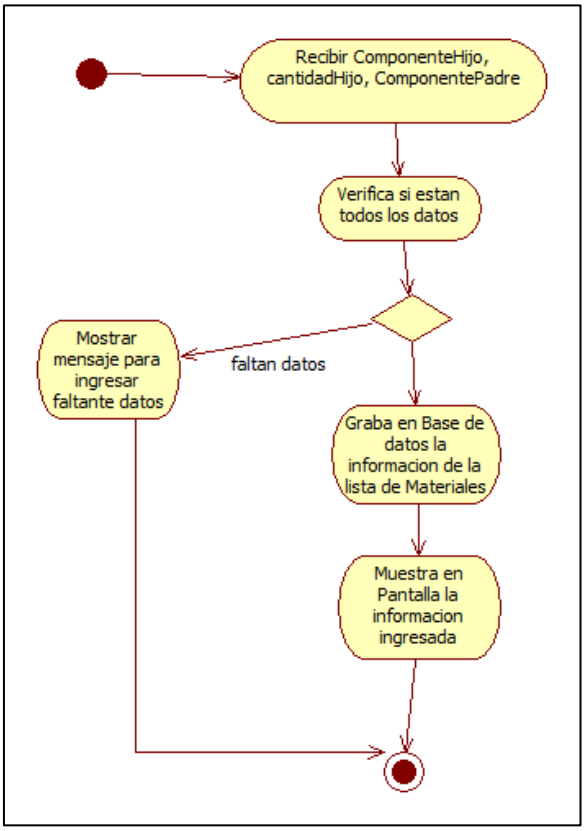


Figura 35

Diagrama de Actividades Ruta del Producto

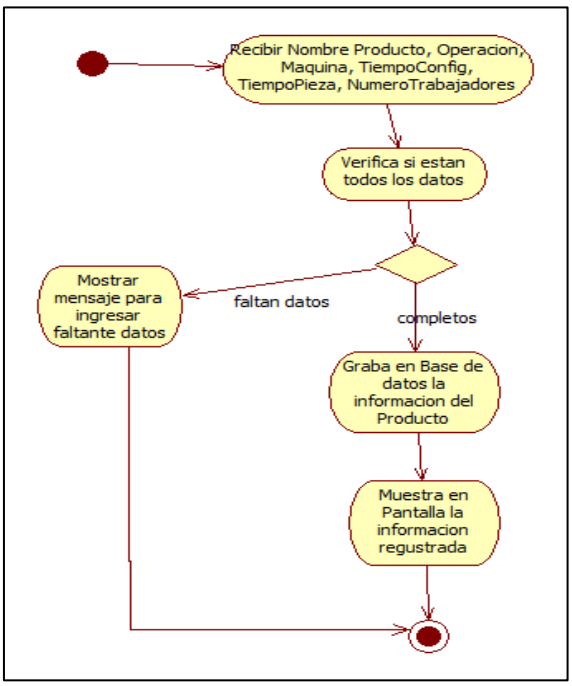


Figura 34

Diagrama de Actividades Ingresar Orden de Fabricación

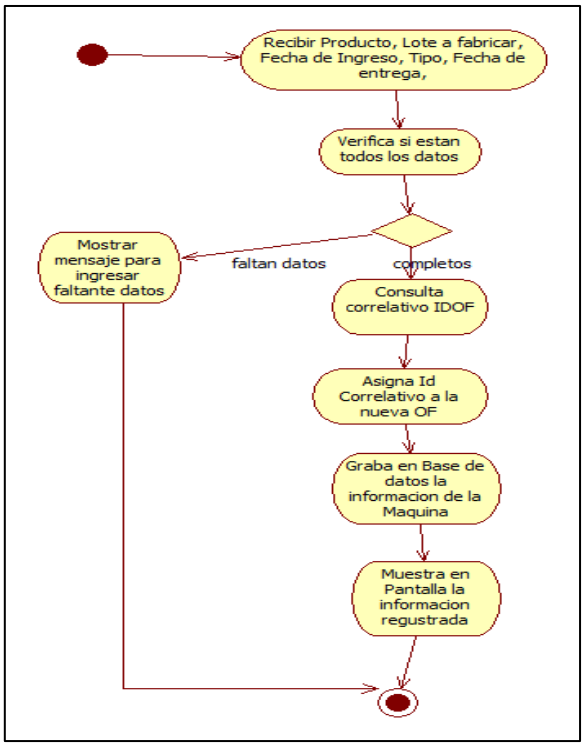


Figura 36

Diagrama de Actividades Consulta Consumo

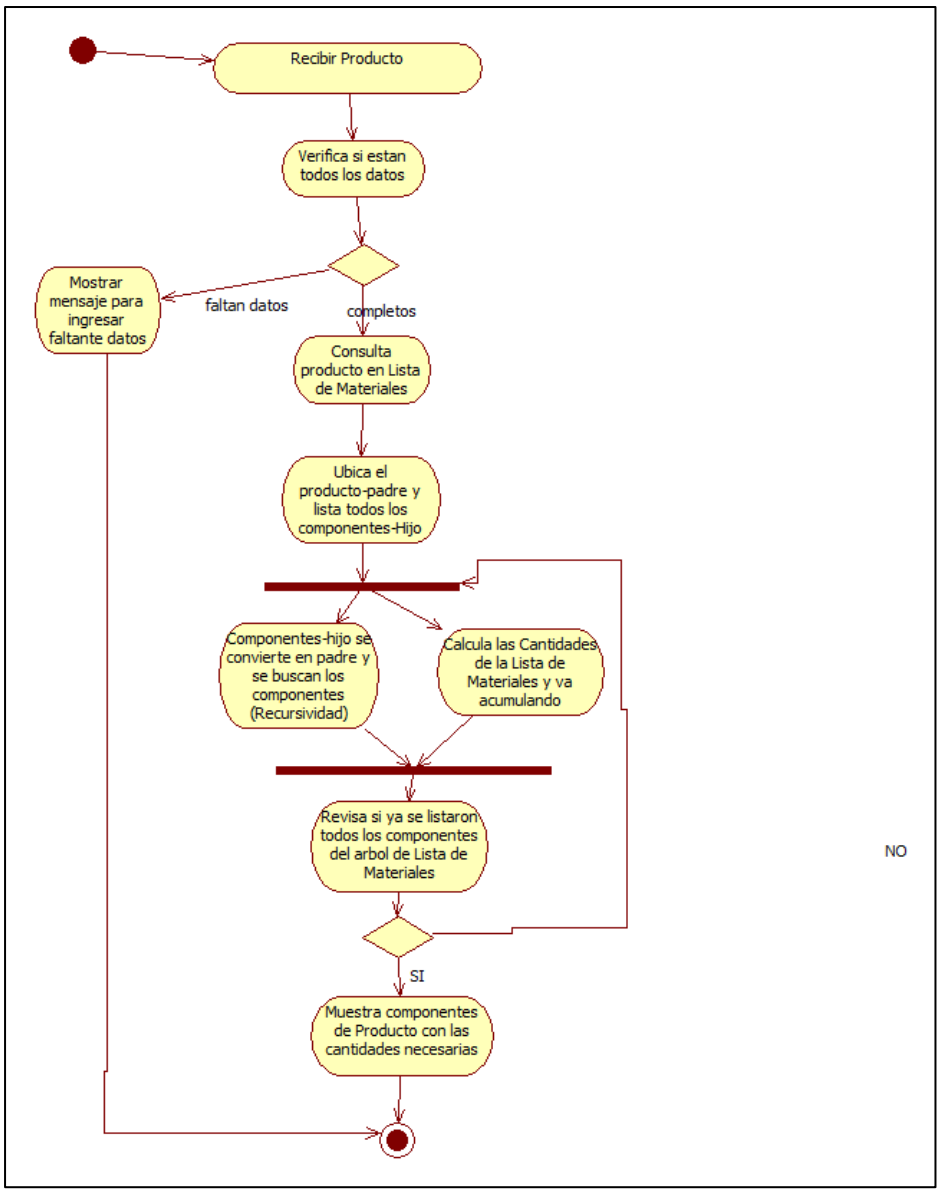


Figura 37

Diagrama de Actividades Consulta Ruta Producto

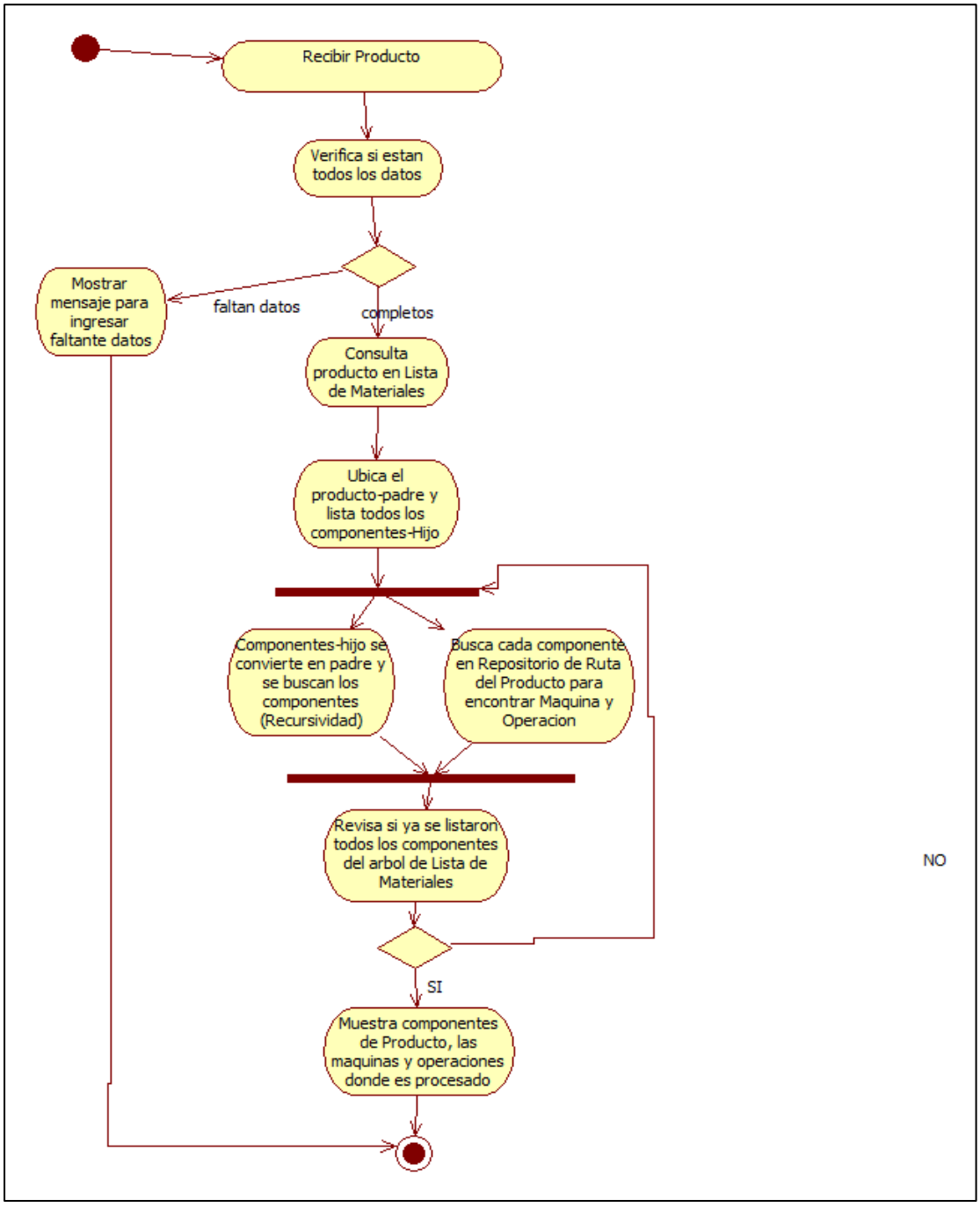
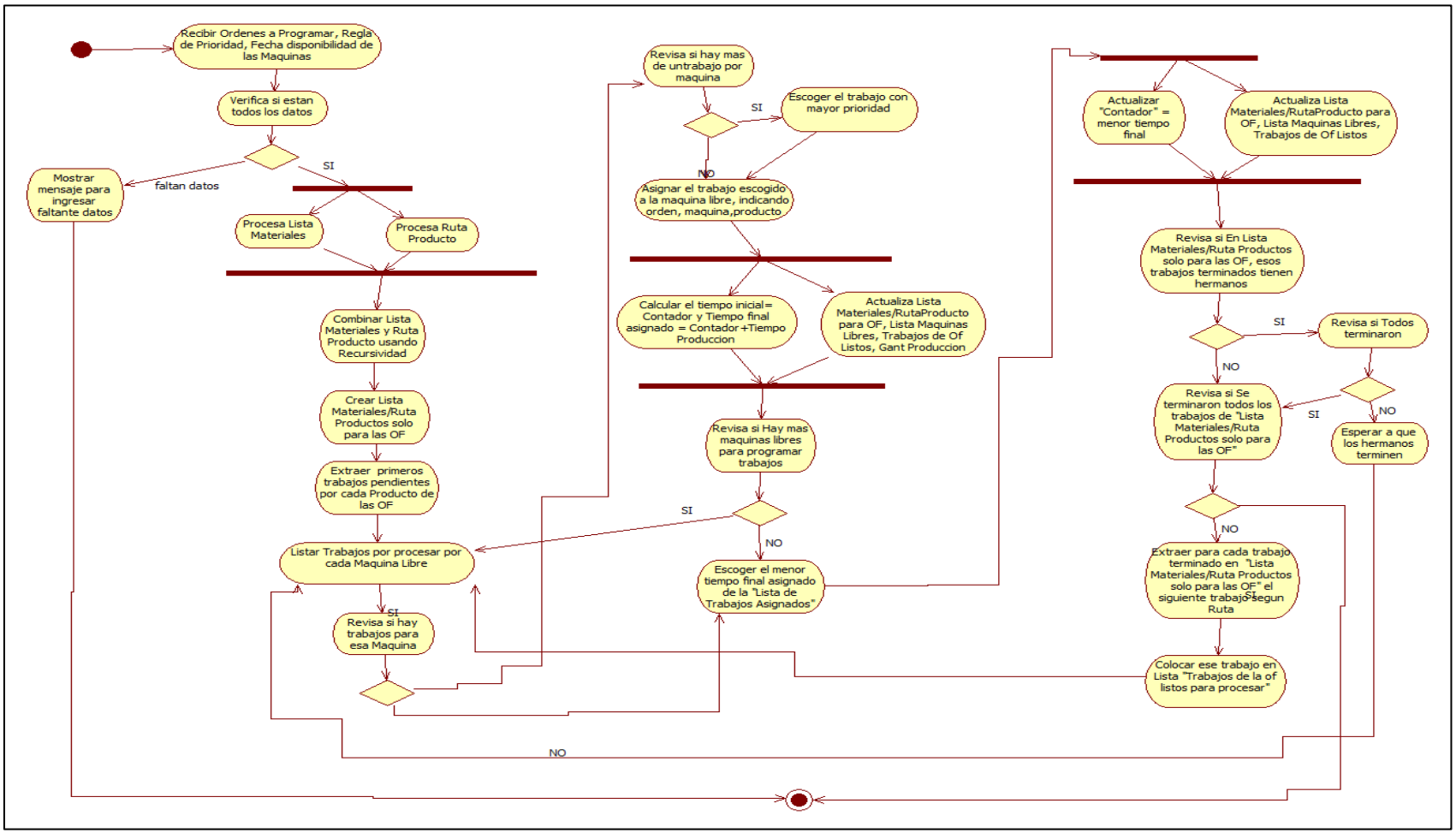


Figura 38

Diagrama de Actividades Consulta Gantt Ordenes



4.5.3 Diagrama de Clases

Con la información recabada de los casos de uso, diagramas de flujo, diagrama de actividades, descripciones de los casos de uso y requisitos se puede determinar las clases del Sistema. Se podría definir las siguientes clases:

Clase Producto: Viene de la lectura de la descripción del Caso de Uso Ingresar Producto

Clase Maquina: Viene de la lectura de la descripción del Caso de Uso Ingresar Maquina

Clase Lista Materiales: Viene de la lectura de la descripción del Caso de Uso Ingresar Lista de Materiales

Clase Ruta del Producto: Viene de la lectura de la descripción del Caso de Uso Ingresar Ruta del Producto

Clase Orden de Producción: Viene de la lectura de la descripción del Caso de Uso Ingresar Orden de Producción

Clase Gantt: Viene de la lectura de la descripción del Caso de Uso Consultar Gantt de Ordenes. Esta clase es donde implementara el algoritmo Heurístico.

Clase Consumo: Viene de la lectura de la descripción del Caso de Uso Consultar Consumos. Esta clase es donde se implementará el algoritmo recursivo para acceder a la lista de materiales.

Todas las mencionadas anteriormente son clases de entidad, ya que la información que implemente la clase se mantendrá permanentemente. También vemos que hay un caso de uso común para los casos de uso Consulta Gantt, Consulta Consumos y Consulta Ruta del Producto,

que es el caso de uso Recursivo. Esta seria una Clase llamada recursivo de tipo Control, ya que la información no se mantiene, si no es controlador que extrae información de una clase para uso de otra pero de manera temporal

Clase Recursivo: Viene de la lectura de la descripción del caso de uso Recursivo. Esta clase es donde se implementará el algoritmo recursivo.

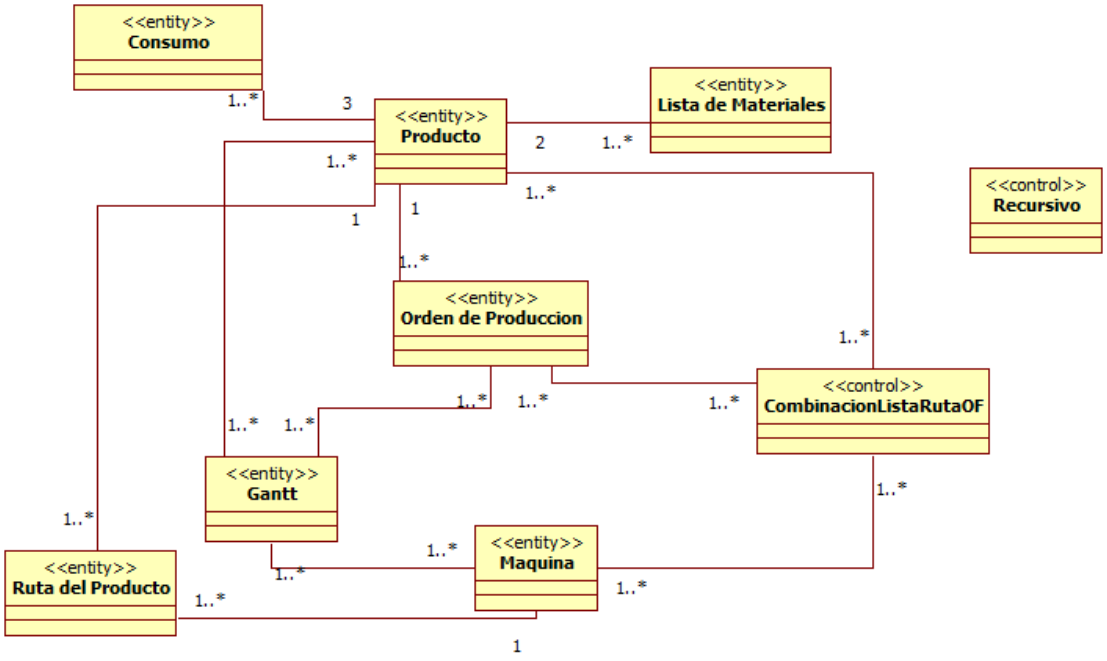
Igualmente de la lectura del Caso de Uso Consultar Gantt de Ordenes, los diagramas de flujo y actividades, se hace necesario una clase control que modele la combinación de Lista de Materiales/Ruta de Productos para las OF programadas, esta seria la Clase CombinacionListaRutaOF.

Clase CombinacionListaRutaOF: Viene de la lectura de la descripción del caso de uso Consultar Gantt de Ordenes. Esta clase usara el algoritmo recursivo.

A continuación de muestra el Diagrama de Clases

Figura 39

Diagrama de Clases del Sistema Propuesto



4.5.4 Diagrama de Estados

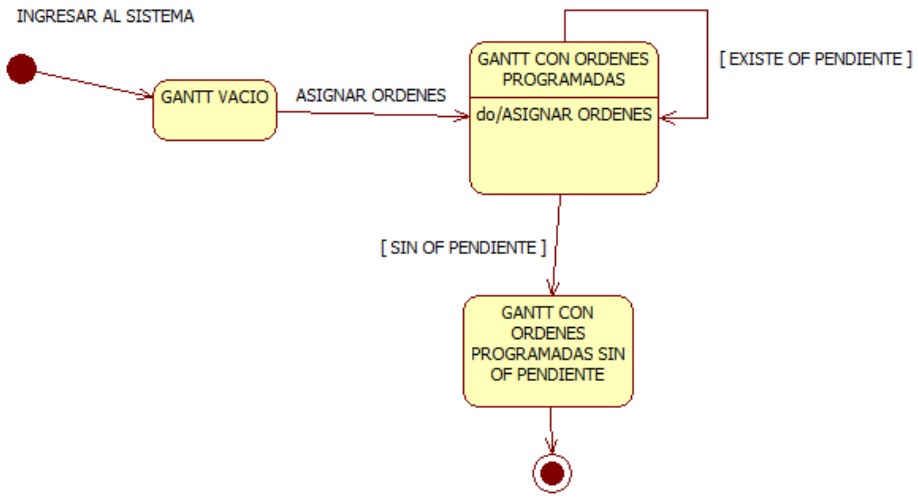
Una vez obtenido el Diagrama de Clases podemos elaborar un Diagrama de Estados, que va indicar los cambios en el estado de una clase en respuesta a una serie de eventos. Para ellos mostraremos el Diagrama de Estados para la clase Gantt ya que es la clase que nos va a mostrar el fin mismo del Sistema informático que es la Programación de las ordenes de Fabricación.

Los estados por los que pasa la Clase Gantt son los siguientes:

1. Gantt Vacío. Evento que lo genera: Ingreso al Sistema
2. Gantt con ordenes programadas, pero con OF pendiente. Evento que lo genera: Asignación de Ordenes Programadas.
3. Gantt con ordenes programadas con ninguna OF pendiente. Evento que lo genera: Comprobación que no hay OF pendientes.

Figura 40

Diagrama de Estados de la Clase Gantt

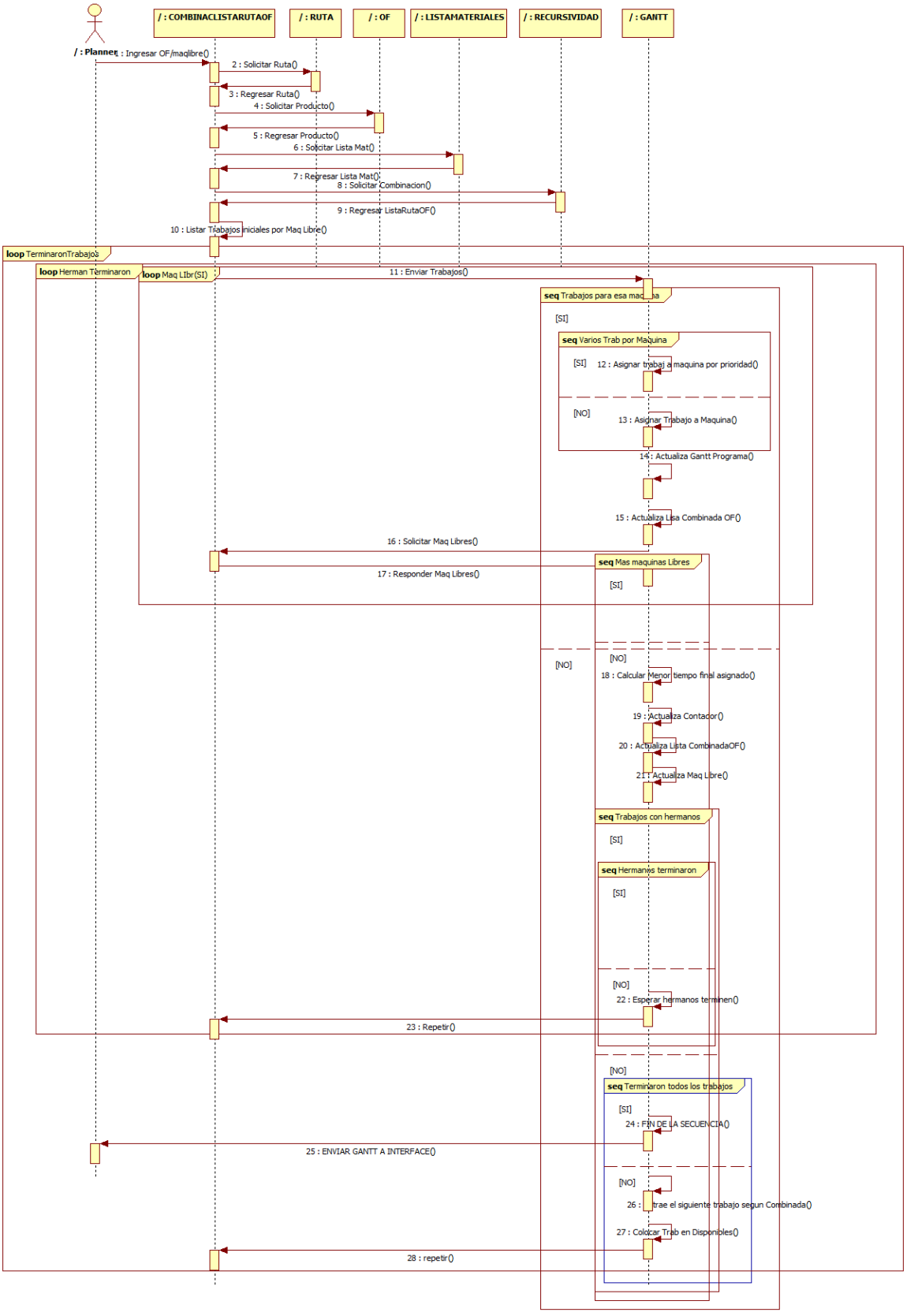


4.5.5 Diagrama de Secuencia.

De la misma manera que para el Diagrama de Estados, una vez obtenido el Diagrama de Clases podemos elaborar un Diagrama de Secuencias, para ilustrar el comportamiento e interrelación de las instancias de Clase del modelo informático. Para ello haremos el diagrama de secuencias para una instancia del Caso de Uso Consulta Gantt Ordenes. Este Diagrama de Secuencia esta fuertemente relacionado con el diagrama de Actividades del Caso de Uso Consulta Gantt Ordenes, ya que el Diagrama de Secuencia grafica una instancia del Caso de Uso.

Figura 41

Diagrama de Secuencia de una instancia del Caso de Uso Consulta Gantt Ordenes.



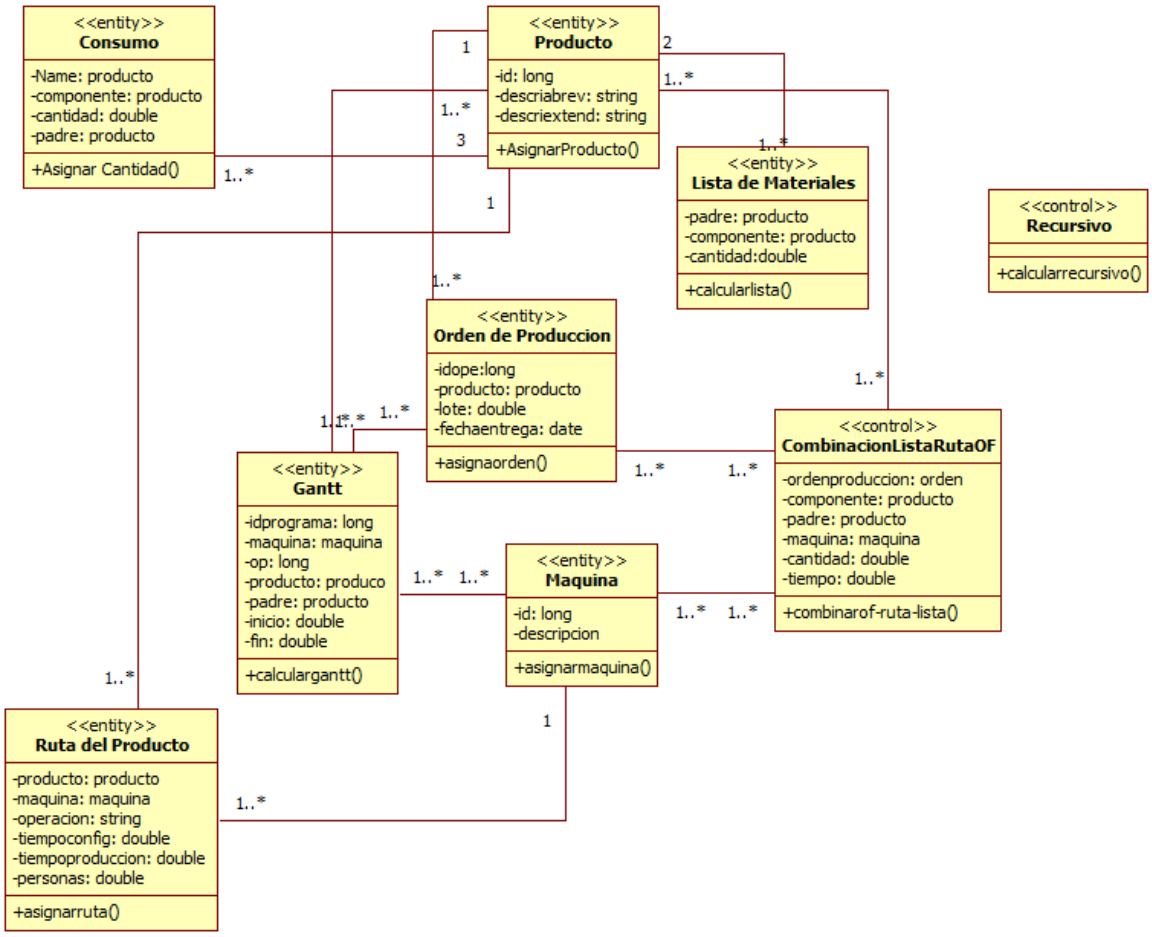
4.6 Diseño del Sistema

En este nivel de acuerdo a la etapa anterior del análisis ya podemos diseñar un diagrama de clases con los atributos necesarios y los métodos del mismo, es decir el Diagrama de Clases Físico.

4.6.1 Diagrama de Clases Físico.

Figura 42

Diagrama de Clases Físico



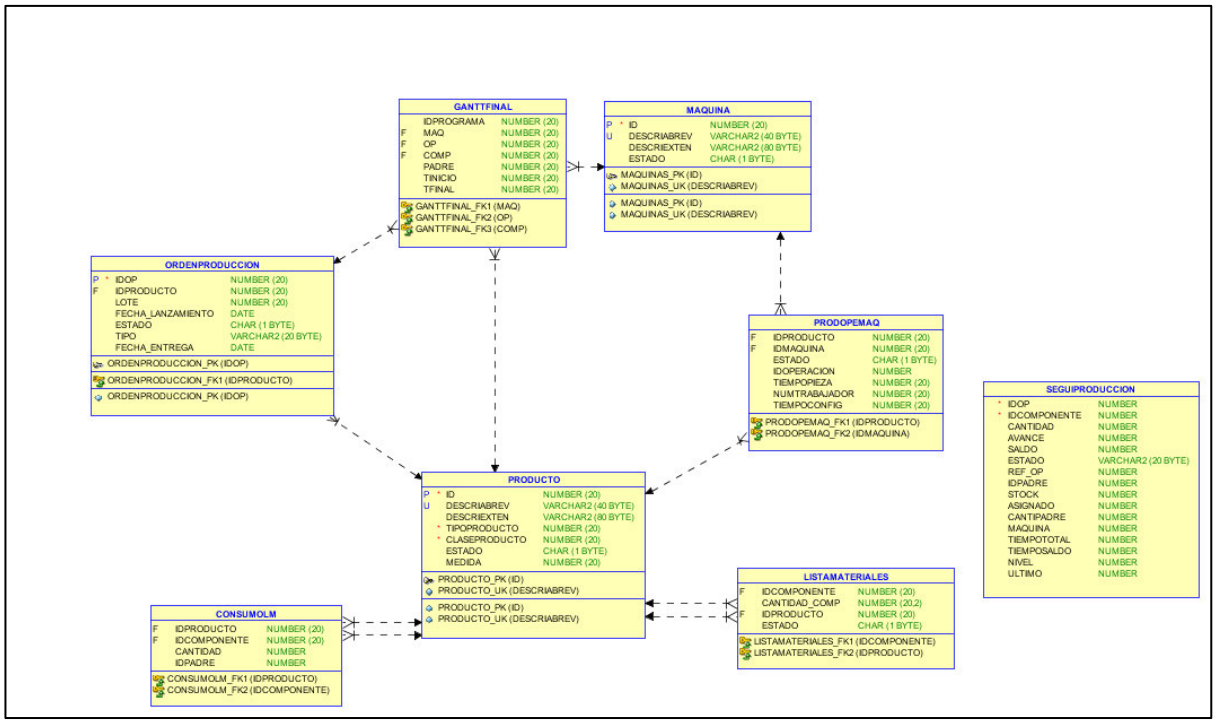
Se muestra clases entidad y clases control. Las clases entidad son las que posteriormente se tendrá que implementar sus clases interfaz para interactuar con el usuario, así como sus clases propias que implementan el modelo.

4.6.2 Diagrama Entidad Relación

Una vez que ya tenemos el diagrama de clases físico diseñado se puede hacer un símil entre una clase entidad con una entidad del Diagrama Entidad Relación, usando los mismos conceptos de las relaciones entre clases, asociaciones, atributos y cardinalidad. Se usa Diagrama Entidad Relación porque el Gestor de base de datos Oracle usa ese concepto para poder efectuar el diseño de las tablas que van a almacenar la información del Sistema Informático de Ordenes. Se muestra el diagrama Entidad Relación de un ambiente Oracle a través del Visor Sql Developer.

Figura 43

Diagrama Entidad Relación



4.6.3 Patrón de Diseño

En el patrón de diseño se usará el modelo vista controlador (MVC), donde se separa claramente para facilitar el orden en el acceso a datos entre el modelo, vista y el controlador.

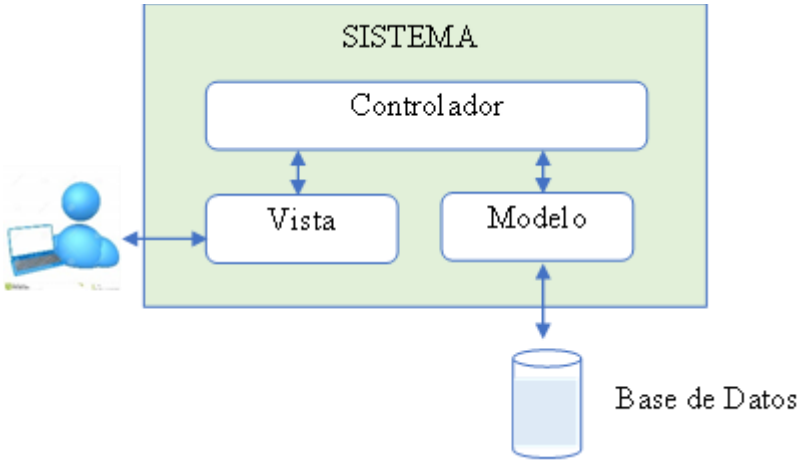
La vista es la interfaz de usuario, por donde el sistema va a interactuar con el mismo, esta vista para acceder a los datos y a la lógica del sistema (modelo) hace uso de los controladores, que son los entes intermediarios entre la interfaz visual y el modelo o datos.

Para ellos se crea clases de vista y una clase de interface para implementar los controladores, aparte de lase clases de entidad y control ya descritas.

El flujo que se sigue es el siguiente:

Figura 44

Patrón de Diseño



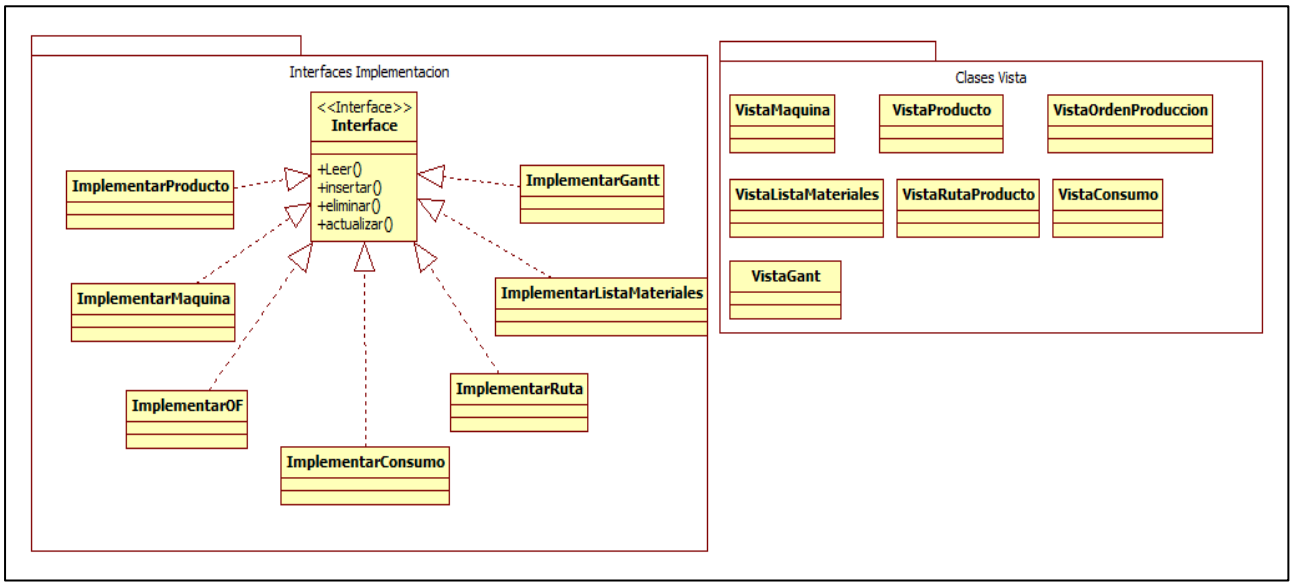
Para ello se implementará las clases necesarias para usar el patrón MVC, clases de Vista (Interfaz de Usuario) y clases Controladoras. Estas últimas Clases como ejecutan métodos comunes a todos ellos, lo implementamos a partir de una clase Interface. Esta clase interface implementada es la que interactúa con la Base de Datos en base a los requerimientos de las clases Vista procesando la información en base al modelo del Sistema. Habrá una clase Vista por cada Clase Entidad que se modelo, que es la que interactúa con el Usuario

4.6.4 Diagrama de Clases Interface-Vistas

Se muestra un diagrama de Clases con la interface implementada y las Clases Vista necesarias para la interacción.

Figura 45

Diagrama de Clases con Interfaces y Clase Vista



4.7 Implementación del Sistema. Base de Datos, Interface Grafica y Codificación.

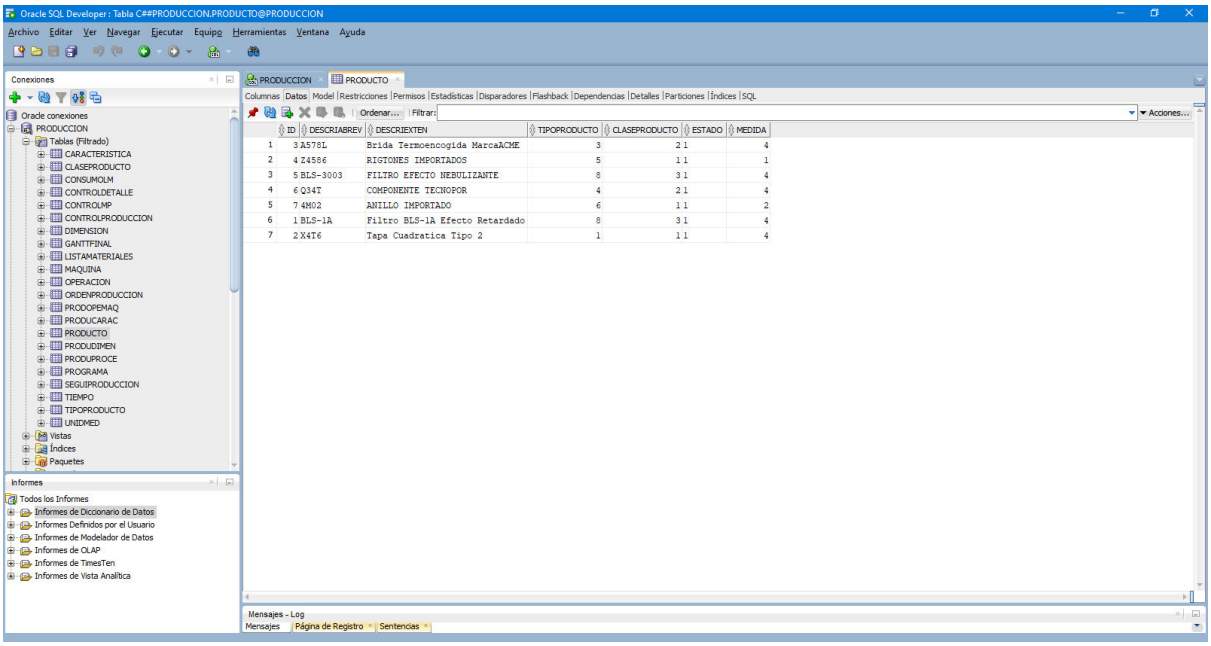
4.7.1 Entornos de Desarrollo

Base de Datos

Para la implementación de la Base de Datos se ha usado la base de Datos Oracle en su versión express.: Oracle Database 21c Express Edition Release 21.0.0.0.0 – Production Version 21.3.0.0.0 y esta se gestionado mediante el Entorno de Desarrollo integrado Oracle SQL Developer Version 21.4.2.018. Se muestra el entorno de desarrollo:

Figura 46

Base de Datos del Sistema en Oracle Express



Lenguaje de Programación

Se ha usado el lenguaje de programación JAVA versión 15.0.2 y como plataforma de Desarrollo Apache NetBeans IDE versión 12.6 .

Figura 47

NetBeans

4.7.2 Costo de la Implementación

En la implementación del Sistema se considera los siguientes conceptos:

- Costo de las licencias de uso de la base de datos
- Costo de la licencia del lenguaje de programación.
- Costo de Codificación del Sistema
- Hardware Necesario.
- Capacitación y Soporte a los usuarios.

A continuación, se detalla cada ítem descrito:

- **Costo de las licencias de uso de la base de datos:**

La base de datos Oracle Express es de descarga gratuita, desde la web de la empresa: https://www.oracle.com/pe/database/technologies/appdev/xe.html esta base de datos Oracle gratuita, puede almacenar hasta 2 GB de datos del usuario. Por lo que el costo de la Base de datos es S/. 0.

- **Costo de la licencia del lenguaje de programación**

El lenguaje de Programación Java es un lenguaje de Software Libre bajo licencia **GNU General Public License**. Es decir Software de Uso libre para estudiar y compartir Java. Se ha usado el entorno de desarrollo NetBeans, también de acceso libre en la página https://netbeans.apache.org/download/index.html. Por lo que el Costo de licencia del lenguaje Java es S/0.

- **Costo de Codificación del Sistema.**

Se considera un tiempo de programación de 3 meses, trabajando 8 horas al día y jornada semanal de 48 horas. Pero considerando que es un Sistema ya elaborado se considera un Costo de S/.0.

- **Costo del Hardware Necesario.**

Se considera que los 3 usuarios del Sistema, Planner, Ingeniería y Mantenimiento poseen sus Computadores Propias, por lo que no habría costo de Hardware. Costo = S/.0.

- **Costo de Capacitación y Soporte a los usuarios.**

En este concepto se considera un costo por capacitar en el uso del Sistema, y también posibles personalizaciones posteriores que la MYPE crea conveniente. En este caso se acuerda una cantidad de acuerdo a los cambios requeridos. Costo de Capacitación y Soporte = S/.300

- **Resumen del Costo de Implementación:**

Concepto	Costo
Costo de las licencias de uso de la base de datos	S/.0
Costo de la licencia del lenguaje de programación	S/.0
Costo de Codificación del Sistema	S/.0
Costo del Hardware Necesario	S/.0
Costo de Capacitación y Soporte a los usuarios	S/.300
Costo Total	**S/.300**

Este costo de Implementación del Sistema de Programación es competitivo frente al costo de implementación de Sistemas como SAP Business One. Para este caso tenemos los siguientes costos:

Licencias SAP por usuario profesional = $ 2,700

Instalación del Sistema SAP Business One = $ 50,000.

Costo total del SAP = $2,700 x 3 (usuarios) +$ 50,000 = $58,100

Como se aprecia el costo de implementación del Sistema de Programación resulta mas accesible al presupuesto usual de una PYME en el Perú, ya que según el ministerio de la Producción, las Pymes tienen ventas mensuales entre S/. 61,875 y S/. 948,000.

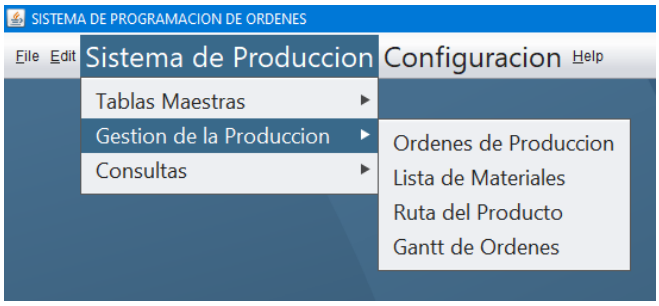
4.7.3 Interfaces Graficas

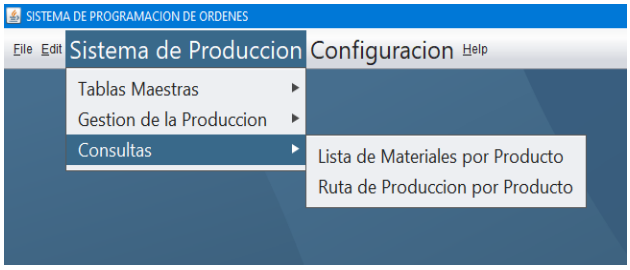
Las interfaces graficas se elaboraron siguiendo los conceptos vertidos en el análisis y diseño del Sistema. Se mostrará la secuencia necesaria para lograr el objetivo de la Programación de Ordenes de Fabricación.

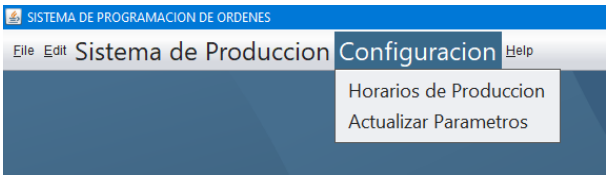
Ventana Inicial. Se muestra la interface de entrada al Sistema, con un menú para acceso a las opciones.

Figura 48

Ventana Inicial del Sistema



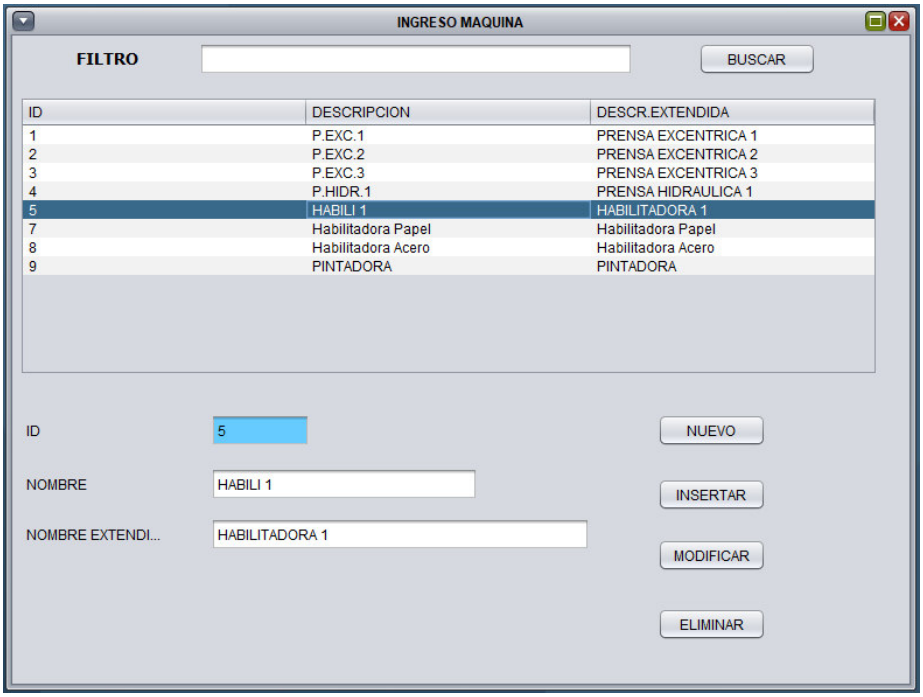




Ingreso de Maquinas. Aquí se ingresa el nombre corto y el nombre extendido de la máquina, el ID se genera automáticamente. Se puede Insertar(nueva máquina), Modificar(Cambiar datos a una maquina), Eliminar (Eliminar una maquina), Nuevo (Para poner en vacío a las celdas).

Figura 49

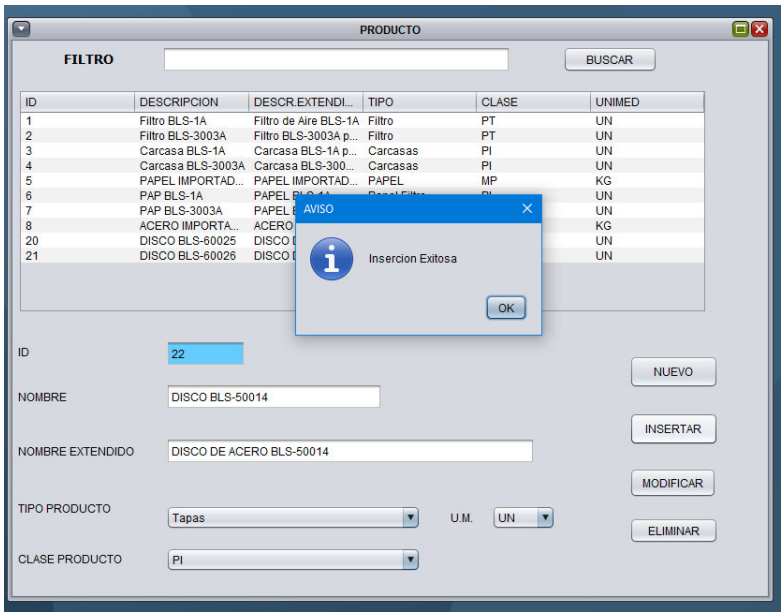
Ventana Ingreso de Maquinas



Ingreso de Producto. Se ingresa los datos de Nombre del Producto, Nombre Extendido (Nombre largo), Tipo de Producto, Clase y Unidad de Medida. El id se genera automáticamente.

Figura 50

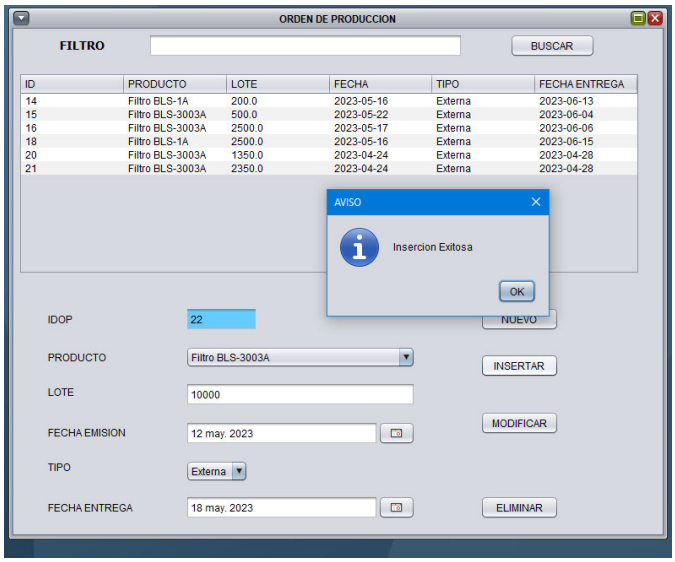
Ventana Ingreso del Producto



Ingreso de la Orden de Producción. Se ingresa los datos de la orden, el producto, lote (cantidad solicitada), Fecha de emisión (fecha de ingreso a Planeamiento de la Orden), Tipo(si es orden del cliente o interna) y Fecha de Entrega solicitada por Ventas o el Cliente.

Figura 51

Ventana Ingreso de Orden de Producción



Ingreso de Lista de Materiales. Se ingresa los componentes para un producto determinado, es decir, si para hacer 1 unidad de A, necesito 2 unidades de B y 3 unidades de C, y además, para hacer 1 unidad de C, necesito 2 unidades de D y 3 unidades de E. Se ingresa 4 veces y en cada transacción se aprieta el botón Insertar. De esta manera:

Tabla 31

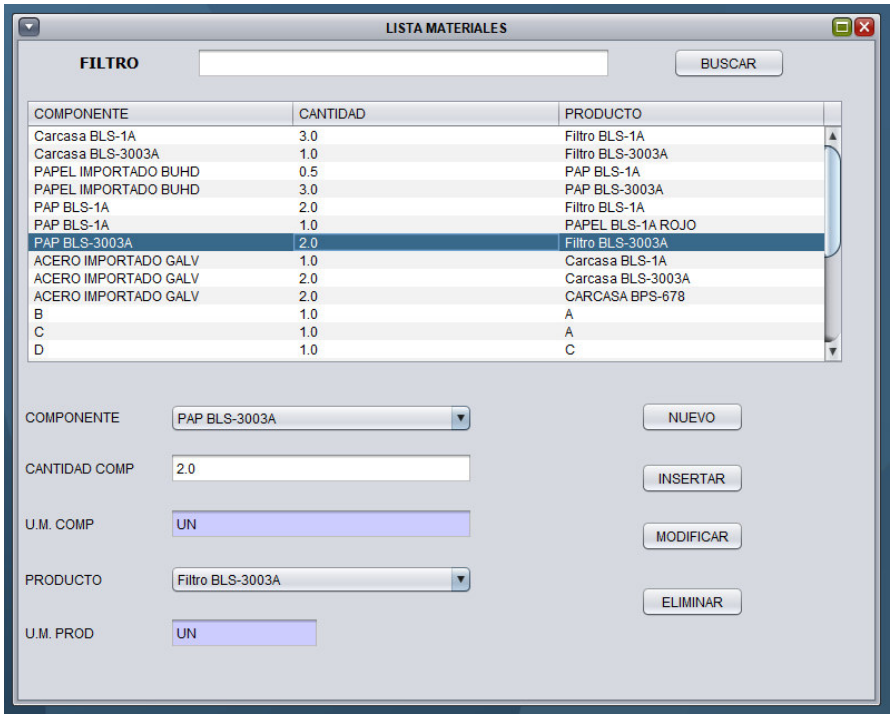
Estructura Lista de Materiales a Ingresar al Sistema

Transacción	Componente	Cantidad Comp	Producto(Padre)
Primera	B	2	A
Segunda	C	3	A
Tercera	D	2	C
Cuarta	E	3	C

La ventaja de este método de ingreso es que es intuitivo, y además se puede ingresar después cualquier componente adicional que resulte.

Figura 52

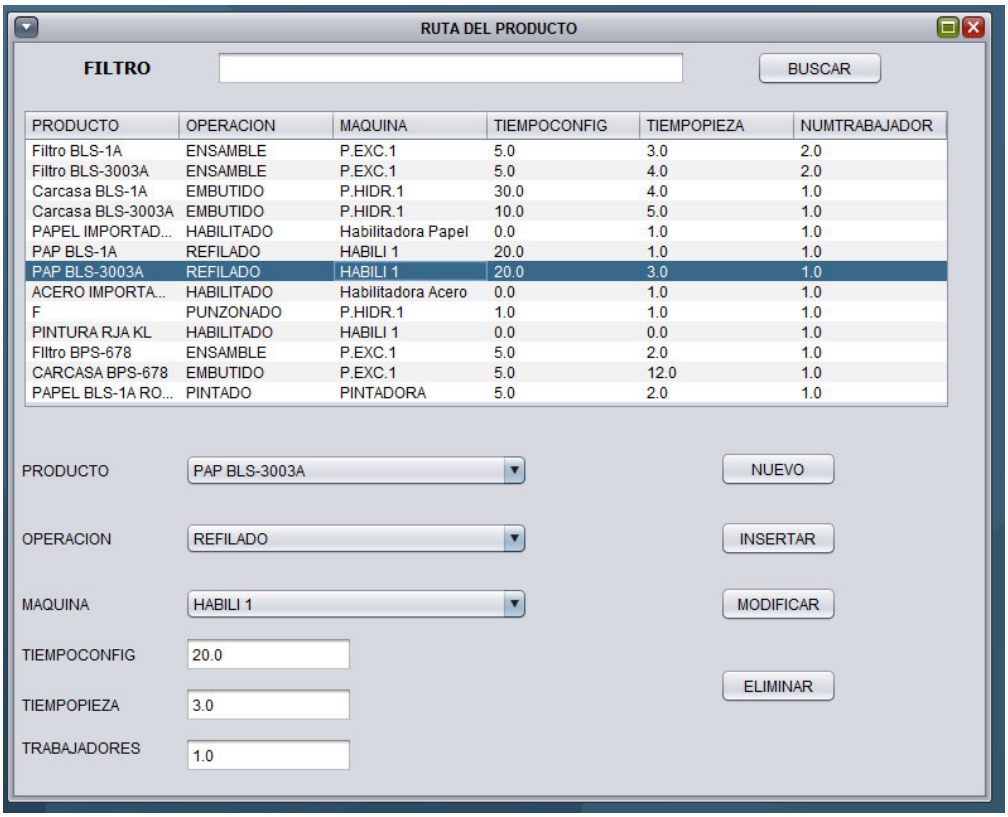
Ventana Ingreso Lista de Materiales



Ingreso de Ruta de Producción. En esta ventana se ingresa la ruta del producto, es decir, la operación y la maquina necesaria para fabricar ese producto. Siempre desde el punto de vista del producto que se fabrica, es decir, si es un producto que es ensamble de otros 2 productos, la ensamble ya esta ingresada en la lista de materiales. Y en esta ventana solo se registra la operación y la maquina necesaria para ensamblar el producto resultante. La ruta de producción junto a la lista de materiales describe la "Hoja de Vida" de un producto. Se ingresa el producto, la operación, la máquina, el tiempo de producción, el tiempo de configuración y cuantos trabajadores se necesita.

Figura 53

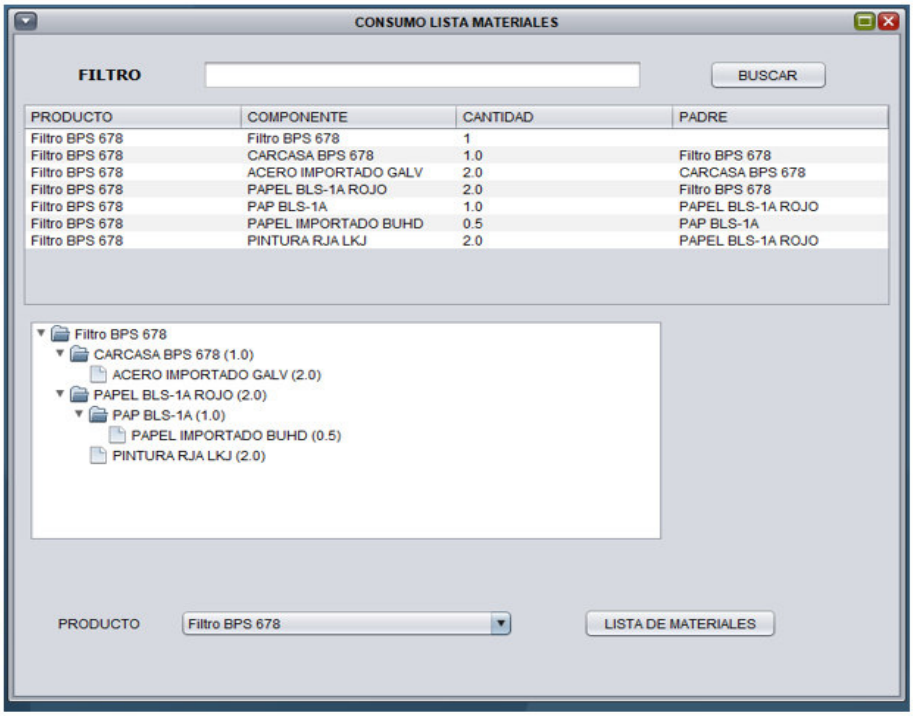
Ventana Ingreso Ruta del Producto



Consulta de Materiales. Aquí se consulta cuales son los componentes de un determinado producto, es decir el árbol de componentes, desde el Padre (el producto buscado) hasta los hijos, y a la vez, estos hijos se convierten en padres y se muestra los hijos de ellos también, con las cantidades necesarias. Para el ejemplo de la lista de materiales, al escoger el producto A para consultar los materiales, arrojaría la siguiente información: 2 unidades de B, 3 unidades de C, 6 unidades de D y 9 unidades de E

Figura 54

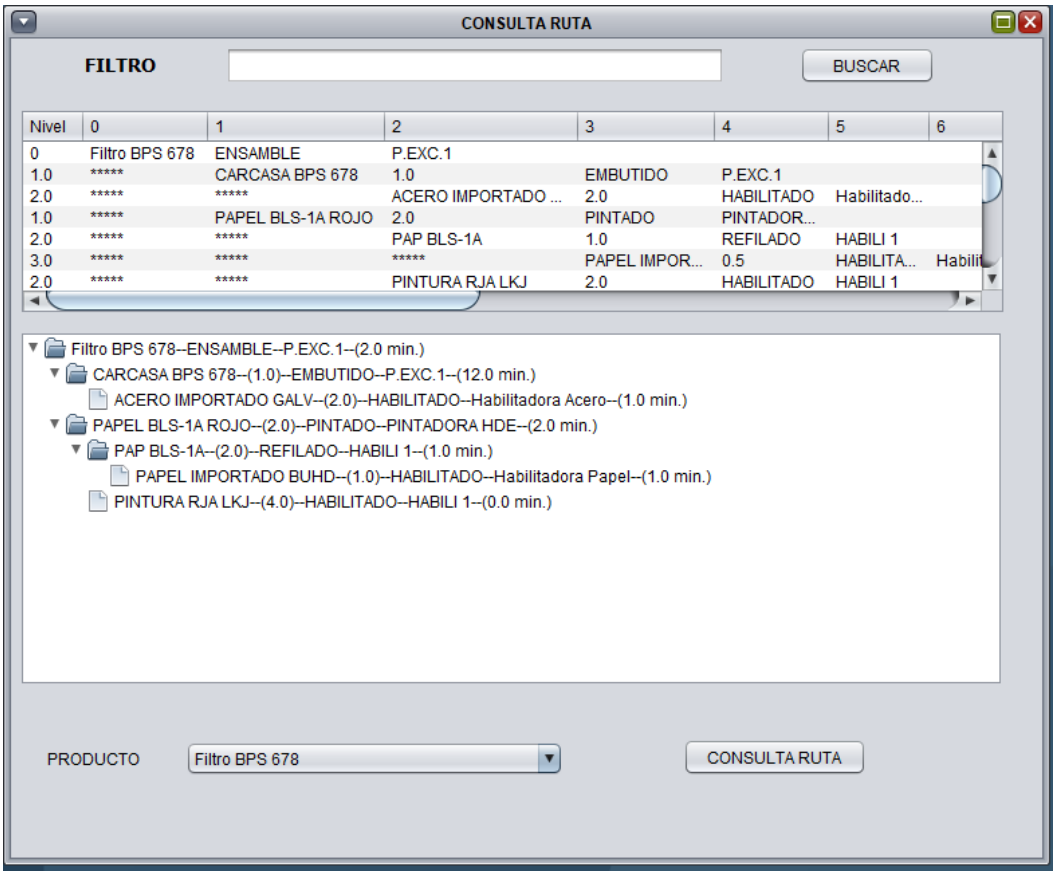
Ventana Consulta Consumo de Materiales



Consulta de Ruta de Productos. Aquí se muestra el árbol que se mencionó en la consulta anterior de Consumo, pero unida a la Ruta de Producción, es decir, muestra la maquina y la operación por donde pasan todos los componentes del árbol de materiales para el producto indicado.

Figura 55

Ventana Consulta Ruta del Producto

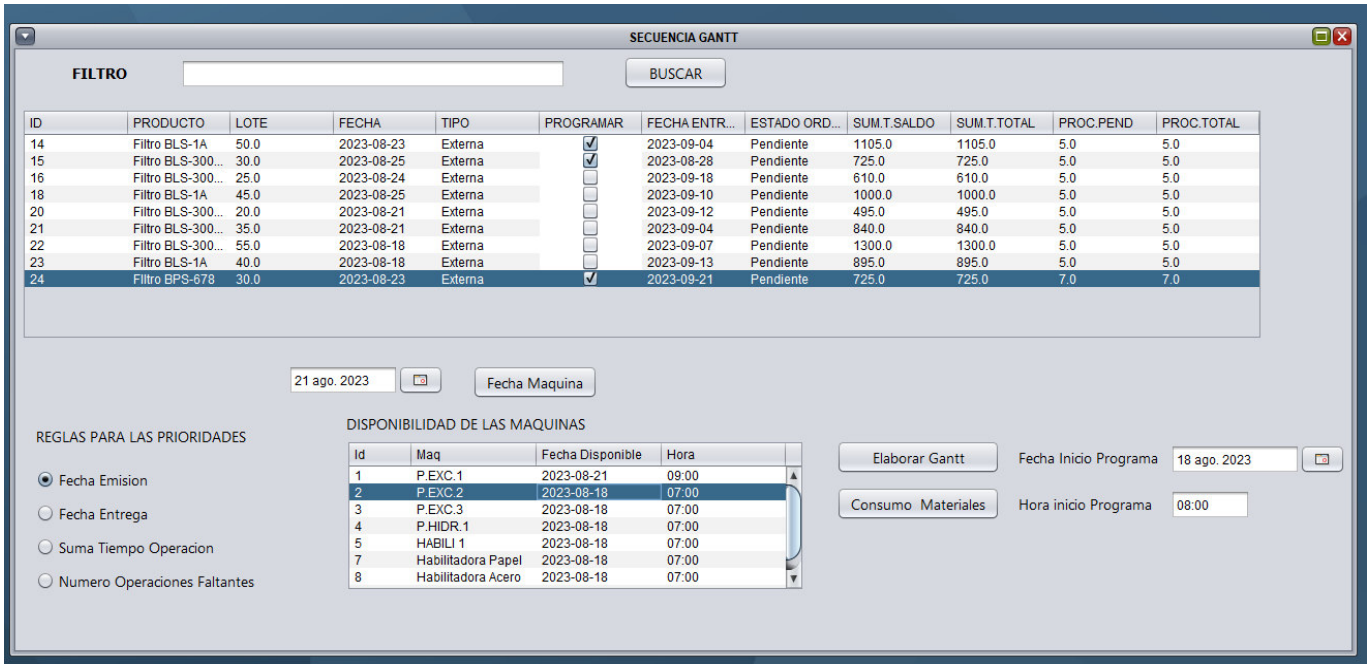


Consulta de Gantt de Producción Ventana Inicial. Esta es la ventana, que al procesarla, va a resultar en la Programación de Ordenes objetivo del Sistema Informático propuesto. En esta Ventana se va a ingresar la siguiente información:

- Ordenes de Producción a programar. En el sistema pueden estar ingresados muchas ordenes, pero el Planner va a escoger las que el crea conveniente para programarlas en planta, dentro de las pendientes de producir. En la ventana el sistema desplegara todas las ordenes pendientes de producir (que han sido previamente ingresadas), mediante unas casillas de check, el Planner escogerá cuales desea que ingresen a la programación de planta.
- Reglas de Prioridad. En la ventana el sistema mostrara 4 opciones de reglas de prioridad. El planner escogerá la mas conveniente de acuerdo a su criterio. El sistema usara esa prioridad escogida para tomar decisiones en la ejecución del programa al momento que 2 o mas trabajos compitan por una maquina libre.
- Tiempo Disponible de las Maquinas. Aquí el Planner ingresara el tiempo en el cual la maquina o maquinas estarán disponibles. Si todas están en cero, significa que todas las maquinas están disponibles en el momento de la programación. Si hay alguna que no va a estar disponible hasta dentro de un tiempo, entonces el Planner pondrá en el campo Disponible, el tiempo en que la maquina estará lista para recibir trabajos. Y el Sistema usara estos datos para hacer la programación.
- Luego de ingresar la información, el botón Elaborar Gantt mostrara el programa.

Figura 56

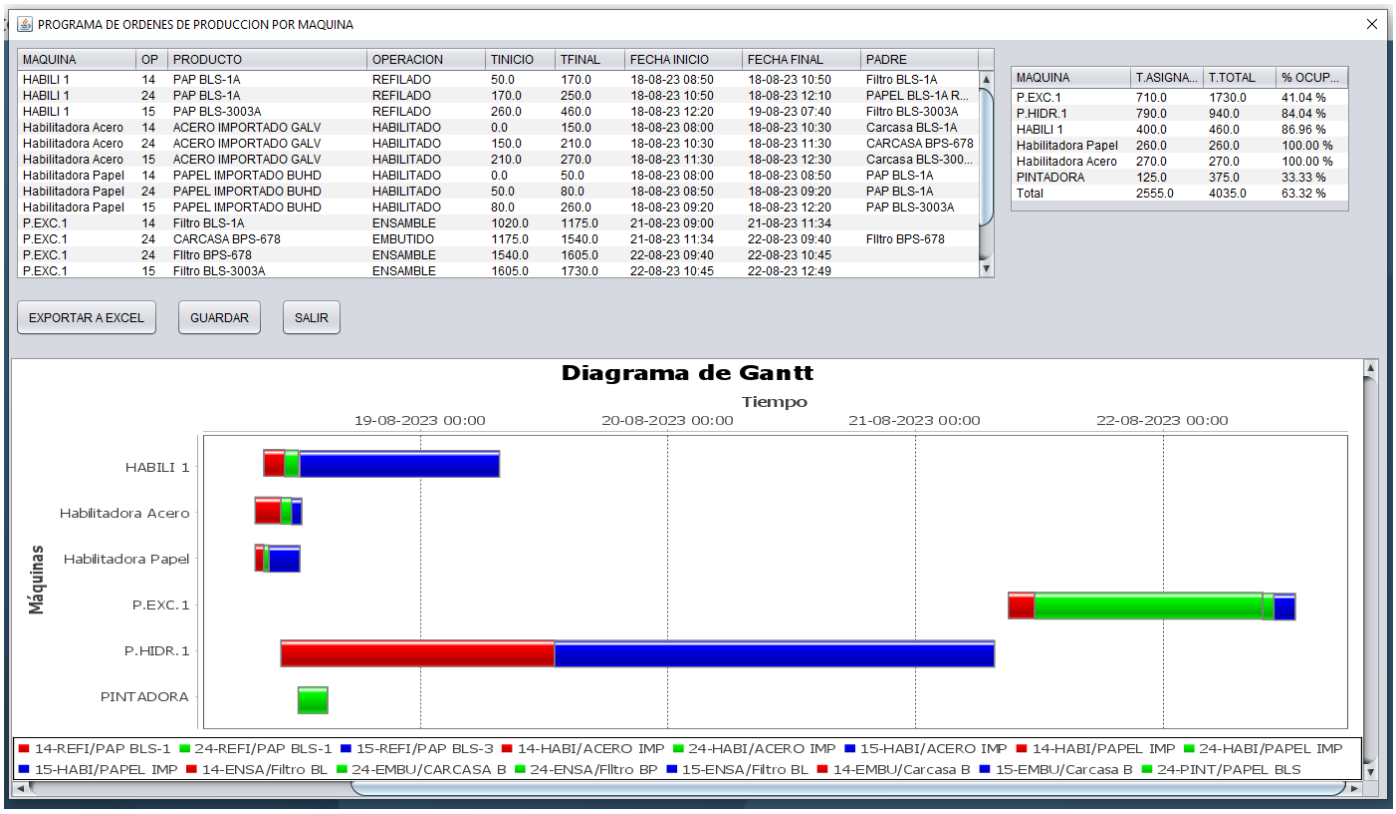
Ventana Inicial de Configuración para el Diagrama Gantt



Consulta de Gantt de Producción Resultado. Este es la ventana que mostrara el Programa de Ordenes de Fabricación, donde se indica, la Maquina, la OP, el producto (Componente) que se va a trabajar, el Padre (como referencia), el tiempo de inicio de la producción y el tiempo final de la producción (en unidades de tiempo, ejm. Segundos.).

Figura 57

Ventana Resultado Final Diagrama Gantt del Sistema de Ordenes de Producción por Maquina



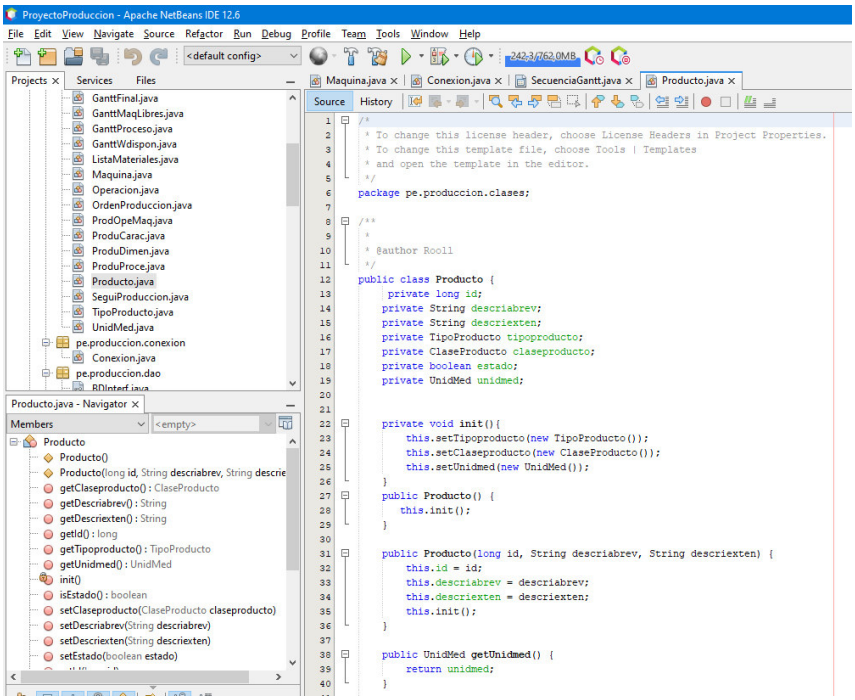
4.7.4 Codificación

Se ha codificado las clases entidad, las clases control, las interfaces, las implementaciones de las interfaces, las clases ventana, la conexión a la base de datos, la secuencia Gantt y la secuencia Recursiva. Se mostrará un ejemplo de cada uno. La documentación completa se indicará en los anexos.

Clase entidad Producto:

Figura 58

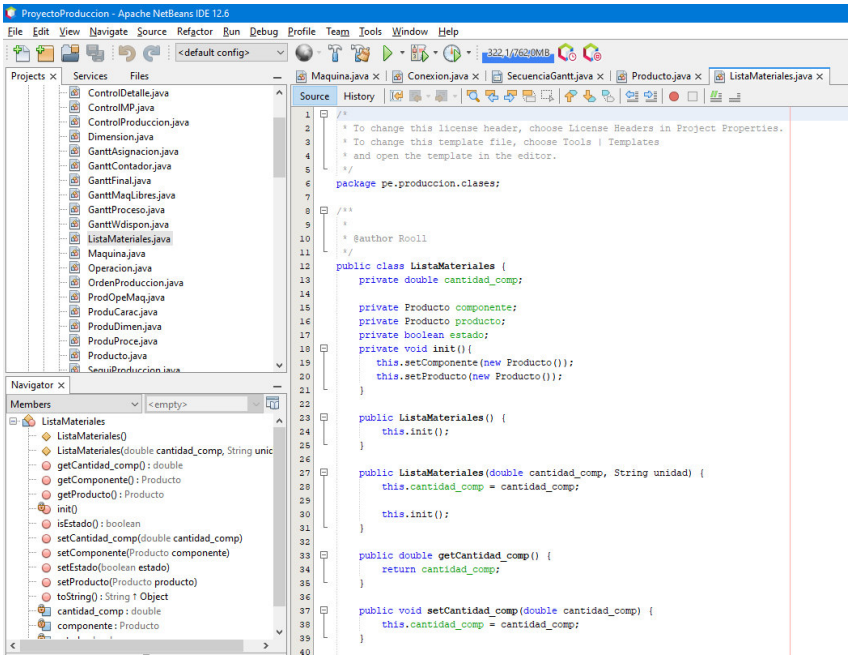
Codificación Clase Entidad Producto



Clase Entidad Lista de Materiales.

Figura 59

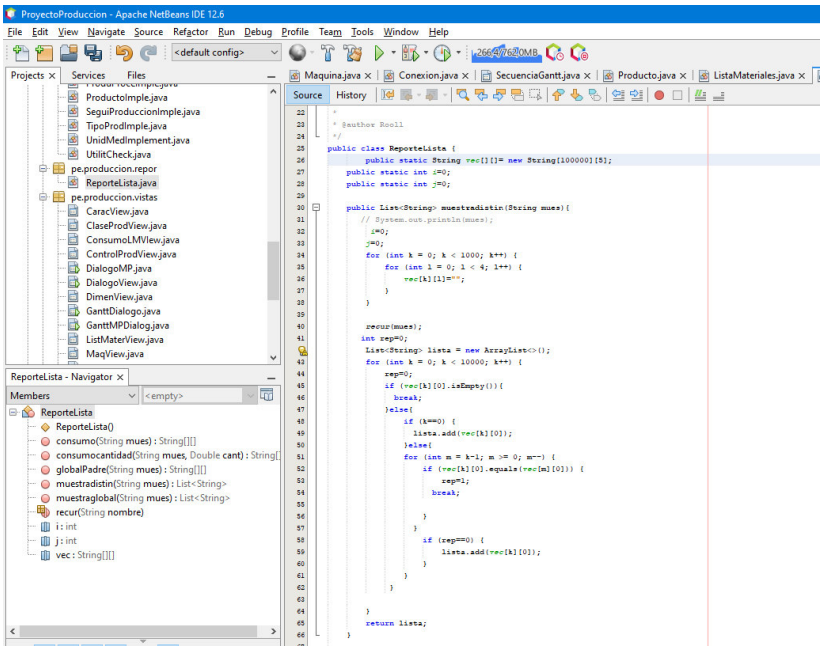
Codificación Clase Entidad Lista de Materiales



La clase control Recursivo (Reporte Lista)

Figura 60

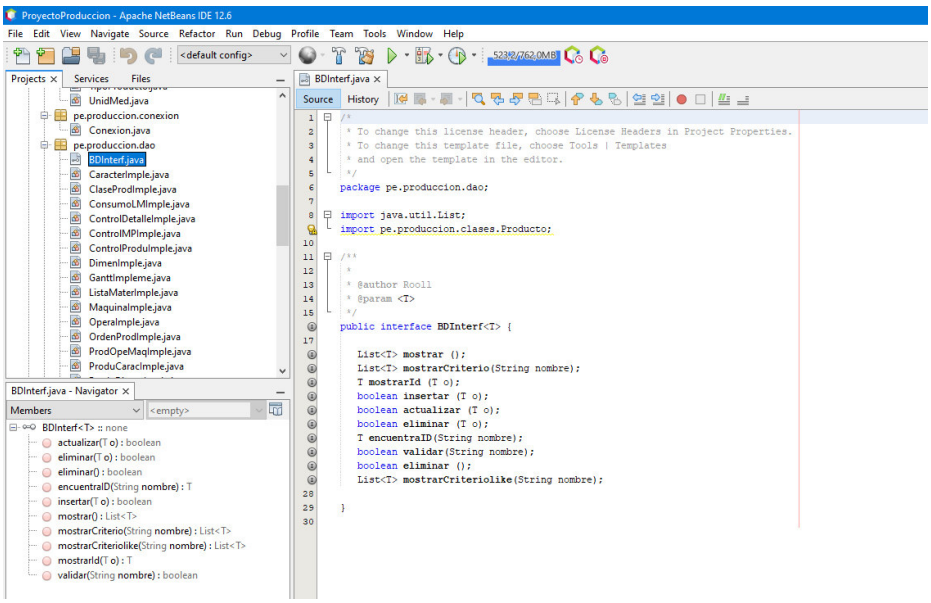
Codificación Clase Recursivo



La clase Interface

Figura 61

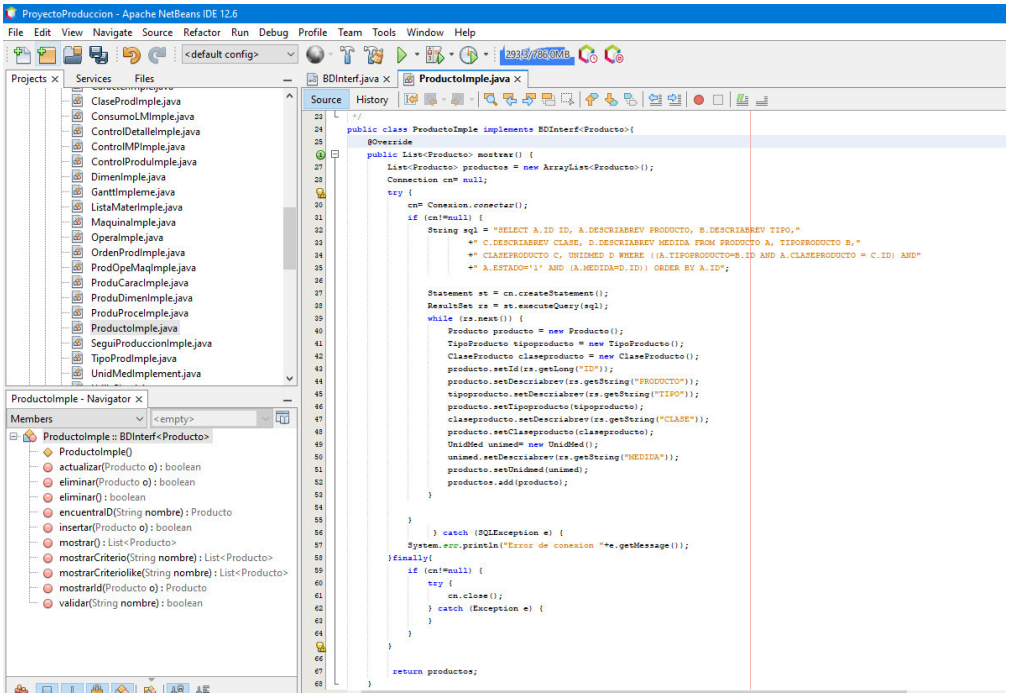
Codificación Clase Interface



La clase Implementación Producto

Figura 62

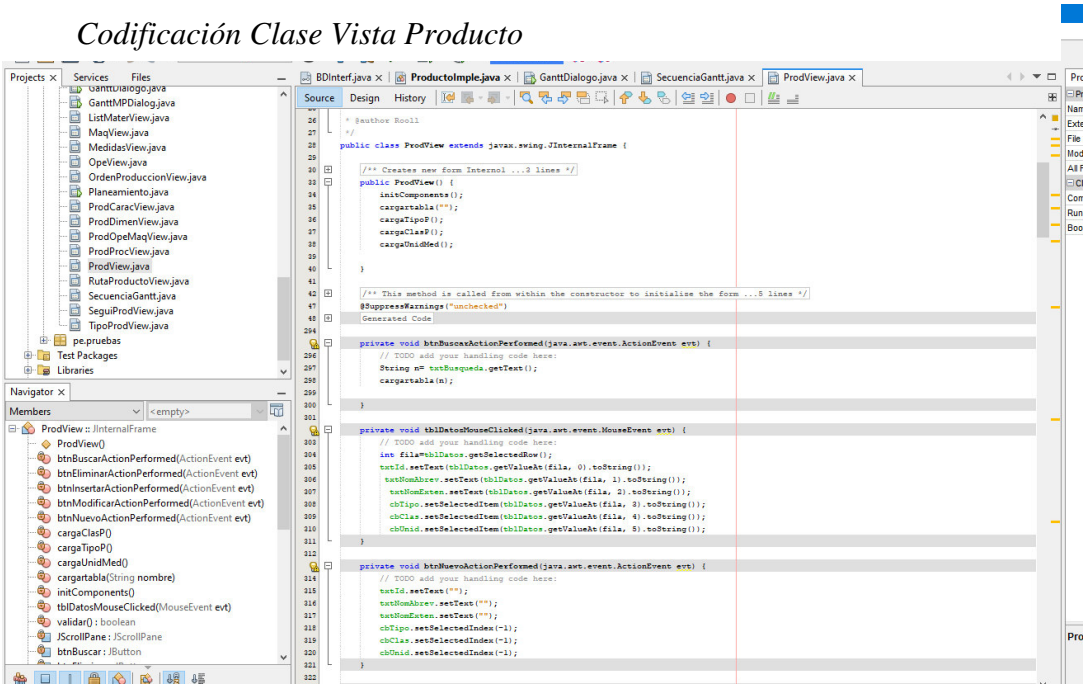
Codificación Clase Implementación Producto



La clase vista Producto

Figura 63

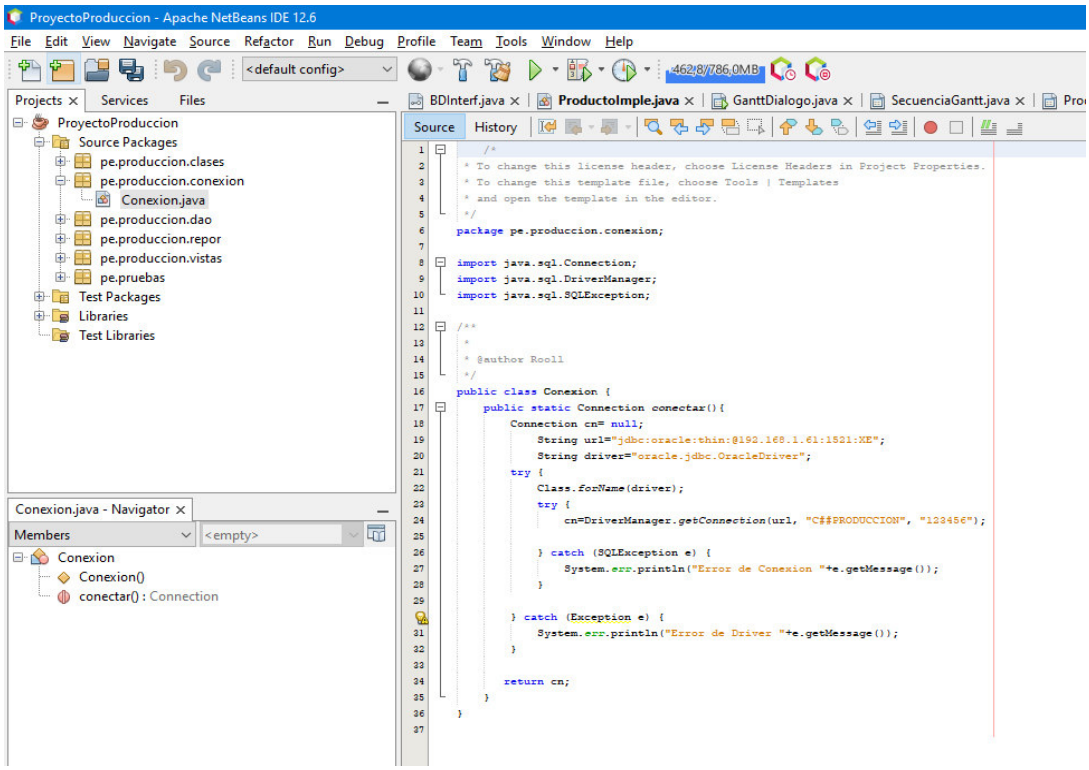
Codificación Clase Vista Producto



La clase Conexión a la base de Datos

Figura 64

Codificación Clase Conexión a Base de Datos



4.8 Resultados Descriptivos

1. Validez de la Evaluación del Instrumento

Para validar los instrumentos de observación se sometió a la evaluación propia de un grupo de expertos en el área (3), ellos a su juicio determinaron la validez y consistencia de los mismos.

Una vez implementado el Sistema, se procede a recolectar la data con el instrumento, tanto para el PreTest (Sin Sistema Informático) como para el ProTest (con Sistema Informático) considerando la medición de las variables dependientes (dimensiones): Tiempo de Programar la Producción, Tiempo de Registrar Lista de Materiales y Visualizar y Tiempo de Registrar Rutas de Producción y visualizar resultando el siguiente cuadro:

Tabla 32

Cuadro de Tomas de Tiempo Pre y Post Sistema de Programación

	PRE-TEST			PRO-TEST		
ITEM	Tiempo de Programar la Produccion (MIN)	Tiempo de Registrar Lista de Materiales y Visualizar (MIN)	Tiempo de Registrar Rutas de Produccion y Visualizar (MIN)	Tiempo de Programar la Produccion (MIN)	Tiempo de Registrar Lista de Materiales y Visualizar (MIN)	Tiempo de Registrar Rutas de Produccion y Visualizar (MIN)
1	214	18	9	2	9	4
2	249	20	10	2	6	9
3	129	13	8	1	8	5
4	207	11	20	2	6	8
5	148	11	19	2	7	9
6	256	17	14	2	4	6
7	240	15	20	3	6	7
8	183	20	15	1	7	9
9	226	13	13	1	6	7
10	243	20	18	2	7	6
11	117	18	10	2	6	4
12	207	14	8	3	6	7
13	225	18	8	3	4	5
14	173	18	11	2	7	8
15	177	17	20	1	9	9

Fuente: Elaboración Propia

A continuación se muestran tablas con estadística descriptiva para cada uno de los indicadores en estudio, tanto para el Pre-Test y el Post-Test.

2. Dimensión Tiempo Programación de la Producción (Pre-Test y Pro-Test)

Figura 65

Estadística Descriptiva Variable Gestión Industrial -Dimensión: Tiempo Programa de Producción

Descriptivos

Estadísticos descriptivos

	N	Rango	Mínimo	Máximo	Media	
	Estadístico	Estadístico	Estadístico	Estadístico	Estadístico	Error estándar
T_Prog_Prod_Pre	15	139,00	117,00	256,00	199,6000	11,28834
T_Prog_Prod_Post	15	2,00	1,00	3,00	1,9333	,18170
N válido (por lista)	15					

Estadísticos descriptivos

	Desv. estándar	Varianza	Asimetría		Curtosis
	Estadístico	Estadístico	Estadístico	Error estándar	Estadístico
T_Prog_Prod_Pre	43,71956	1911,400	-,593	,580	-,684
T_Prog_Prod_Post	,70373	,495	,092	,580	-,669
N válido (por lista)					

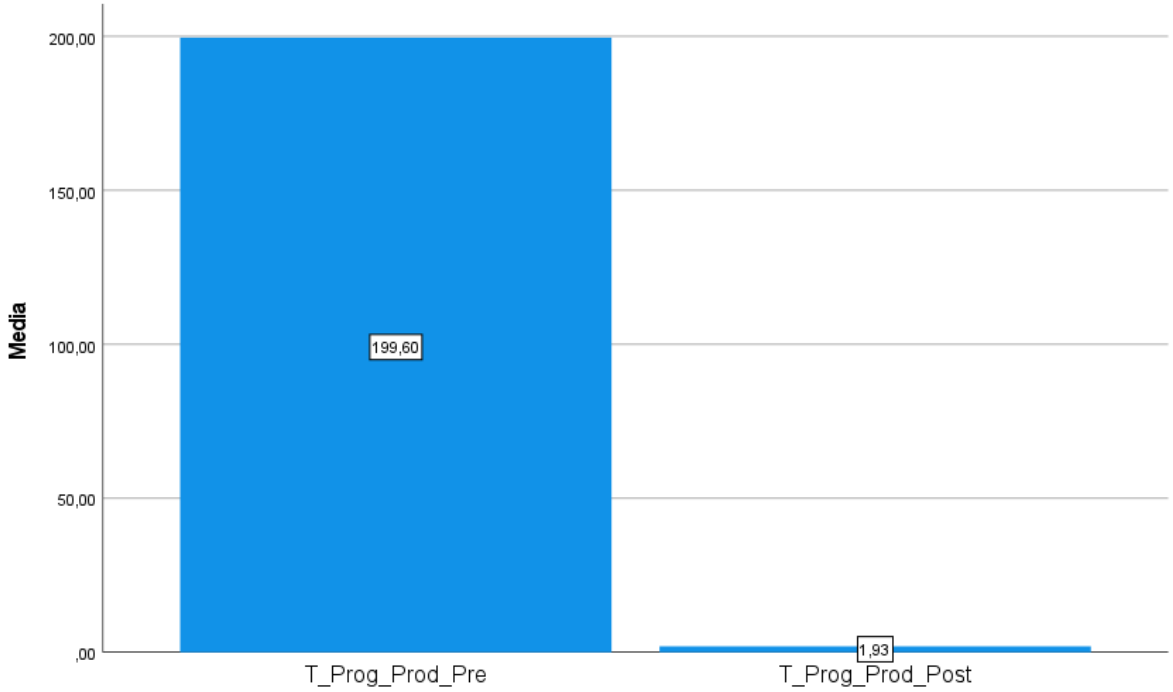
Estadísticos descriptivos

	Curtosis
	Error estándar
T_Prog_Prod_Pre	1,121
T_Prog_Prod_Post	1,121
N válido (por lista)	

Graficamos la comparación de Medias

Figura 66

Grafica Comparación de Medias Variable Gestión Industrial -Dimensión: Tiempo Programa de Producción Pre-Post



Interpretación:

De acuerdo al análisis se obtuvo una media de 199.6 min para programar las ordenes sin el Sistema Informático, con las herramientas habituales de Excel, y luego usando el Sistema Informático se obtuvo una media de 1.93 min para programar las ordenes en el Sistema. Se evidencia una gran diferencia en el Tiempo de la Programación de Ordenes cuando se usa el Sistema Propuesto. Resultando en una disminución del 99.03% en el tiempo de procesamiento. Este resultado se va a corroborar con la contrastación de las Hipótesis.

3. Dimensión Tiempo Lista de Materiales (Pre-Test y Pro-Test)

Figura 67

Estadística Descriptiva Variable Gestión Industrial -Dimensión: Tiempo Gestión Lista de Materiales

Estadísticos descriptivos

	N	Rango	Mínimo	Máximo	Media	
	Estadístico	Estadístico	Estadístico	Estadístico	Estadístico	Error estándar
T_List_Mat_Pre	15	9,00	11,00	20,00	16,2000	,81182
T_List_Mat_Post	15	5,00	4,00	9,00	6,5333	,37628
N válido (por lista)	15					

Estadísticos descriptivos

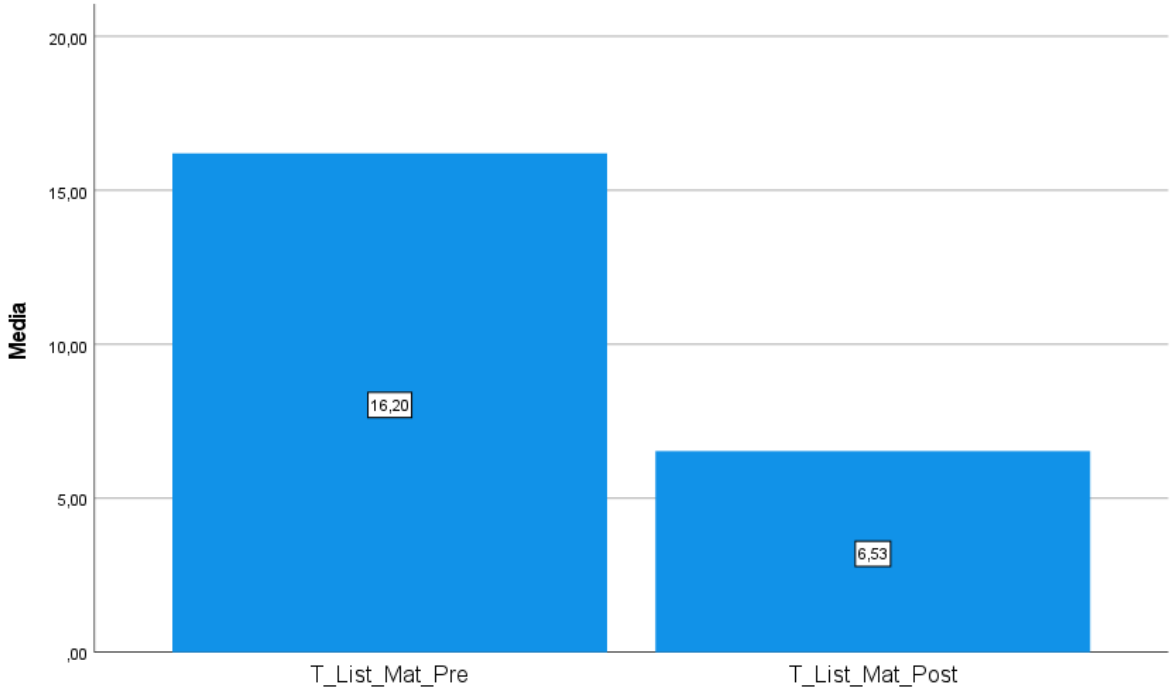
	Desv. estándar	Varianza	Asimetría		Curtosis
	Estadístico	Estadístico	Estadístico	Error estándar	Estadístico
T_List_Mat_Pre	3,14416	9,886	-,451	,580	-1,109
T_List_Mat_Post	1,45733	2,124	,004	,580	,171
N válido (por lista)					

Estadísticos descriptivos

	Curtosis
	Error estándar
T_List_Mat_Pre	1,121
T_List_Mat_Post	1,121
N válido (por lista)	

Figura 68

Grafica Comparación de Medias Variable Gestión Industrial -Dimensión: Gestión Lista de Materiales Pre-Post



Interpretación:

De acuerdo al análisis se obtuvo una media 16.2 min para Gestionar la Lista de Materiales sin el Sistema Informático, con las herramientas habituales de Excel, y luego usando el Sistema Informático se obtuvo una media de 6.53 min Gestionar la Lista de Materiales en el Sistema (Registrar y Visualizar). Se evidencia una notable diferencia en el Tiempo de la Gestión de la Lista de Materiales cuando se usa el Sistema Propuesto. Resultando en una disminución del 59.69% en el tiempo de procesamiento. Este resultado se va a corroborar con la contrastación de las Hipótesis.

4. Dimensión Tiempo Ruta del Producto (Pre-Test y Pro-Test)

Figura 69

Estadística Descriptiva Variable Gestión Industrial -Dimensión: Tiempo Ruta de Producto Pre-Post

Estadísticos descriptivos

	N	Rango	Mínimo	Máximo	Media	
	Estadístico	Estadístico	Estadístico	Estadístico	Estadístico	Error estándar
T_Ruta_Pre	15	12,00	8,00	20,00	13,5333	1,23777
T_Ruta_Post	15	5,00	4,00	9,00	6,8667	,46667
N válido (por lista)	15					

Estadísticos descriptivos

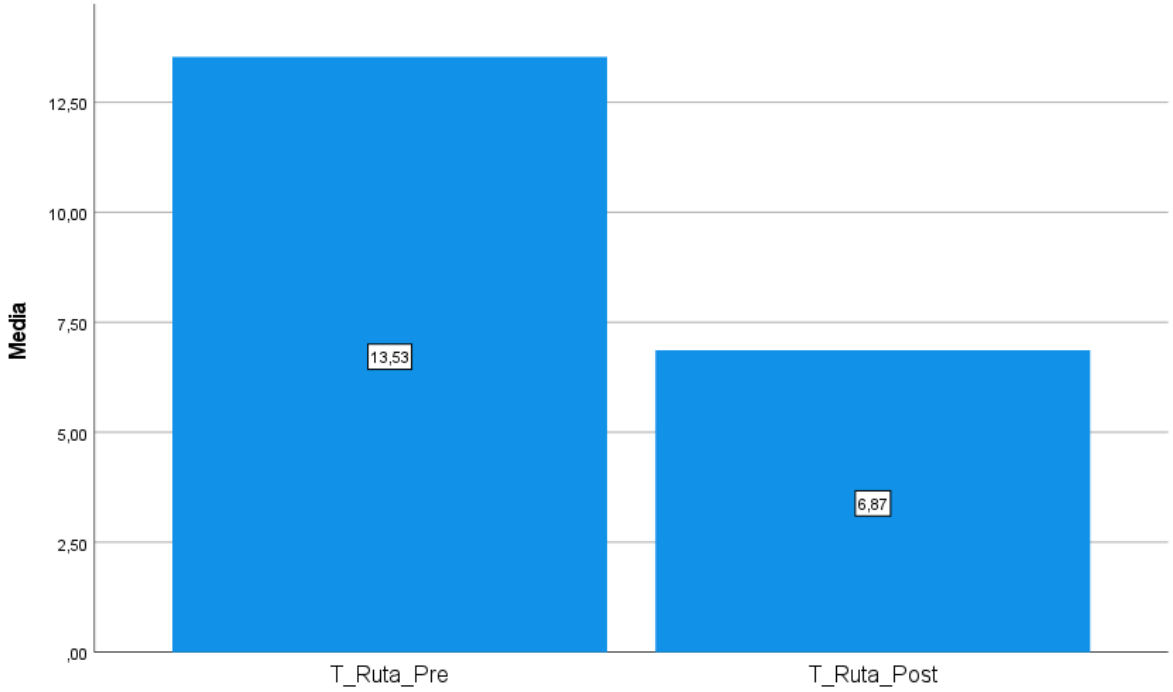
	Desv. estándar	Varianza	Asimetría		Curtosis
	Estadístico	Estadístico	Estadístico	Error estándar	Estadístico
T_Ruta_Pre	4,79385	22,981	,270	,580	-1,661
T_Ruta_Post	1,80739	3,267	-,275	,580	-1,230
N válido (por lista)					

Estadísticos descriptivos

	Curtosis
	Error estándar
T_Ruta_Pre	1,121
T_Ruta_Post	1,121
N válido (por lista)	

Figura 70

Grafica Comparación de Medias Variable Gestión Industrial -Dimensión: Gestión Ruta del Producto Pre-Post



Interpretación:

De acuerdo al análisis se obtuvo una media 13.53 min para Gestionar la Ruta de Productos sin el Sistema Informático, con las herramientas habituales de Excel, y luego usando el Sistema Informático se obtuvo una media de 6.86 min Gestionar la Lista de Materiales en el Sistema (Registrar y Visualizar). Se evidencia una notable diferencia en el Tiempo de la Gestión de la Ruta de Productos cuando se usa el Sistema Propuesto. Resultando en una disminución del 49.29% en el tiempo de procesamiento. Este resultado se va a corroborar con la contrastación de las Hipótesis.

4.9 Resultados Diferenciales

1. Prueba de Normalidad de los Datos

Antes de hacer las pruebas de hipótesis para cada una de las dimensiones de la variable vamos a determinar si hay una distribución normal de los datos para cada dimensión con la prueba de Normalidad de Shapiro-Wilk. Si hay una distribución normal aplicaremos la prueba paramétrica t de Student para la Hipótesis, en caso contrario la Prueba No Paramétrica de Wilcoxon.

Para ello plantearemos la siguiente Hipótesis:

H0: Los datos tienen una distribución normal.

Ha: Los datos no tienen una distribución normal

Nivel de Significancia : Confianza : 95% y Significancia(Alfa): 5%

Se acepta Ha y se rechaza H0, si y solo si: p_valor (nivel de sig.) <= 0,05. Los datos no tienen una distribución normal.

Se acepta H0 y se rechaza Ha, si y solo si: p_valor (nivel de sig.) > 0,05. Los datos si tienen una distribución normal.

Corremos los Datos en SPSS para todas las dimensiones de la Variable Gestión Industrial:

Figura 71

Prueba de Normalidad de los Datos

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
T_Prog_Prod_Pre	,167	15	,200*	,935	15	,323
T_Prog_Prod_Post	,271	15	,004	,815	15	,006
T_List_Mat_Pre	,200	15	,107	,897	15	,087
T_List_Mat_Post	,224	15	,042	,903	15	,107
T_Ruta_Pre	,169	15	,200*	,864	15	,028
T_Ruta_Post	,148	15	,200*	,902	15	,103

*. Esto es un límite inferior de la significación verdadera.

a. Corrección de significación de Lilliefors

Del cuadro anterior en base a la prueba de Normalidad Shapiro-Wilk resumimos los siguiente:

Dimensión Programa de Producción (Post)

Sig = 0.006 < 0.05. Entonces los datos no tienen una distribución normal, usaremos la Prueba No Paramétrica de Wilcoxon para plantear la Hipótesis.

Dimensión Lista de Materiales (Post)

Sig = 0.107 > 0.05. Entonces los datos si tienen una distribución normal, usaremos la Prueba paramétrica t de Student para plantear la Hipótesis.

Dimensión Ruta del Producto (Post)

Sig = 0.103 > 0.05. Entonces los datos si tienen una distribución normal, usaremos la Prueba paramétrica t de Student para plantear la Hipótesis.

2. Pruebas de Hipótesis

Como se describió en el capítulo 1, las hipótesis planteadas son las siguientes:

Hipótesis General:

Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces se mejora la Gestión Industrial.

Hipótesis Especificas:

1. Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces se obtiene rapidez en la Programación de la Producción.
2. Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces se mejora la eficiencia en la Gestión de la Lista de Materiales.
3. Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces se mejora la eficiencia en la Gestión de la Ruta del Producto.

Procederemos a contrastar las Hipótesis Especificas, que una vez validas demostraran la Hipótesis general.

3. Hipótesis Especifica 1 :

Variable: Gestión Industrial

Dimensión: Programa de Producción (Post)

Ha: Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces se obtiene rapidez en la Programación de la Producción

H0: Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces **NO** se obtiene rapidez en la Programación de la Producción

H0, si y solo si: sig (p_valor) > 0,05

Ha, si y solo si: sig (p_valor) < 0,05

Analizamos en SPSS los datos:

Figura 72

Prueba de Hipótesis Especifica 1

Pruebas NPar

Prueba de rangos con signo de Wilcoxon

Rangos

		N	Rango promedio	Suma de rangos
T_Prog_Prod_Post - T_Prog_Prod_Pre	Rangos negativos	15[a]	8,00	120,00
	Rangos positivos	0[b]	,00	,00
	Empates	0[c]		
	Total	15		

a. T_Prog_Prod_Post < T_Prog_Prod_Pre
b. T_Prog_Prod_Post > T_Prog_Prod_Pre
c. T_Prog_Prod_Post = T_Prog_Prod_Pre

Estadísticos de prueba[a]

	T_Prog_Prod_Post - T_Prog_Prod_Pre
Z	-3,408[b]
Sig. asin. (bilateral)	<.001

a. Prueba de rangos con signo de Wilcoxon
b. Se basa en rangos positivos.

Del Resultado el Nivel de Significancia es <0.001 lo que es < a 0.05, por lo que se acepta la Ha, por lo que el Sistema de Programación de Ordenes para empresas de fabricación

tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos SI genera rapidez en la Programación de la Producción.

4. **Hipótesis Especifica 2:**

Variable: Gestión Industrial

Dimensión: Lista de Materiales (Post)

H0: NO hay diferencia Significativa Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos con respecto a la rapidez en el Tiempo de la Gestión en Lista de Materiales.

H1: SI hay diferencia Significativa Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos con respecto a la rapidez en el Tiempo de la Gestión en Lista de Materiales.

Nivel de Confianza 95%, Alpha =5%.

Si P-valor <= Alpha . Se rechaza H0 y se acepta H1. Entonces El sistema informático SI mejora la Gestión de Lista de Materiales

Si P-valor > Alpha . Se acepta H0 y se rechaza H1. Entonces El sistema informático NO mejora la Gestión de Lista de Materiales

Analizamos en SPSS

Figura 73

Prueba de Hipótesis Especifica 2

Prueba de muestras emparejadas

		Diferencias emparejadas			
		Media	Desv. estándar	Media de error estándar	95% de intervalo de confianza de . Inferior
Par 1	T_List_Mat_Pre - T_List_Mat_Post	9,66667	3,45722	,89265	7,75212

Prueba de muestras emparejadas

		Diferencias ... 95% de intervalo de confianza de ... Superior	t	gl	Significación P de un factor
Par 1	T_List_Mat_Pre - T_List_Mat_Post	11,58121	10,829	14	<.001

Prueba de muestras emparejadas

		Significación P de dos factores
Par 1	T_List_Mat_Pre - T_List_Mat_Post	<.001

Como (P<0.001) <Alpha(0.05) entonces se acepta H1. Entonces El sistema informático SI mejora la Gestión de Lista de Materiales

5. **Hipótesis Especifica 2:**

Variable: Gestión Industrial

Dimensión: Ruta del Producto (Post)

H0: NO hay diferencia Significativa Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos con respecto a la rapidez en el Tiempo de la Gestión de la Ruta del Producto.

H1: SI hay diferencia Significativa Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos con respecto a la rapidez en el Tiempo de la Gestión en la Ruta del Producto.

Nivel de Confianza 95%, Alpha =5%.

Si P-valor <= Alpha . Se rechaza H0 y se acepta H1. Entonces El sistema informático SI mejora la Gestión de la Ruta del Producto

Si P-valor > Alpha . Se acepta H0 y se rechaza H1. Entonces El sistema informático NO mejora la Gestión de la Ruta del Producto

Analizamos en SPSS

Figura 74

Prueba de Hipótesis Especifica 2

Prueba de muestras emparejadas

		Diferencias emparejadas			
		Media	Desv. estándar	Media de error estándar	95% de intervalo de confianza de .. Inferior
Par 1	T_Ruta_Pre - T_Ruta_Post	6,66667	4,11733	1,06309	4,38657

Prueba de muestras emparejadas

		Diferencias ... 95% de intervalo de confianza de ...			Significación
		Superior	t	gl	P de un factor
Par 1	T_Ruta_Pre - T_Ruta_Post	8,94677	6,271	14	<.001

Prueba de muestras emparejadas

		Significación
		P de dos factores
Par 1	T_Ruta_Pre - T_Ruta_Post	<.001

Como (P<0.001) <Alpha(0.05) entonces se acepta H1. Entonces El sistema informático SI mejora la Gestión de las Rutas del Producto.

6. Análisis de la Hipótesis General:

Luego de la evaluación de las 3 Hipótesis especificas planteadas, podemos afirmar que se verifica el cumplimiento de la Hipótesis General:

Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces se mejora la Gestión Industrial.

Por lo que se puede concluir que el Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop Ensamble usando algoritmos heurísticos y recursivos permite mejorar la Gestión Industrial.

V DISCUSIÓN DE RESULTADOS

Como resultado de la contrastación de la Hipótesis General planteada podemos aceptar que el Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos mejora la Gestión Industrial, siendo esta última variable sustentada en los indicadores Tiempo de Programar la producción, tiempo de registrar Lista de Materiales-Visualizar y Tiempo de Registrar Rutas de Producción-Visualizar. Es decir el sistema de Programación de Ordenes aumenta la velocidad en la Programación de la Producción bajando drásticamente el tiempo de esa actividad, de igual modo para el caso de Gestionar la Lista de Materiales y las Rutas del Producto.

Estos resultados guardan relación con los sostenido por Tello Ojeda (2014) donde la autora demuestra que la implementación de un algoritmo Genético para el problema de Job Shop Sheduling genera resultados altamente confiables. El algoritmo genético es parte de la misma familia de los algoritmos heurísticos que tratan el problema de la Optimización Combinatoria aunque de manera diferente. Mientras los algoritmos genéticos simulan los procesos biológicos para tratar de alcanzar una solución óptima, los algoritmos heurísticos van construyendo la solución óptima paso a paso evaluando siempre un parámetro en cada etapa (en el caso particular de la presente investigación este parámetro es la regla de decisión). La autora se concentra en minimizar el MakeSpan (el tiempo total en el cual se terminan todas las tareas), en cambio en la presente investigación nos concentramos en las reglas de prioridad, donde el MakeSpan depende directamente de la regla de prioridad elegida por el usuario del Sistema. En ambos trabajos resaltamos la aplicación sobre el problema Job Shop, aunque existe el aspecto del Ensamble de Productos (que si es considerado en la presente investigación).

De igual manera en Ramirez Rodriguez (2006) el autor demuestra que la implementación de un algoritmo Grasp en un ambiente de Flow Shop (misma secuencia de

producción para los productos) genera mejoras en la programación de los trabajos en planta. Si bien la investigación esta concentrada en un Flow Shop (diferente a JobShop que abordamos en el presente trabajo, en este último diferentes secuencias para los productos) el uso de un algoritmo como Grasp que es un metaheurístico (otra rama de los algoritmos Heurísticos) viene a confirmar el uso e investigación de estas técnicas en la resolución de estos problemas de secuenciación en planta.

VI CONCLUSIONES

- En esta investigación se desarrolló un Sistema de Programación de Ordenes para empresas de fabricación Tipo Job Shop-Ensamble con algoritmos heurísticos y recursivos que mejora la Gestión Industrial. La mejora en la Gestión Industrial se demostró mediante la disminución en el Tiempo de ejecución de las 3 dimensiones de la Gestión Industrial: Tiempo de Programar la Producción (desde 199.6 min a 1.93 min- disminución de 99.03%), Tiempo de Registrar Lista de Materiales (desde 16.2 min a 6.53 min- disminución de 59.69%) y el Tiempo de Registrar Rutas de Producción (desde 13.53 min a 6.86 min- disminución de 49.29%).

- En esta investigación se desarrolló un Sistema de Programación de Ordenes para empresas de fabricación Tipo Job Shop-Ensamble con algoritmos heurísticos y recursivos para obtener rapidez en la Programación de la Producción.. El Tiempo de Programar la Producción disminuyo en un 99.03% y se usó la Prueba No paramétricas de Wilconoz para probar la Hipótesis: “Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos entonces SI se obtiene rapidez en la Programación de la Producción” ya que la variable presenta distribución No Normal (según prueba Shapiro-Wilk). Se obtuvo el estadístico $p<0.05$ por lo que se aceptó la Hipótesis.

- En esta investigación se desarrolló un Sistema de Programación de Ordenes para empresas de fabricación Tipo Job Shop-Ensamble con algoritmos heurísticos y recursivos para mejorar la eficiencia en la Gestión de la Lista de Materiales. El Tiempo de Registrar Lista de Materiales disminuyo en un 59.69% y se usó la Prueba paramétricas T-Student para probar la

Hipótesis “SI hay diferencia Significativa si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos con respecto a la rapidez en el Tiempo de la Gestión en Lista de Materiales” ya que la variable presenta distribución Normal (según prueba Shapiro-Wilk).). Se obtuvo el estadístico $p<0.05$ por lo que se aceptó la Hipótesis.

- En esta investigación se desarrolló un Sistema de Programación de Ordenes para empresas de fabricación Tipo Job Shop-Ensamble con algoritmos heurísticos y recursivos para mejorar la eficiencia en la Gestión de la Ruta del Producto. El Tiempo de Registrar la Ruta del Producto disminuyo en un 49.29% y se usó la Prueba paramétricas T-Student para probar la Hipótesis “SI hay diferencia Significativa si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heurísticos y recursivos con respecto a la rapidez en el Tiempo de la Gestión de la Ruta del Producto” ya que la variable presenta distribución Normal (según prueba Shapiro-Wilk). Se obtuvo el estadístico $p<0.05$ por lo que se aceptó la Hipótesis.

- Lo mas importante del desarrollo del Sistema de Programación Ordenes para empresas de fabricación Tipo Job Shop-Ensamble con algoritmos heurísticos y recursivos fue que se elaboró un Gantt de Programación de ordenes secuenciados por cada maquina y Visualización de Lista-Ruta de Materiales a consecuencia de los ingresos de la data de: Lista de Materiales, Ruta de Productos, Ordenes de Producción, prioridad de las Ordenes y Disponibilidad de Maquinas además de incluir los parámetros de: Ensamble de los productos, un producto puede pasar mas de 1 vez por una máquina, maquinas no disponibles al inicio y manejo de componentes comunes, ya que estos últimos 4 parámetros mencionados no son considerados en las otras investigaciones encontradas en los antecedentes. Esto hace que la

investigación presente sea de real utilidad en el ambiente industrial peruano, donde el 99% de las empresas de manufactura son Pymes con procesos Job-Shop Ensamble, y donde los pedidos de los clientes tienen una gran variedad de productos en lotes pequeños que tienen diferentes listas de materiales, rutas de producción y todo esto dentro de un ambiente altamente dinámico.

- Lo que mas ayudo en el desarrollo del Sistema fue el conocimiento práctico del proceso de programación de la producción en las fábricas de producción tipo Job Shop Ensamble porque permitió elaborar el diseño de la base de datos, el algoritmo recursivo para la lista de materiales-ruta del producto (concepto árbol Padre-Hijo) y el algoritmo heurístico para la programación de las ordenes en las máquinas de producción resultando en el Gantt de Programación.

- Lo mas difícil en el desarrollo del Sistema fue la transcripción de los algoritmos recursivos y heurísticos al lenguaje de programación Java, porque hubo que aprender sintaxis propia del lenguaje para poder plasmar los algoritmos.

- El costo del Sistema (S/. 300) para una empresa PYME esta dentro de sus posibilidades, ya que según el ministerio de la Producción, las Pymes tienen ventas anuales entre S/. 742,500 y S/. 11’385,000, tomando en cuenta el soporte al Usuario de 6 meses para darle seguimiento al Sistema de Programación en el trabajo diario.

- En un ambiente industrial Job Shop Ensamble (productos con diferentes rutas de producción y con ensambles de componentes) propias de las PYME en el Perú, se demostró

la utilidad de los algoritmos heurísticos y recursivos para abordar el problema de la programación de ordenes estos ambientes industriales (problema Computable de Complejidad NP), donde la cantidad enorme de opciones posibles hace físicamente imposible resolver mediante algoritmos clásicos en las computadoras actuales. El Sistema informático permitirá, como parte de la Mejora en la Gestión Industrial lograda, facilitar el ingreso de la información al sistema, centralizar la información industrial de planta, agilizar la programación de órdenes, rápidas respuestas de fechas de entrega y comparar escenarios de Gantt a variantes de prioridad de órdenes.

VII RECOMENDACIONES

- Se recomienda implementar el Sistema de Programación de Ordenes en las empresas de fabricación PYMES (de realidad Job Shop Ensamble), ya que se ha demostrado que mejora la Gestión Industrial en lo referente a la rapidez en la programación de la producción, elaboración de lista de materiales y de las rutas del producto, considerando también el bajo costo de implementación.

- Todas las entradas y salidas del Sistema consideradas en este trabajo son las mas fundamentales, pero no son todas las posibles. Se recomienda aumentar mas ítems de entrada y salida en futuras investigaciones. Los problemas que existen en la Gestión Industrial son variados y no se han considerado todas en este trabajo. Ejemplo de Nuevas Entradas y Salidas: Listas de Materiales y Rutas de Producción Alternativas, Maquinas Alternativas, Costos de los Materiales y Productos, Lotes mínimos de Producción y Gantt costeados. Como nuevos parámetros se podría establecer: considerar empezar el siguiente trabajo sin esperar a que el anterior termine, programar en base al proceso mas lento o cuello de botella y establecer producciones que respeten el lote mínimo de producción. Todas estas nuevas posibilidades enriquecen nuevas investigaciones para el problema de la programación de Ordenes.

- El problema planteado en la presente investigación tomo como referencia la experiencia práctica en Plantas tipo Job Shop Ensamble, pero puede seguirse el estudio en plantas de líneas en Serie, donde la secuencia es única y con lotes de producción definidos. En

este caso se podría determinar Ordenes de Producción como Lotes de Producción y trabajando las demás entradas y parámetros como en la presente investigación.

- Se recomienda seguir investigando acerca de la aplicación de algoritmos Heurísticos en el problema de Programación de Ordenes. Este trabajo uso un tipo de Heurístico basado en Prioridades con resultados eficientes de mejora en la Gestión Industrial, pero el universo de posibilidades de programación permite encontrar otras muchas soluciones también eficientes, como por ejemplo usar algoritmos Genéticos o de Búsqueda Tabú. Estos algoritmos pueden encontrar otras secuencias para la Secuenciación de Ordenes cumpliendo con el precepto de dar soluciones eficientes a problemas complejos.

- Se recomienda escalar el sistema a Ambiente Web con Base de Datos en la Nube, así como en el Dispositivo Móvil, para evitar la limitante de la ubicación física para programar la producción, adicionando a esta mejora el control de la Producción como aspecto que retroalimente al Sistema y actualice el avance de la Producción y así generar Nuevos Gantt de Avance.

VIII REFERENCIAS

Aparicio, Y. y Gonzales, C. (2016). *Metaheurística Greedy Randomized Adaptive Search Procedure Aplicada al Jobshop Scheduling Problem (Jssp)*. [Tesis de pregrado, Universidad Industrial de Santander, Bucaramanga]. Biblioteca UIS. tangara.uis.edu.co/biblioweb/tesis/2016/164752.pdf

Gaither, N. y Frazier, G. (2000). *Administración de Producción y Operaciones*. (8ª ed.).International Thomson Editores.

Garcia, J. (2006). *Modelado mediante Optimización Combinatoria.* [Archivo PDF]. Jose P. Garcia-Sabater. Mi Official Site. personales.upv.es/jpgarcia/LinkedDocuments/MCOIOptimizacionCombinatoria.pdf

Gonzales, C. (2015). *Algoritmos Constructivos para la programación en entornos JobShop flexibles*. [Tesis de pregrado, Universidad de Valladolid, Escuela de Ingenieros Industriales]. Repositorio Documental. https://uvadoc.uva.es/bitstream/handle/10324/18700/TFG-I-442.pdf?sequence=1.

Joyanes, L. (2003). *Fundamentos de Programación. Algoritmos, Estructuras de Datos y Objetos*. (3ª ed.). McGraw-Hill.

Kendall, K. y Kendall, J. (2011). *Análisis y Diseño de Sistemas*. (8ª ed.). Prentice Hall.

Medrano, E. (2007). *Problemas de Scheduling: Heurísticas para el JobShop*. [Tesis de pregrado, Universidad Central de las Villas, Santa Clara]. Repositorio Institucional de la UCLV. https://dspace.uclv.edu.cu/server/api/core/bitstreams/7feb2254-148c-43c6-861a-348cb15fde41/content

Pressman, R. (2005). *Ingeniería del Software*. (6ª ed.). McGraw Hill.

Ramirez, C. (2006). *Un algoritmo Grasp con doble Relajación para resolver el problema del Flow Shop Scheduling*. [Tesis de pregrado, Pontificia Universidad Católica del Perú]. Repositorio de Tesis PUCP. https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/368/RAMIREZ_CESAR_UN_ALGORITMO_GRASP_CON_DOBLE_RELAJACION_PARA_RESOLVER_EL_PROBLEMA_DEL_FLOW_SHOP_SCHEDULING.pdf?sequence=1&isAllowed=y

Romano, A. (2016). *Algoritmo "Shifting bottleneck" para la secuenciación de trabajos en Jobshop | | UPV.* [Archivo de video]. https://www.youtube.com/watch?v=6YHLzqBT8mY

Schach, S. (2006). *Ingeniería de Software Clásica y Orientada a Objetos.* (6ª ed.). McGraw - Hill.

Sipper, Daniel y Bulfin, Roberto (1998). *Planeación y Control de la Producción*. (1ª ed.). McGraw-Hill.

Stern School of Business, NYU. (s.f.). *LEKIN® – Flexible Job-Shop Scheduling System*. New York. http://web-static.stern.nyu.edu/om/software/lekin/

Suarez, O. (18 de abril de 2011). *Una aproximación a la Heurística y Metaheurística*. INGE@UAN – Tendencias de la Ingeniería, 1(1), 44-51. https://revistas.uan.edu.co/index.php/ingeuan/article/view/217/179

Tello, C. (2014). *Algoritmo Evolutivo para Asignar Trabajos a Maquina, con diferente secuencia de Operaciones, desde la perspectiva de la Programación Detallada*. [Tesis de pregrado, Universidad Nacional de Piura]. Repositorio Institucional UNP.

http://repositorio.unp.edu.pe/bitstream/handle/UNP/702/IND-TEL-OJE-14.pdf?sequence=1&isAllowed=y

Universitat Politècnica de València – UPV.(28 de enero de 2016). *Algoritmo "Shifting bottleneck" para la secuenciación de trabajos en Jobshop | | UPV*. [Archivo de video].https://www.youtube.com/watch?v=6YHLzqBT8mY

Valle, D. (2013). *Programación de Rutas de Materiales en Configuración de Trabajo JobShop con Algoritmos Genéticos*. [Tesis de maestría, Centro de Innovación Aplicada en Tecnologías Competitivas]. Repositorio Institucional CIATEC. https://ciatec.repositorioinstitucional.mx/jspui/bitstream/1019/52/1/Tesis_%20David%20A.%20Valle.pdf

Vicentini, F. y Puddu, S. (2003). *Algoritmos Heurísticos y el Problema del JobShop Scheduling*. [Archivo PDF]. Universidad de Buenas Aires, Facultad de Ciencias Exactas y Naturales, Buenos Aires. www.dm.uba.ar/materias/optimizacion/2008/1/tesis.PDF

Vidal, A. (1 de julio de 2013). *Algoritmos Heurísticos en Optimización.* Universidad de Santiago de Compostela. eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_782.pdf

Yepes, V. (10 de septiembre de 2014). *¿Qué es la optimización combinatoria?*. victoryepes.blogs.upv.es. https://victoryepes.blogs.upv.es/2014/09/10/optimizacion-combinatoria/

IX ANEXOS

9.1 ANEXO A Matriz de Consistencia

SISTEMA DE PROGRAMACION DE ORDENES EN AMBIENTE INDUSTRIAL JOBSHOP-ENSAMBLE UTILIZANDO ALGORITMOS HEURISTICOS Y RECURSIVOS PARA MEJORAR LA GESTION INDUSTRIAL						
PROBLEMA	OBJETIVOS		JUSTIFICACION	HIPOTESIS	VARIABLES	INDICADORES
	GENERAL	ESPECIFICOS				
Problema General ¿Cómo el Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble utilizando algoritmos heuristicos y recursivos permitirá mejorar la Gestión Industrial? **Problemas Secundarios** • ¿ Cómo el Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble utilizando algoritmos heuristicos y recursivos permitirá obtener rapidez en la Programacion de la Produccion? • ¿ Cómo el Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble utilizando algoritmos heuristicos y recursivos permitirá obtener eficiencia en la Gestion de Lista de Materiales? • ¿ Cómo el Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble utilizando algoritmos heuristicos y recursivos permitirá obtener eficiencia en la Gestion de la Ruta del Producto?	Desarrollar un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble que utiliza algoritmos heuristicos y recursivos para mejorar la Gestión Industrial.	• Desarrollar un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble que utiliza algoritmos heuristicos y recursivos para obtener rapidez en la Programacion de la Produccion. • Desarrollar un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble que utiliza algoritmos heuristicos y recursivos para mejorar la eficiencia en la Gestion de la Lista de Materiales. • Desarrollar un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble que utiliza algoritmos heuristicos y recursivos para mejorar la eficiencia en la Gestion de la Ruta del Producto.	El presente trabajo permite aportar en la investigación de la programación de ordenes en ambiente JobShop-Ensamble. La demora en la generación de un calendario de trabajo por órdenes para las maquinas dentro la planta industrial y tener repartida la información de producción en diferentes repositorios de datos no permite una eficiente gestión en planta. EL presente trabajo propone una solución acorde con la necesidad de las empresas peruanas. La industria puede beneficiarse de esta investigación, ya que presenta una solución practica a los problemas de gestión de ordenes	**Hipótesis General** Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heuristicos y recursivos entonces se mejora la Gestión Industrial. **Hipótesis Secundarias** • Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heuristicos y recursivos entonces se obtiene rapidez en la Programacion de la Produccion • Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heuristicos y recursivos entonces se mejora la eficiencia en la Gestion de la Lista de Materiales. • Si se elabora un Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heuristicos y recursivos entonces se mejora la eficiencia en la Gestion de la Ruta del Producto.	**Variables Independientes** • Sistema de Programación de Ordenes para empresas de fabricación tipo Job Shop-Ensamble usando algoritmos heuristicos y recursivos. **Variables Dependientes** * Gestion Industrial	**Programacion de la Produccion** • Tiempo de Programar la Produccion **Gestion de Lista de Materiales :** • Tiempo de Registrar Lista de Materiales y Visualizar **Gestion de Rutas del Producto :** • Tiempo de Registrar Rutas de Produccion y Visualizar

9.2 ANEXO B Codificación Clases

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.clases;

/**
 *
 * @author Rooll
 */
public class ConsumoLM {
    private Producto producto;
    private Producto componente;
    private double cantidad;
    private Producto padre;

    private void init(){
    this.setProducto(new Producto());
    this.setComponente(new Producto());
    this.setPadre(new Producto());

    }

    public ConsumoLM() {
        this.init();
    }

    public ConsumoLM(Producto producto, Producto componente, double cantidad) {
        this.producto = producto;
        this.componente = componente;
        this.cantidad = cantidad;
        this.init();
    }

    public Producto getPadre() {
        return padre;
    }

    public void setPadre(Producto padre) {
        this.padre = padre;
    }


    public Producto getProducto() {
        return producto;
    }

    public void setProducto(Producto producto) {
        this.producto = producto;
    }

    public Producto getComponente() {
        return componente;
    }

    public void setComponente(Producto componente) {
        this.componente = componente;
    }

    public double getCantidad() {
        return cantidad;
    }

    public void setCantidad(double cantidad) {
        this.cantidad = cantidad;
    }

    @Override
    public String toString() {
        return "ConsumoLM{" + "producto=" + producto + ", componente=" + componente + ", cantidad=" + cantidad + '}';
    }



}


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
package pe.produccion.clases;

/**
 *
 * @author Rooll
 */
public class GanttFinal {
    private long idprograma;
     private Maquina maq;
    private long op;
    private Producto comp;
    private Producto padre;
    private double tinicio;
    private double tfinal;

    public GanttFinal() {
        this.init();
    }

    private void init(){
        this.setMaq(new Maquina());
        this.setPadre(new Producto());
        this.setComp(new Producto());
    }

    public GanttFinal(long idprograma, Maquina maq, long op, Producto comp, Producto padre, double tinicio, double tfinal) {
        this.idprograma = idprograma;
        this.maq = maq;
        this.op = op;
        this.comp = comp;
        this.padre = padre;
        this.tinicio = tinicio;
        this.tfinal = tfinal;
        this.init();
    }

    public long getIdprograma() {
        return idprograma;
    }

    public void setIdprograma(long idprograma) {
        this.idprograma = idprograma;
    }

    public Maquina getMaq() {
        return maq;
    }

    public void setMaq(Maquina maq) {
        this.maq = maq;
    }

    public long getOp() {
        return op;
    }

    public void setOp(long op) {
        this.op = op;
    }

    public Producto getComp() {
        return comp;
    }

    public void setComp(Producto comp) {
        this.comp = comp;
    }

    public Producto getPadre() {
        return padre;
    }

    public void setPadre(Producto padre) {
        this.padre = padre;
    }

    public double getTinicio() {
        return tinicio;
    }

    public void setTinicio(double tinicio) {
        this.tinicio = tinicio;
    }
```

```
    public double getTfinal() {
        return tfinal;
    }

    public void setTfinal(double tfinal) {
        this.tfinal = tfinal;
    }

    @Override
    public String toString() {
        return "GanttFinal{" + "idprograma=" + idprograma + ", maq=" + maq + ", op=" + op + ", comp=" + comp + ", padre=" + padre + ", tinicio="
+ tinicio + ", tfinal=" + tfinal + '}';
    }




}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.clases;

/**
 *
 * @author Rooll
 */
public class GanttProceso {
    private OrdenProduccion op;
    private Producto componente;
    private Producto padre;
    private double cantipadre;
    private Maquina maquina;
    private double saldo;
    private double tiemposaldo;
    private double ultimo;
    private String estado;
    private double asignado;
    private double CompletoVirtual;

    private void init(){
        this.setOp(new OrdenProduccion());
       this.setComponente(new Producto());
       this.setPadre(new Producto());
       this.setMaquina(new Maquina());
    }

    public GanttProceso() {
        this.init();
    }

    public GanttProceso(OrdenProduccion op, Producto componente, Producto padre, double cantipadre, Maquina maquina, double saldo, double
tiemposaldo, double ultimo, String estado, double asignado, double CompletoVirtual) {
        this.op = op;
        this.componente = componente;
        this.padre = padre;
        this.cantipadre = cantipadre;
        this.maquina = maquina;
        this.saldo = saldo;
        this.tiemposaldo = tiemposaldo;
        this.ultimo = ultimo;
        this.estado = estado;
        this.asignado = asignado;
        this.CompletoVirtual = CompletoVirtual;
        this.init();

    }

    public OrdenProduccion getOp() {
        return op;
    }

    public void setOp(OrdenProduccion op) {
        this.op = op;
    }

    public Producto getComponente() {
        return componente;
    }

    public void setComponente(Producto componente) {
        this.componente = componente;
```

```
    }

    public Producto getPadre() {
        return padre;
    }

    public void setPadre(Producto padre) {
        this.padre = padre;
    }

    public double getCantipadre() {
        return cantipadre;
    }

    public void setCantipadre(double cantipadre) {
        this.cantipadre = cantipadre;
    }

    public Maquina getMaquina() {
        return maquina;
    }

    public void setMaquina(Maquina maquina) {
        this.maquina = maquina;
    }

    public double getSaldo() {
        return saldo;
    }

    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }

    public double getTiemposaldo() {
        return tiemposaldo;
    }

    public void setTiemposaldo(double tiemposaldo) {
        this.tiemposaldo = tiemposaldo;
    }

    public double getUltimo() {
        return ultimo;
    }

    public void setUltimo(double ultimo) {
        this.ultimo = ultimo;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public double getAsignado() {
        return asignado;
    }

    public void setAsignado(double asignado) {
        this.asignado = asignado;
    }

    public double getCompletoVirtual() {
        return CompletoVirtual;
    }

    public void setCompletoVirtual(double CompletoVirtual) {
        this.CompletoVirtual = CompletoVirtual;
    }

    @Override
    public String toString() {
        return "GanttProceso{" + "op=" + op + ", componente=" + componente + ", padre=" + padre + ", cantipadre=" + cantipadre + ", maquina=" +
maquina + ", saldo=" + saldo + ", tiemposaldo=" + tiemposaldo + ", ultimo=" + ultimo + ", estado=" + estado + ", asignado=" + asignado + ", CompletoVirtual="
+ CompletoVirtual + '}';
    }


}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.clases;

/**
 *
 * @author Rooll
 */
public class ListaMateriales {
    private double cantidad_comp;

    private Producto componente;
    private Producto producto;
    private boolean estado;
    private void init(){
       this.setComponente(new Producto());
       this.setProducto(new Producto());
    }

    public ListaMateriales() {
        this.init();
    }

    public ListaMateriales(double cantidad_comp, String unidad) {
        this.cantidad_comp = cantidad_comp;

        this.init();
    }

    public double getCantidad_comp() {
        return cantidad_comp;
    }

    public void setCantidad_comp(double cantidad_comp) {
        this.cantidad_comp = cantidad_comp;
    }



    public Producto getComponente() {
        return componente;
    }

    public void setComponente(Producto componente) {
        this.componente = componente;
    }

    public Producto getProducto() {
        return producto;
    }

    public void setProducto(Producto producto) {
        this.producto = producto;
    }

    public boolean isEstado() {
        return estado;
    }

    public void setEstado(boolean estado) {
        this.estado = estado;
    }

    @Override
    public String toString() {
        return "ListaMateriales{" +componente+ " cantidad_comp=" + cantidad_comp + " "+producto+'}';
    }

}


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.clases;

/**
 *
 * @author Rooll
 */
public class Maquina {
```

```
     private long id;
    private String descriabrev;
    private String descriexten;
private boolean estado;

    public Maquina(){
    }

    public Maquina (long id, String descriabrev, String descriexten) {
        this.id = id;
        this.descriabrev = descriabrev;
        this.descriexten = descriexten;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getDescriabrev() {
        return descriabrev;
    }

    public void setDescriabrev(String descriabrev) {
        this.descriabrev = descriabrev;
    }

    public String getDescriexten() {
        return descriexten;
    }

    public void setDescriexten(String descriexten) {
        this.descriexten = descriexten;
    }

    public boolean isEstado() {
        return estado;
    }

    public void setEstado(boolean estado) {
        this.estado = estado;
    }

    @Override
    public String toString() {
        return "Maquina{" + "id=" + id + ", descriabrev=" + descriabrev + ", descriexten=" + descriexten + '}';
    }
}



/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.clases;

import java.sql.Date;

/**
 *
 * @author Rooll
 */
public class OrdenProduccion {
    private long idop;
    private double lote;
    private Date fecha_lanzamiento;
    private Producto producto;
    private boolean estado;
    private String tipo;
    private Date fecha_entrega;

    private void init(){
        this.setProducto(new Producto());
    }
    public OrdenProduccion() {
        this.init();
    }

    public String getTipo() {
        return tipo;
    }
```

```
    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    public OrdenProduccion(long idop, double lote, Date fecha_lanzamiento) {
        this.idop = idop;
        this.lote = lote;
        this.fecha_lanzamiento = fecha_lanzamiento;
        this.init();

    }

    public Date getFecha_entrega() {
        return fecha_entrega;
    }

    public void setFecha_entrega(Date fecha_entrega) {
        this.fecha_entrega = fecha_entrega;
    }

    public long getIdop() {
        return idop;
    }

    public void setIdop(long idop) {
        this.idop = idop;
    }

    public double getLote() {
        return lote;
    }

    public void setLote(double lote) {
        this.lote = lote;
    }

    public Date getFecha_lanzamiento() {
        return fecha_lanzamiento;
    }

    public void setFecha_lanzamiento(Date fecha_lanzamiento) {
        this.fecha_lanzamiento = fecha_lanzamiento;
    }

    public Producto getProducto() {
        return producto;
    }

    public void setProducto(Producto producto) {
        this.producto = producto;
    }

    public boolean isEstado() {
        return estado;
    }

    public void setEstado(boolean estado) {
        this.estado = estado;
    }

    @Override
    public String toString() {
        return String.valueOf(this.idop);
    }


}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.clases;

/**
 *
 * @author Rooll
 */
public class ProdOpeMaq {
    private Producto producto;
    private Maquina maquina;
    private Operacion operacion;
    private boolean estado;
```

```
    private double tiempoconfig;
    private double tiempopieza;
    private double numtrabajador;


    private void init(){
        this.setMaquina(new Maquina());
        this.setProducto(new Producto());
        this.setOperacion(new Operacion());
    }

    public ProdOpeMaq() {
        this.init();
    }

    public ProdOpeMaq(Producto producto, Maquina maquina, Operacion operacion) {
        this.producto = producto;
        this.maquina = maquina;
        this.operacion=operacion;
        this.init();
    }

    public double getTiempoconfig() {
        return tiempoconfig;
    }

    public void setTiempoconfig(double tiempoconfig) {
        this.tiempoconfig = tiempoconfig;
    }

    public double getTiempopieza() {
        return tiempopieza;
    }

    public void setTiempopieza(double tiempopieza) {
        this.tiempopieza = tiempopieza;
    }

    public double getNumtrabajador() {
        return numtrabajador;
    }

    public void setNumtrabajador(double numtrabajador) {
        this.numtrabajador = numtrabajador;
    }

    public Producto getProducto() {
        return producto;
    }

    public void setProducto(Producto producto) {
        this.producto = producto;
    }

    public Maquina getMaquina() {
        return maquina;
    }

    public void setMaquina(Maquina maquina) {
        this.maquina = maquina;
    }

    public boolean isEstado() {
        return estado;
    }

    public void setEstado(boolean estado) {
        this.estado = estado;
    }

    public Operacion getOperacion() {
        return operacion;
    }

    public void setOperacion(Operacion operacion) {
        this.operacion = operacion;
    }

    @Override
    public String toString() {
        return "Ensamble{" + "producto=" + producto + ", maquina=" + maquina + ",operacion="+operacion +'}';
    }


}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.clases;

/**
 *
 * @author Rooll
 */
public class Producto {
     private long id;
    private String descriabrev;
    private String descriexten;
    private TipoProducto tipoproducto;
    private ClaseProducto claseproducto;
    private boolean estado;
    private UnidMed unidmed;


    private void init(){
        this.setTipoproducto(new TipoProducto());
        this.setClaseproducto(new ClaseProducto());
        this.setUnidmed(new UnidMed());
    }
    public Producto() {
       this.init();
    }

    public Producto(long id, String descriabrev, String descriexten) {
        this.id = id;
        this.descriabrev = descriabrev;
        this.descriexten = descriexten;
        this.init();
    }

    public UnidMed getUnidmed() {
        return unidmed;
    }

    public void setUnidmed(UnidMed unidmed) {
        this.unidmed = unidmed;
    }


    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getDescriabrev() {
        return descriabrev;
    }

    public void setDescriabrev(String descriabrev) {
        this.descriabrev = descriabrev;
    }

    public String getDescriexten() {
        return descriexten;
    }

    public void setDescriexten(String descriexten) {
        this.descriexten = descriexten;
    }

    public TipoProducto getTipoproducto() {
        return tipoproducto;
    }

    public void setTipoproducto(TipoProducto tipoproducto) {
        this.tipoproducto = tipoproducto;
    }

    public ClaseProducto getClaseproducto() {
        return claseproducto;
    }

    public void setClaseproducto(ClaseProducto claseproducto) {
        this.claseproducto = claseproducto;
    }
```

```
    public boolean isEstado() {
        return estado;
    }

    public void setEstado(boolean estado) {
        this.estado = estado;
    }

    @Override
    public String toString() {
        return "Producto{" + "id=" + id + ", descriabrev=" + descriabrev + ", descriexten=" + descriexten + '}'+tipoproducto+'}'+claseproducto;
    }
}


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.clases;

/**
 *
 * @author Rooll
 */
public class SeguiProduccion {
    private long op;
    private Producto producto;
    private Producto componente;
    private double cantidad;
    private double avance;
    private double saldo;
    private String Estado;
    private long idpadre;
    private double cantipadre;
    private double stock;
    private double asignado;
    private Maquina maquina;
    private double tiempototal;
    private double tiemposaldo;
    private double nivel;
    private double ultimo;

    private void init(){
        this.setComponente(new Producto());
        this.setProducto(new Producto());
        this.setMaquina(new Maquina());

    }
    public SeguiProduccion() {
        this.init();
    }

    public SeguiProduccion(long op, Producto componente, double cantidad, double avance, String Estado) {
        this.op = op;
        this.componente = componente;
        this.cantidad = cantidad;
        this.avance = avance;
        this.Estado = Estado;
        this.init();
    }

    public Maquina getMaquina() {
        return maquina;
    }

    public void setMaquina(Maquina maquina) {
        this.maquina = maquina;
    }

    public double getTiempototal() {
        return tiempototal;
    }

    public void setTiempototal(double tiempototal) {
        this.tiempototal = tiempototal;
    }

    public double getTiemposaldo() {
        return tiemposaldo;
    }

    public void setTiemposaldo(double tiemposaldo) {
        this.tiemposaldo = tiemposaldo;
```

```
}

public double getNivel() {
    return nivel;
}

public void setNivel(double nivel) {
    this.nivel = nivel;
}

public double getUltimo() {
    return ultimo;
}

public void setUltimo(double ultimo) {
    this.ultimo = ultimo;
}

public double getCantipadre() {
    return cantipadre;
}

public void setCantipadre(double cantipadre) {
    this.cantipadre = cantipadre;
}


public double getStock() {
    return stock;
}

public void setStock(double stock) {
    this.stock = stock;
}

public double getAsignado() {
    return asignado;
}

public void setAsignado(double asignado) {
    this.asignado = asignado;
}



public long getIdpadre() {
    return idpadre;
}

public void setIdpadre(long idpadre) {
    this.idpadre = idpadre;
}


public Producto getProducto() {
    return producto;
}

public void setProducto(Producto producto) {
    this.producto = producto;
}


public double getsaldo() {
    return saldo;
}

public void setSaldo(double saldo) {
    this.saldo = saldo;
}



public long getOp() {
    return op;
}

public void setOp(long op) {
    this.op = op;
}

public Producto getComponente() {
    return componente;
}

public void setComponente(Producto componente) {
```

```
        this.componente = componente;
    }

    public double getCantidad() {
        return cantidad;
    }

    public void setCantidad(double cantidad) {
        this.cantidad = cantidad;
    }

    public double getAvance() {
        return avance;
    }

    public void setAvance(double avance) {
        this.avance = avance;
    }

    public String getEstado() {
        return Estado;
    }

    public void setEstado(String Estado) {
        this.Estado = Estado;
    }

    @Override
    public String toString() {
        return "SeguiProduccion{" + "op=" + op + ", componente=" + componente + ", cantidad=" + cantidad + ", avance=" + avance + ", Estado=" +
Estado + '}';
    }


}
```

9.3 ANEXO C Codificación Conexión con la Base de datos:

```
    /*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.conexion;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

/**
 *
 * @author Rooll
 */
public class Conexion {
    public static Connection conectar(){
        Connection cn= null;
            String url="jdbc:oracle:thin:@192.168.1.61:1521:XE";
            String driver="oracle.jdbc.OracleDriver";
        try {
            Class.forName(driver);
            try {
                cn=DriverManager.getConnection(url, "C##PRODUCCION", "123456");

            } catch (SQLException e) {
                System.err.println("Error de Conexion "+e.getMessage());
            }

        } catch (Exception e) {
            System.err.println("Error de Driver "+e.getMessage());
        }

       return cn;
    }
}
```

9.4 ANEXO D Codificacion Interfaces y sus implementaciones

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.dao;

import java.util.List;
import pe.produccion.clases.Producto;

/**
 *
 * @author Rooll
 * @param <T>
 */
public interface BDInterf<T> {

  List<T> mostrar ();
  List<T> mostrarCriterio(String nombre);
  T mostrarId (T o);
  boolean insertar (T o);
  boolean actualizar (T o);
  boolean eliminar (T o);
  T encuentraID(String nombre);
  boolean validar(String nombre);
  boolean eliminar ();
  List<T> mostrarCriteriolike(String nombre);

}


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import pe.produccion.clases.ConsumoLM;
import pe.produccion.clases.Producto;
import pe.produccion.conexion.Conexion;

/**
 *
 * @author Rooll
 */
public class ConsumoLMImple implements BDInterf<ConsumoLM>{

   @Override
   public List<ConsumoLM> mostrar() {
      List<ConsumoLM> consumoLms = new ArrayList<>();
      Connection cn=null;
      try {
         cn= Conexion.conectar();
         if (cn!=null) {
            String sql= "SELECT B.DESCRIABREV PRODUCTO, C.DESCRIABREV COMPONENTE, D.DESCRIABREV PADRE,"
               + " A.CANTIDAD CANTIDAD"
               + " FROM CONSUMOLM A"
               + " INNER JOIN PRODUCTO C ON(C.ID = A.IDCOMPONENTE)"
                + " INNER JOIN PRODUCTO B ON(B.ID = A.IDPRODUCTO)"
                +  " INNER JOIN PRODUCTO D ON(D.ID = A.IDPADRE)" ;

         Statement st = cn.createStatement();
         ResultSet rs=st.executeQuery(sql);
         while (rs.next()) {
            ConsumoLM consumoLM = new ConsumoLM();
            Producto producto= new Producto();
            Producto componente= new Producto();
            Producto padre= new Producto();
            producto.setDescriabrev(rs.getString("PRODUCTO"));
            componente.setDescriabrev(rs.getString("COMPONENTE"));
            padre.setDescriabrev(rs.getString("PADRE"));
            consumoLM.setProducto(producto);
            consumoLM.setComponente(componente);
            consumoLM.setPadre(padre);
```

```
            consumoLM.setCantidad(rs.getDouble("CANTIDAD"));
            consumoLms.add(consumoLM);

        }
        }


    } catch (SQLException e) {
        System.err.println("Error de conexion mostrar "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {
                System.err.println("Error de cerrado mostrar"+e.getMessage());
            }
        }
    }

    return consumoLms;
}

@Override
public List<ConsumoLM> mostrarCriterio(String nombre) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
}

@Override
public ConsumoLM mostrarId(ConsumoLM o) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
}

@Override
public boolean insertar(ConsumoLM o) {
  boolean sw;
    Connection cn=null;
    try {
        cn= Conexion.conectar();
        if (cn!=null) {
            String sql= "INSERT INTO CONSUMOLM VALUES"
                    + "(?,?,?,?)";
        PreparedStatement pstm = cn.prepareStatement(sql);
        pstm.setDouble(1, o.getProducto().getId());
        pstm.setDouble(2, o.getComponente().getId());
        pstm.setDouble(3, o.getCantidad());
        pstm.setDouble(4, o.getPadre().getId());
       pstm.executeUpdate();

        }
     sw=true;

    } catch (SQLException e) {
        System.err.println("Error de conexion Insertar "+e.getMessage());
        sw=false;
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {
                System.err.println("Error de cerrado insertar "+e.getMessage());
            }
        }
    }

    return sw;

}

@Override
public boolean actualizar(ConsumoLM o) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
}



@Override
public ConsumoLM encuentraID(String nombre) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
}

@Override
public boolean validar(String nombre) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
}

@Override
```

```
    public boolean eliminar() {
         boolean sw;
        Connection cn=null;
        try {
            cn= Conexion.conectar();
            if (cn!=null) {
                String sql= "delete from CONSUMOLM";
            Statement stm = cn.createStatement();

            stm.executeUpdate(sql);

             }
           sw=true;

        } catch (SQLException e) {
            System.err.println("Error de conexion eliminar "+e.getMessage());
            sw=false;
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                } catch (Exception e) {
                    System.err.println("Error de cerrado eliminar"+e.getMessage());
                }
            }
        }

        return sw;
    }

    @Override
    public boolean eliminar(ConsumoLM o) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public List<ConsumoLM> mostrarCriteriolike(String nombre) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.dao;

import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.Vector;
//import java.util.Date;
import pe.produccion.clases.ClaseProducto;
import pe.produccion.clases.GanttFinal;
import pe.produccion.clases.Maquina;
import pe.produccion.clases.Producto;
import pe.produccion.clases.TipoProducto;
import pe.produccion.conexion.Conexion;

/**
 *
 * @author Rooll
 */
public class GanttImpleme {

    public List<GanttFinal> mostrar(){

        List<GanttFinal> lista = new ArrayList<GanttFinal>();
        Connection cn= null;
        try {
            cn= Conexion.conectar();
            if (cn!=null) {
                String sql = "SELECT A.IDPROGRAMA ID, B.DESCRIABREV MAQUINA, A.OP OP,"
                        +" C.DESCRIABREV COMPONENTE, D.DESCRIABREV PADRE, A.TINICIO TINICIO, A.TFINAL TFINAL"
                        + " FROM GANTTFINAL A "
                        + "INNER JOIN MAQUINA B ON (B.ID = A.MAQ)"
                        + "INNER JOIN PRODUCTO C ON (C.ID = A.COMP)"
```

```
                + "LEFT JOIN PRODUCTO D ON (D.ID = A.PADRE)"
                +" ORDER BY A.IDPROGRAMA, B.DESCRIABREV, A.TINICIO";


            Statement st = cn.createStatement();
            ResultSet rs = st.executeQuery(sql);
            while (rs.next()) {
             GanttFinal gfinal= new GanttFinal();

                gfinal.setIdprograma(rs.getLong("ID"));
                Maquina maq = new Maquina();
                maq.setDescriabrev(rs.getString("MAQUINA"));
                gfinal.setMaq(maq);
                gfinal.setOp(rs.getLong("OP"));
                Producto comp = new Producto();
                comp.setDescriabrev(rs.getString("COMPONENTE"));
                gfinal.setComp(comp);

                Producto padre = new Producto();
                    padre.setDescriabrev(rs.getString("PADRE"));
                gfinal.setPadre(padre);
                gfinal.setTinicio(rs.getDouble("TINICIO"));
                gfinal.setTfinal(rs.getDouble("TFINAL"));
                lista.add(gfinal);
            }

        }

    } catch (SQLException e) {
        System.err.println("Error de conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {
            }
        }
    }

    return lista;
}




public boolean insertar(double[][] ga) {

    boolean sw;
    Connection cn= null;

    try {
        cn= Conexion.conectar();

        if (cn!=null) {
            long max=0;
            String sqlmax="SELECT MAX(IDPROGRAMA) max from PROGRAMA";
            Statement st = cn.createStatement();
            ResultSet rs = st.executeQuery(sqlmax);
            while (rs.next()) {
                max=rs.getLong("max")+1;
            }

            String sql0= "INSERT INTO PROGRAMA VALUES (?,?)";
            PreparedStatement pstm= cn.prepareStatement(sql0);
            pstm.setLong(1, max);
            java.util.Date date = new java.util.Date();

            java.text.SimpleDateFormat sdf= new java.text.SimpleDateFormat("yyyy-MM-dd");
            String fecha= sdf.format(date);
            Date ndate = java.sql.Date.valueOf(fecha);
            //pstm.setString(2, fecha);
            pstm.setDate(2, ndate);
            pstm.executeUpdate();

            for (int i = 0; i < ga.length; i++) {
                if (ga[i][0]==0) {
                    break;
                } else {
                    String sql= "INSERT INTO GANTTFINAL VALUES (?,?,?,?,?,?,?)";
            PreparedStatement pstm1= cn.prepareStatement(sql);
            pstm1.setLong(1, max);
            pstm1.setDouble(2, ga[i][0]);
```

```
          pstm1.setDouble(3, ga[i][1]);
          pstm1.setDouble(4, ga[i][2]);
          pstm1.setDouble(5, ga[i][3]);
          pstm1.setDouble(6, ga[i][4]);
          pstm1.setDouble(7, ga[i][5]);

         pstm1.executeUpdate();
               }

           }

       }
     sw=true;
    } catch (SQLException e) {
       sw=false;
       System.err.println("Error en la conexion "+e.getMessage());
    }finally{
       if (cn!=null) {
          try {
             cn.close();
          } catch (Exception e) {

          }
       }
    }

  return sw;
}




public boolean eliminar(double o) {
     boolean sw;
    Connection cn= null;

    try {
       cn= Conexion.conectar();

       if (cn!=null) {

          String sql= "DELETE PROGRAMA WHERE IDPROGRAMA=?";

          PreparedStatement pstm= cn.prepareStatement(sql);
          pstm.setDouble(1, o);
          pstm.executeUpdate();

          String sql2= "DELETE GANTTFINAL WHERE IDPROGRAMA=?";

          PreparedStatement pstm2= cn.prepareStatement(sql2);
          pstm2.setDouble(1, o);
          pstm2.executeUpdate();
         }
     sw=true;
    } catch (SQLException e) {
       sw=false;
       System.err.println("Error en la conexion "+e.getMessage());
    }finally{
       if (cn!=null) {
          try {
             cn.close();
          } catch (Exception e) {

          }
       }
    }

  return sw;
}

public double max() {

    double maximo =0;
    Connection cn= null;

    try {
       cn= Conexion.conectar();

       if (cn!=null) {
          long max=0;
          String sqlmax="SELECT MAX(IDPROGRAMA) max from PROGRAMA";
          Statement st = cn.createStatement();
          ResultSet rs = st.executeQuery(sqlmax);
          while (rs.next()) {
             maximo=rs.getLong("max");
```

```
            }


              }



    } catch (SQLException e) {

       System.err.println("Error en la conexion "+e.getMessage());
    }finally{
       if (cn!=null) {
          try {
             cn.close();
          } catch (Exception e) {

          }
       }
    }

   return maximo;
  }

   public List<GanttFinal> mostrarultimo(double maxim){

    List<GanttFinal> lista = new ArrayList<GanttFinal>();
    Connection cn= null;
    try {
       cn= Conexion.conectar();
       if (cn!=null) {
          String sql = "SELECT A.IDPROGRAMA ID, B.DESCRIABREV MAQUINA, A.OP OP,"
                +" C.DESCRIABREV COMPONENTE, D.DESCRIABREV PADRE, A.TINICIO TINICIO, A.TFINAL TFINAL"
                + " FROM GANTTFINAL A "
                + "INNER JOIN MAQUINA B ON (B.ID = A.MAQ)"
                + "INNER JOIN PRODUCTO C ON (C.ID = A.COMP)"
                + "LEFT JOIN PRODUCTO D ON (D.ID = A.PADRE)"
                +" WHERE IDPROGRAMA=?"
                +" ORDER BY A.IDPROGRAMA, B.DESCRIABREV, A.TINICIO";


          PreparedStatement pstm = cn.prepareStatement(sql);
          pstm.setDouble(1, maxim);
          ResultSet rs = pstm.executeQuery();
          while (rs.next()) {
           GanttFinal gfinal= new GanttFinal();

             gfinal.setIdprograma(rs.getLong("ID"));
             Maquina maq = new Maquina();
             maq.setDescriabrev(rs.getString("MAQUINA"));
             gfinal.setMaq(maq);
             gfinal.setOp(rs.getLong("OP"));
             Producto comp = new Producto();
             comp.setDescriabrev(rs.getString("COMPONENTE"));
             gfinal.setComp(comp);

             Producto padre = new Producto();
                padre.setDescriabrev(rs.getString("PADRE"));
             gfinal.setPadre(padre);
             gfinal.setTinicio(rs.getDouble("TINICIO"));
             gfinal.setTfinal(rs.getDouble("TFINAL"));
             lista.add(gfinal);
          }

       }

    } catch (SQLException e) {
       System.err.println("Error de conexion "+e.getMessage());
    }finally{
       if (cn!=null) {
          try {
             cn.close();
          } catch (Exception e) {
          }
       }
    }

     return lista;
  }

  }


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
package pe.produccion.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import pe.produccion.clases.ListaMateriales;
import pe.produccion.clases.Maquina;
import pe.produccion.clases.Operacion;


import pe.produccion.clases.Producto;

import pe.produccion.conexion.Conexion;

/**
 *
 * @author Rooll
 */
public class ListaMaterImple implements BDInterf<ListaMateriales>{

    @Override
    public List<ListaMateriales> mostrar() {
        List<ListaMateriales> listaMaterialess = new ArrayList<ListaMateriales>();
        Connection cn= null;
        try {
            cn= Conexion.conectar();
            if (cn!=null) {
                String sql = "SELECT A.DESCRIABREV COMPONENTE, B.CANTIDAD_COMP CANTI,"
                        +" C.DESCRIABREV PRODUCTOPADRE "
                        +" FROM LISTAMATERIALES B"
                       +" INNER JOIN PRODUCTO A ON (A.ID=B.IDCOMPONENTE)"
                      +" INNER JOIN PRODUCTO C ON (C.ID=B.IDPRODUCTO)"
                     +" WHERE B.ESTADO='1'";

                Statement st = cn.createStatement();
                ResultSet rs = st.executeQuery(sql);
                while (rs.next()) {
                    ListaMateriales listaMateriales = new ListaMateriales();
                    Producto producto = new Producto();
                    producto.setDescriabrev(rs.getString("COMPONENTE"));
                    listaMateriales.setComponente(producto);
                    listaMateriales.setCantidad_comp(rs.getDouble("CANTI"));

                    Producto producto2= new Producto();
                    producto2.setDescriabrev(rs.getString("PRODUCTOPADRE"));
                    listaMateriales.setProducto(producto2);
                    listaMaterialess.add(listaMateriales);
                }

            }

        } catch (SQLException e) {
            System.err.println("Error de conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                } catch (Exception e) {
                }
            }
        }

         return listaMaterialess;
    }




    @Override
    public boolean insertar(ListaMateriales o) {

        boolean sw=false;
        Connection cn= null;
        long count=0;
        try {
            cn= Conexion.conectar();

            if (cn!=null) {
                String sqlo="select COUNT(IDCOMPONENTE || IDPRODUCTO) AS COMBINACION FROM LISTAMATERIALES"
```

```
                    +" WHERE IDCOMPONENTE=? AND IDPRODUCTO=? AND ESTADO='1'";
            PreparedStatement pstm = cn.prepareStatement(sqlo);
            pstm.setLong(1, o.getComponente().getId());
            pstm.setLong(2, o.getProducto().getId());
            ResultSet rs = pstm.executeQuery();
            while (rs.next()) {
              count=rs.getLong("COMBINACION");
            }
            if (count<1) {
                String sql= "INSERT INTO LISTAMATERIALES VALUES (?,?,?,'1')";
            PreparedStatement pstm2= cn.prepareStatement(sql);
            pstm2.setDouble(1, o.getComponente().getId());
            pstm2.setDouble(2, o.getCantidad_comp());

            pstm2.setDouble(3,o.getProducto().getId());



            pstm2.executeUpdate();
             sw=true;
             } else {
                sw=false;
                System.out.println("Esa combinacion de materiales ya existe");
             }


        }

    } catch (SQLException e) {
        sw=false;
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {

            }
        }
    }

  return sw;
}

@Override
public boolean actualizar(ListaMateriales o) {
     boolean sw;
    Connection cn= null;

    try {
        cn= Conexion.conectar();

        if (cn!=null) {

            String sql= "UPDATE LISTAMATERIALES SET CANTIDAD_COMP=?"

                  + " WHERE IDCOMPONENTE=? AND IDPRODUCTO=? and ESTADO='1'";

            PreparedStatement pstm= cn.prepareStatement(sql);
            pstm.setDouble(1, o.getCantidad_comp());

            pstm.setDouble(2, o.getComponente().getId());
            pstm.setDouble(3, o.getProducto().getId());

            pstm.executeUpdate();

        }
      sw=true;
    } catch (SQLException e) {
        sw=false;
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {

            }
        }
    }

  return sw;
}

@Override
public boolean eliminar(ListaMateriales o) {
```

```
    boolean sw;
   Connection cn= null;

   try {
      cn= Conexion.conectar();

      if (cn!=null) {

         String sql= "UPDATE LISTAMATERIALES SET ESTADO='0'"
                  +" WHERE IDCOMPONENTE=? AND IDPRODUCTO=? and ESTADO='1'";
         PreparedStatement pstm= cn.prepareStatement(sql);
          pstm.setDouble(1, o.getComponente().getId());
         pstm.setDouble(2, o.getProducto().getId());

         pstm.executeUpdate();

      }
     sw=true;
   } catch (SQLException e) {
      sw=false;
      System.err.println("Error en la conexion "+e.getMessage());
   }finally{
      if (cn!=null) {
         try {
            cn.close();
         } catch (Exception e) {

         }
      }
   }

 return sw;
}
```

```
@Override
public List<ListaMateriales> mostrarCriterio(String nombre) {
    List<ListaMateriales> listaMaterialess = new ArrayList<ListaMateriales>();
   Connection cn= null;
   try {
      cn= Conexion.conectar();
      if (cn!=null) {
         String sql = "SELECT A.DESCRIABREV COMPONENTE, B.CANTIDAD_COMP CANTI,"
                  +" C.DESCRIABREV PRODUCTOPADRE "
                  +" FROM LISTAMATERIALES B"
                 +" INNER JOIN PRODUCTO A ON (A.ID=B.IDCOMPONENTE)"
                +" INNER JOIN PRODUCTO C ON (C.ID=B.IDPRODUCTO)"
               +" WHERE UPPER(C.DESCRIABREV) = UPPER(?) AND B.ESTADO='1'";

         PreparedStatement pstm = cn.prepareStatement(sql);
         pstm.setString(1, nombre);
         ResultSet rs = pstm.executeQuery();
         while (rs.next()) {
            ListaMateriales listaMateriales = new ListaMateriales();
            Producto producto = new Producto();
            producto.setDescriabrev(rs.getString("COMPONENTE"));
            listaMateriales.setComponente(producto);
            listaMateriales.setCantidad_comp(rs.getDouble("CANTI"));

            Producto producto2= new Producto();
            producto2.setDescriabrev(rs.getString("PRODUCTOPADRE"));
            listaMateriales.setProducto(producto2);
            listaMaterialess.add(listaMateriales);
         }

      }

   } catch (SQLException e) {
      System.err.println("Error de conexion "+e.getMessage());
   }finally{
      if (cn!=null) {
         try {
            cn.close();
         } catch (Exception e) {
         }
      }
   }

    return listaMaterialess;
```

```
    }

    @Override
 public List<ListaMateriales> mostrarCriteriolike(String nombre) {
        List<ListaMateriales> listaMaterialess = new ArrayList<ListaMateriales>();
       Connection cn= null;
       try {
          cn= Conexion.conectar();
          if (cn!=null) {
             String sql = "SELECT A.DESCRIABREV COMPONENTE, B.CANTIDAD_COMP CANTI,"
                     +" C.DESCRIABREV PRODUCTOPADRE "
                     +" FROM LISTAMATERIALES B"
                    +" INNER JOIN PRODUCTO A ON (A.ID=B.IDCOMPONENTE)"
                   +" INNER JOIN PRODUCTO C ON (C.ID=B.IDPRODUCTO)"
                  +" WHERE UPPER(C.DESCRIABREV) LIKE UPPER(?) AND B.ESTADO='1'";

             PreparedStatement pstm = cn.prepareStatement(sql);
             pstm.setString(1, "%"+nombre+"%");
             ResultSet rs = pstm.executeQuery();
             while (rs.next()) {
                ListaMateriales listaMateriales = new ListaMateriales();
                Producto producto = new Producto();
                producto.setDescriabrev(rs.getString("COMPONENTE"));
                listaMateriales.setComponente(producto);
                listaMateriales.setCantidad_comp(rs.getDouble("CANTI"));

                Producto producto2= new Producto();
                producto2.setDescriabrev(rs.getString("PRODUCTOPADRE"));
                listaMateriales.setProducto(producto2);
                listaMaterialess.add(listaMateriales);
             }

          }

       } catch (SQLException e) {
          System.err.println("Error de conexion "+e.getMessage());
       }finally{
          if (cn!=null) {
             try {
                cn.close();
             } catch (Exception e) {
             }
          }
       }

        return listaMaterialess;
    }
    @Override
    public ListaMateriales mostrarId(ListaMateriales o) {
       throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public ListaMateriales encuentraID(String nombre) {
       throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public boolean validar(String nombre) {
       throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public boolean eliminar() {
       throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }



}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
```

```
import pe.produccion.clases.Maquina;


import pe.produccion.conexion.Conexion;

/**
 *
 * @author Rooll
 */
public class MaquinaImple implements BDInterf<Maquina>{

    @Override
    public List<Maquina> mostrar() {
        List<Maquina> maquinas = new ArrayList<Maquina>();
        Connection cn= null;
        try {
            cn= Conexion.conectar();
            if (cn!=null) {
                String sql = "SELECT A.ID ID, A.DESCRIABREV MAQUINA,"
                        +" A.DESCRIEXTEN EXTEN FROM MAQUINA A"
                        + " WHERE A.ESTADO='1' ORDER BY A.ID";

                Statement st = cn.createStatement();
                ResultSet rs = st.executeQuery(sql);
                while (rs.next()) {
                    Maquina maquina = new Maquina();

                    maquina.setId(rs.getLong("ID"));
                    maquina.setDescriabrev(rs.getString("MAQUINA"));
                    maquina.setDescriexten(rs.getString("EXTEN"));
                    maquinas.add(maquina);
                }

            }

        } catch (SQLException e) {
            System.err.println("Error de conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                } catch (Exception e) {
                }
            }
        }

         return maquinas ;
    }

    @Override
    public List<Maquina> mostrarCriterio(String nombre) {
        List<Maquina> maquinas = new ArrayList<Maquina>();
        Connection cn= null;

        try {
            cn= Conexion.conectar();

            if (cn!=null) {

                 String sql="SELECT A.ID ID, A.DESCRIABREV ABREV, A.DESCRIEXTEN EXTEN"
                        +" FROM MAQUINA A"
                        +" WHERE (UPPER(A.DESCRIABREV) LIKE UPPER(?) OR UPPER(A.DESCRIEXTEN) LIKE UPPER(?))"
                         + " AND A.ESTADO='1' ORDER BY A.ID";
                PreparedStatement pstm= cn.prepareStatement(sql);
                pstm.setString(1, "%"+nombre+"%");
                pstm.setString(2, "%"+nombre+"%");
                ResultSet rs= pstm.executeQuery();

                while (rs.next()) {
                    Maquina maquina = new Maquina();
                     maquina.setId(rs.getLong("ID"));
                    maquina.setDescriabrev(rs.getString("ABREV"));
                    maquina.setDescriexten(rs.getString("EXTEN"));

                    maquinas.add(maquina);
                }

            }

        } catch (SQLException e) {
            System.err.println("Error en la conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
```

```
            } catch (Exception e) {
            }
        }
    }

  return maquinas;
}

@Override
public Maquina mostrarId(Maquina o) {
    Maquina maquina = new Maquina();
    Connection cn= null;

    try {
        cn= Conexion.conectar();

        if (cn!=null) {

            String sql="SELECT A.ID ID, A.DESCRIABREV ABREV, A.DESCRIEXTEN EXTEN"
                    +" FROM MAQUINA A"
                    +" WHERE A.ID=? AND A.ESTADO='1'";
            PreparedStatement pstm= cn.prepareStatement(sql);
            pstm.setLong(1, o.getId());

            ResultSet rs= pstm.executeQuery();

            while (rs.next()) {


                maquina.setId(rs.getLong("ID"));
                maquina.setDescriabrev(rs.getString("ABREV"));
                maquina.setDescriexten(rs.getString("EXTEN"));


            }

        }

    } catch (SQLException e) {
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {
            }
        }
    }

  return maquina;
}

@Override
public boolean insertar(Maquina o) {

    boolean sw;
    Connection cn= null;

    try {
        cn= Conexion.conectar();

        if (cn!=null) {
            String sqlmax="SELECT MAX(ID) max from MAQUINA";
            Statement st = cn.createStatement();
            ResultSet rs = st.executeQuery(sqlmax);
            while (rs.next()) {
                o.setId(rs.getLong("max")+1);
            }

            String sql= "INSERT INTO MAQUINA VALUES (?,?,?,'1')";
            PreparedStatement pstm= cn.prepareStatement(sql);
            pstm.setLong(1, o.getId());
            pstm.setString(2, o.getDescriabrev());
            pstm.setString(3, o.getDescriexten());

            pstm.executeUpdate();

        }
      sw=true;
    } catch (SQLException e) {
        sw=false;
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
```

```
            } catch (Exception e) {

            }
        }
    }

  return sw;
}

@Override
public boolean actualizar(Maquina o) {
     boolean sw;
    Connection cn= null;

    try {
        cn= Conexion.conectar();

        if (cn!=null) {

          String sql= "UPDATE MAQUINA SET DESCRIABREV=?, DESCRIEXTEN=?"
                +" WHERE ID=?";

          PreparedStatement pstm= cn.prepareStatement(sql);
          pstm.setLong(3, o.getId());
          pstm.setString(1, o.getDescriabrev());
          pstm.setString(2, o.getDescriexten());

          pstm.executeUpdate();

        }
      sw=true;
    } catch (SQLException e) {
        sw=false;
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {

            }
        }
    }

  return sw;
}

@Override
public boolean eliminar(Maquina o) {
     boolean sw;
    Connection cn= null;

    try {
        cn= Conexion.conectar();

        if (cn!=null) {

          String sql= "UPDATE MAQUINA SET ESTADO='0' WHERE ID=?";

          PreparedStatement pstm= cn.prepareStatement(sql);
          pstm.setLong(1, o.getId());
          pstm.executeUpdate();
          }
      sw=true;
    } catch (SQLException e) {
        sw=false;
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {

            }
        }
    }

  return sw;
}



@Override
public Maquina encuentraID(String nombre) {
```

```
 Maquina maquina = new Maquina();
   Connection cn= null;

   try {
      cn= Conexion.conectar();

      if (cn!=null) {

          String sql="SELECT A.ID FROM MAQUINA A"
            +" WHERE(upper(A.DESCRIABREV)=upper(?) OR upper(A.DESCRIEXTEN)=upper(?))"
                + " AND A.ESTADO='1'";
          PreparedStatement pstm= cn.prepareStatement(sql);
          pstm.setString(1, nombre);
          pstm.setString(2, nombre);
          ResultSet rs= pstm.executeQuery();

          while (rs.next()) {

             maquina.setId(rs.getLong("ID"));

          }

      }

   } catch (SQLException e) {
      System.err.println("Error en la conexion "+e.getMessage());
   }finally{
      if (cn!=null) {
         try {
            cn.close();
         } catch (Exception e) {
         }
      }
   }

  return maquina;
}

@Override
public boolean validar(String nombre) {
   boolean sw = true;
    Connection cn= null;

   try {
      cn= Conexion.conectar();

      if (cn!=null) {

          String sql="SELECT A.ID FROM MAQUINA A"
                +" WHERE (upper(A.DESCRIABREV) =upper(trim(?)) OR"
                 +" upper(A.DESCRIEXTEN) =upper(trim(?)))"
                 + " AND A.ESTADO='1'";
          PreparedStatement pstm= cn.prepareStatement(sql);
          pstm.setString(1, nombre);
          pstm.setString(2, nombre);
          ResultSet rs= pstm.executeQuery();

          while (rs.next()) {

             if (rs.getLong("ID")>0) {
                sw=false;
             }

          }

      }

   } catch (SQLException e) {
      System.err.println("Error en la conexion "+e.getMessage());
   }finally{
      if (cn!=null) {
         try {
            cn.close();
         } catch (Exception e) {
         }
      }
   }

  return sw;
}

@Override
public boolean eliminar() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
}
```

```
    @Override
    public List<Maquina> mostrarCriteriolike(String nombre) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }



}


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.dao;

import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import pe.produccion.clases.Caracteristica;
import pe.produccion.clases.ConsumoLM;
import pe.produccion.clases.Dimension;
import pe.produccion.clases.ListaMateriales;
import pe.produccion.clases.Maquina;
import pe.produccion.clases.Operacion;
import pe.produccion.clases.OrdenProduccion;
import pe.produccion.clases.ProduCarac;
import pe.produccion.clases.ProduDimen;


import pe.produccion.clases.Producto;

import pe.produccion.conexion.Conexion;
import pe.produccion.repor.ReporteLista;

/**
 *
 * @author Rooll
 */
public class OrdenProdImple implements BDInterf<OrdenProduccion>{

    @Override
    public List<OrdenProduccion> mostrar() {
        List<OrdenProduccion> ordenProduccions = new ArrayList<OrdenProduccion>();
        Connection cn= null;
        try {
            cn= Conexion.conectar();
            if (cn!=null) {
                String sql = "SELECT A.IDOP OP, B.DESCRIABREV PRODUCTO,"
                        +" A.LOTE LOTE, A.FECHA_LANZAMIENTO FECHA, A.TIPO TIPO, A.FECHA_ENTREGA FECHAE "
                        +" FROM ORDENPRODUCCION A"
                       +" INNER JOIN PRODUCTO B ON (B.ID=A.IDPRODUCTO)"
                        +" WHERE A.ESTADO='1' ORDER BY A.IDOP";

                Statement st = cn.createStatement();
                ResultSet rs = st.executeQuery(sql);
                while (rs.next()) {
                    OrdenProduccion ordenProduccion = new OrdenProduccion();
                    Producto producto = new Producto();
                    producto.setDescriabrev(rs.getString("PRODUCTO"));
                    ordenProduccion.setIdop(rs.getLong("OP"));
                   ordenProduccion.setProducto(producto);
                  ordenProduccion.setLote(rs.getDouble("LOTE"));
                  ordenProduccion.setFecha_lanzamiento(rs.getDate("FECHA"));
                   ordenProduccion.setTipo(rs.getString("TIPO"));
                    ordenProduccion.setFecha_entrega(rs.getDate("FECHAE"));
                    ordenProduccions.add(ordenProduccion);
                }

            }

        } catch (SQLException e) {
            System.err.println("Error de conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                } catch (Exception e) {
                }
            }
```

```
    }

    return ordenProduccions;
}



@Override
public boolean insertar(OrdenProduccion o) {

    boolean sw=false;
    Connection cn= null;
    long count=0;
    try {
        cn= Conexion.conectar();

        if (cn!=null) {
            String sqlo="select MAX(IDOP) max FROM ORDENPRODUCCION";
            Statement st = cn.createStatement();
            ResultSet rs = st.executeQuery(sqlo);
            while (rs.next()) {
              o.setIdop(rs.getLong("max")+1);
            }

               String sql= "INSERT INTO ORDENPRODUCCION VALUES (?,?,?,?,'1',?,?)";
           PreparedStatement pstm2= cn.prepareStatement(sql);
           pstm2.setDouble(1, o.getIdop());
           pstm2.setDouble(2, o.getProducto().getId());
           pstm2.setDouble(3, o.getLote());
           pstm2.setDate(4, o.getFecha_lanzamiento());
           pstm2.setString(5, o.getTipo());
           pstm2.setDate(6, o.getFecha_entrega());

           pstm2.executeUpdate();
            sw=true;
        //   BDInterf<ConsumoLM> bdconsumo = new ConsumoLMImple();
    //    bdconsumo.eliminar();
        BDInterf<Producto> bdprod = new ProductoImple();
        Producto producto = bdprod.mostrarId(o.getProducto());
        String prod=producto.getDescriabrev();
         ReporteLista repor = new ReporteLista();
         String[][]lis= repor.consumocantidad(prod, o.getLote());
        SeguiProduccionImple segui = new SeguiProduccionImple();
        double cant=0;
        String comp="";
        String padr="";
        double ult = 0;
        double cantipadre=0;
           for (int i = 0; i < 10000; i++) {
               if (lis[i][0] == null) {
                   break;
               }
             comp= lis[i][0];
              cant= Double.valueOf(lis[i][1]);
              padr=lis[i][2];
              cantipadre=Double.valueOf(lis[i][3]);
              ult=Double.valueOf(lis[i][4]);
               if (segui.insertOP(o, comp,cant,padr, cantipadre, ult)) {

                     System.out.println("Exito");
                  }else{
                     System.out.println("Error");
                  }

           }

        }

    } catch (SQLException e) {
        sw=false;
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {

            }
        }
    }

  return sw;
}
```

```
@Override
public boolean actualizar(OrdenProduccion o) {
    boolean sw;
   Connection cn= null;

   try {
      cn= Conexion.conectar();

      if (cn!=null) {

        String sql= "UPDATE ORDENPRODUCCION SET IDPRODUCTO=?, LOTE=?, "
              +"FECHA_LANZAMIENTO=?, TIPO=?, FECHA_ENTREGA=?"
               + " WHERE IDOP=? and ESTADO='1'";

        PreparedStatement pstm= cn.prepareStatement(sql);
        pstm.setDouble(1, o.getProducto().getId());
        pstm.setDouble(2, o.getLote());
        pstm.setDate(3, o.getFecha_lanzamiento());
         pstm.setString(4,o.getTipo());
         pstm.setDouble(6, o.getIdop());
         pstm.setDate(5, o.getFecha_entrega());
        pstm.executeUpdate();

      }
     sw=true;
   } catch (SQLException e) {
      sw=false;
      System.err.println("Error en la conexion "+e.getMessage());
   }finally{
      if (cn!=null) {
         try {
            cn.close();
         } catch (Exception e) {

         }
      }
   }

 return sw;
}

@Override
public boolean eliminar(OrdenProduccion o) {
     boolean sw;
   Connection cn= null;

   try {
      cn= Conexion.conectar();

      if (cn!=null) {

        String sql= "UPDATE ORDENPRODUCCION SET ESTADO='0'"
                +" WHERE IDOP=? and ESTADO='1'";
        PreparedStatement pstm= cn.prepareStatement(sql);
        pstm.setDouble(1, o.getIdop());


        pstm.executeUpdate();

      }
     sw=true;
   } catch (SQLException e) {
      sw=false;
      System.err.println("Error en la conexion "+e.getMessage());
   }finally{
      if (cn!=null) {
         try {
            cn.close();
         } catch (Exception e) {

         }
      }
   }

 return sw;
}
```

```
@Override
```

```
    public List<OrdenProduccion> mostrarCriterio(String nombre) {
        List<OrdenProduccion> ordenProduccions = new ArrayList<OrdenProduccion>();
        Connection cn= null;
        try {
            cn= Conexion.conectar();
            if (cn!=null) {
                String sql = "SELECT A.IDOP OP, B.DESCRIABREV PRODUCTO,"
                        +" A.LOTE LOTE, A.FECHA_LANZAMIENTO FECHA, A.TIPO TIPO, A.FECHA_ENTREGA FECHAE"
                        +" FROM ORDENPRODUCCION A"
                       +" INNER JOIN PRODUCTO B ON (B.ID=A.IDPRODUCTO)"
                        +" WHERE (upper(B.DESCRIABREV) LIKE UPPER(?) and A.ESTADO='1') ORDER BY A.IDOP";

                PreparedStatement pstm = cn.prepareStatement(sql);
                pstm.setString(1, "%"+nombre+"%");
                ResultSet rs = pstm.executeQuery();
                while (rs.next()) {
                    OrdenProduccion ordenProduccion = new OrdenProduccion();
                    Producto producto = new Producto();
                    producto.setDescriabrev(rs.getString("PRODUCTO"));
                    ordenProduccion.setIdop(rs.getLong("OP"));
                   ordenProduccion.setProducto(producto);
                  ordenProduccion.setLote(rs.getDouble("LOTE"));
                  ordenProduccion.setFecha_lanzamiento(rs.getDate("FECHA"));
                   ordenProduccion.setTipo(rs.getString("TIPO"));
                   ordenProduccion.setFecha_entrega(rs.getDate("FECHAE"));
                    ordenProduccions.add(ordenProduccion);
                }

            }

        } catch (SQLException e) {
            System.err.println("Error de conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                } catch (Exception e) {
                }
            }
        }

        return ordenProduccions;
    }

    @Override
    public OrdenProduccion mostrarId(OrdenProduccion o) {
        OrdenProduccion ordenproduccion = new OrdenProduccion();
        Connection cn= null;

        try {
            cn= Conexion.conectar();

            if (cn!=null) {

                String sql="SELECT B.IDOP OP, C.DESCRIABREV PROD, B.LOTE LOTE , B.FECHA_LANZAMIENTO FECHA, B.TIPO TIPO,
B.FECHA_ENTREGA FECHAE"
                        +" FROM ORDENPRODUCCION B"
                        +" INNER JOIN PRODUCTO C ON (C.ID=B.IDPRODUCTO)"
                        +" WHERE B.IDOP=? AND B.ESTADO='1'";
                PreparedStatement pstm= cn.prepareStatement(sql);
                pstm.setLong(1, o.getIdop());

                ResultSet rs= pstm.executeQuery();

                while (rs.next()) {
                    Producto producto=new Producto();
                    producto.setDescriabrev(rs.getString("PROD"));
                   ordenproduccion.setIdop(rs.getLong("OP"));
                    ordenproduccion.setProducto(producto);
                   ordenproduccion.setLote(rs.getDouble("LOTE"));
                  ordenproduccion.setFecha_lanzamiento(null);
                  ordenproduccion.setTipo(rs.getString("TIPO"));
                   ordenproduccion.setFecha_entrega(rs.getDate(null));
                }

            }

        } catch (SQLException e) {
            System.err.println("Error en la conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                } catch (Exception e) {
                }
            }
```

```
        }

        return ordenproduccion;
    }

    @Override
    public OrdenProduccion encuentraID(String nombre) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public boolean validar(String nombre) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public boolean eliminar() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public List<OrdenProduccion> mostrarCriteriolike(String nombre) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }



}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import pe.produccion.clases.Caracteristica;
import pe.produccion.clases.Dimension;
import pe.produccion.clases.ProdOpeMaq;
import pe.produccion.clases.ListaMateriales;
import pe.produccion.clases.Maquina;
import pe.produccion.clases.Operacion;
import pe.produccion.clases.ProduCarac;
import pe.produccion.clases.ProduDimen;


import pe.produccion.clases.Producto;

import pe.produccion.conexion.Conexion;

/**
 *
 * @author Rooll
 */
public class ProdOpeMaqImple implements BDInterf<ProdOpeMaq>{

    @Override
    public List<ProdOpeMaq> mostrar() {
        List<ProdOpeMaq> ensambles = new ArrayList<ProdOpeMaq>();
        Connection cn= null;
        try {
            cn= Conexion.conectar();
            if (cn!=null) {
                String sql = "SELECT A.DESCRIABREV PRODUCTO, D.DESCRIABREV OPE,"
                        + "  B.DESCRIABREV MAQ, TIEMPOCONFIG TCONF, TIEMPOPIEZA TPZA, NUMTRABAJADOR NUMTR FROM
PRODOPEMAQ C"
                          +" INNER JOIN PRODUCTO A ON (A.ID=C.IDPRODUCTO)"
                         +" INNER JOIN MAQUINA B ON (B.ID=C.IDMAQUINA)"
                        +" INNER JOIN OPERACION D ON (D.ID=C.IDOPERACION)"
                        +" WHERE C.ESTADO='1'";

                Statement st = cn.createStatement();
                ResultSet rs = st.executeQuery(sql);
                while (rs.next()) {
                    ProdOpeMaq ensamble = new ProdOpeMaq();
                    Producto producto = new Producto();
                    Maquina maquina = new Maquina();
                    Operacion operacion = new Operacion();
                   producto.setDescriabrev(rs.getString("PRODUCTO"));
```

```
                    maquina.setDescriabrev(rs.getString("MAQ"));
                    operacion.setDescriabrev(rs.getString("OPE"));
                    ensamble.setProducto(producto);
                    ensamble.setMaquina(maquina);
                    ensamble.setOperacion(operacion);
                    ensamble.setTiempoconfig(rs.getDouble("TCONF"));
                    ensamble.setTiempopieza(rs.getDouble("TPZA"));
                   ensamble.setNumtrabajador(rs.getDouble("NUMTR"));
                     ensambles.add(ensamble);
                  }

               }

            } catch (SQLException e) {
               System.err.println("Error de conexion "+e.getMessage());
            }finally{
               if (cn!=null) {
                  try {
                     cn.close();
                  } catch (Exception e) {
                  }
               }
            }

             return ensambles;
          }



          @Override
          public boolean insertar(ProdOpeMaq o) {

             boolean sw=false;
             Connection cn= null;
             long count=0;
             try {
                cn= Conexion.conectar();

                if (cn!=null) {
                   String sqlo="select COUNT(IDPRODUCTO) AS PROD FROM PRODOPEMAQ"
                           +" WHERE IDPRODUCTO=? AND ESTADO='1'";
                   PreparedStatement pstm = cn.prepareStatement(sqlo);
                   pstm.setLong(1, o.getProducto().getId());

                   ResultSet rs = pstm.executeQuery();
                   while (rs.next()) {
                     count=rs.getLong("PROD");
                   }
                   if (count<1) {
                      String sql= "INSERT INTO PRODOPEMAQ (IDPRODUCTO, IDMAQUINA, IDOPERACION, ESTADO, TIEMPOCONFIG,
TIEMPOPIEZA, NUMTRABAJADOR) VALUES (?,?,?,'1',?,?,?)";
                     PreparedStatement pstm2= cn.prepareStatement(sql);
                     pstm2.setDouble(1, o.getProducto().getId());
                     pstm2.setDouble(2, o.getMaquina().getId());
                     pstm2.setDouble(3, o.getOperacion().getId());
                     pstm2.setDouble(4, o.getTiempoconfig());
                     pstm2.setDouble(5, o.getTiempopieza());
                     pstm2.setDouble(6, o.getNumtrabajador());

                     pstm2.executeUpdate();
                      sw=true;
                      } else {
                         sw=false;
                         System.out.println("El producto ya tiene maquina asignada");
                      }


                }

             } catch (SQLException e) {
                sw=false;
                System.err.println("Error en la conexion "+e.getMessage());
             }finally{
                if (cn!=null) {
                   try {
                      cn.close();
                   } catch (Exception e) {

                   }
                }
             }

           return sw;
          }
```

```
    @Override
    public boolean actualizar(ProdOpeMaq o) {
        boolean sw;
       Connection cn= null;

       try {
          cn= Conexion.conectar();

          if (cn!=null) {

             String sql= "UPDATE PRODOPEMAQ SET IDMAQUINA=?, IDOPERACION=?, TIEMPOPIEZA=?, NUMTRABAJADOR=?,
TIEMPOCONFIG=?"
                   + " WHERE IDPRODUCTO=? and ESTADO='1'";

             PreparedStatement pstm= cn.prepareStatement(sql);
             pstm.setDouble(1, o.getMaquina().getId());
             pstm.setDouble(2, o.getOperacion().getId());
             pstm.setDouble(3, o.getTiempopieza());
             pstm.setDouble(4, o.getNumtrabajador());
             pstm.setDouble(5, o.getTiempoconfig());
             pstm.setDouble(6, o.getProducto().getId());

             pstm.executeUpdate();

          }
         sw=true;
       } catch (SQLException e) {
          sw=false;
          System.err.println("Error en la conexion "+e.getMessage());
       }finally{
          if (cn!=null) {
             try {
                cn.close();
             } catch (Exception e) {

             }
          }
       }

      return sw;
    }

    @Override
    public boolean eliminar(ProdOpeMaq o) {
        boolean sw;
       Connection cn= null;

       try {
          cn= Conexion.conectar();

          if (cn!=null) {

             String sql= "UPDATE PRODOPEMAQ SET ESTADO='0'"
                      +" WHERE IDPRODUCTO=? and ESTADO='1'";
             PreparedStatement pstm= cn.prepareStatement(sql);
             pstm.setDouble(1, o.getProducto().getId());


             pstm.executeUpdate();

          }
         sw=true;
       } catch (SQLException e) {
          sw=false;
          System.err.println("Error en la conexion "+e.getMessage());
       }finally{
          if (cn!=null) {
             try {
                cn.close();
             } catch (Exception e) {

             }
          }
       }

      return sw;
    }
```

```
    @Override
    public List<ProdOpeMaq> mostrarCriterio(String nombre) {
        List<ProdOpeMaq> ensambles = new ArrayList<ProdOpeMaq>();
       Connection cn= null;

       try {
          cn= Conexion.conectar();
          if (cn!=null) {
             String sql = "SELECT A.DESCRIABREV PRODUCTO, D.DESCRIABREV OPE, B.DESCRIABREV MAQ, TIEMPOCONFIG
TCONF, TIEMPOPIEZA TPZA, NUMTRABAJADOR NUMTR"
                    +" FROM PRODOPEMAQ C"
                   +" INNER JOIN PRODUCTO A ON (A.ID=C.IDPRODUCTO)"
                  +" INNER JOIN MAQUINA B ON (B.ID=C.IDMAQUINA)"
                 +" INNER JOIN OPERACION D ON (D.ID=C.IDOPERACION)"
                +" WHERE (UPPER(A.DESCRIABREV) LIKE UPPER(?) OR UPPER(B.DESCRIABREV) LIKE UPPER(?)) AND
C.ESTADO='1'";

             PreparedStatement pstm = cn.prepareStatement(sql);

             pstm.setString(1, "%"+nombre+"%");
             pstm.setString(2, "%"+nombre+"%");
             ResultSet rs = pstm.executeQuery();
             while (rs.next()) {
                ProdOpeMaq ensamble = new ProdOpeMaq();
                Producto producto = new Producto();
                Maquina maquina = new Maquina();
                Operacion operacion = new Operacion();
               producto.setDescriabrev(rs.getString("PRODUCTO"));
               maquina.setDescriabrev(rs.getString("MAQ"));
               operacion.setDescriabrev(rs.getString("OPE"));
               ensamble.setProducto(producto);
               ensamble.setMaquina(maquina);
               ensamble.setOperacion(operacion);
              ensamble.setTiempoconfig(rs.getDouble("TCONF"));
               ensamble.setTiempopieza(rs.getDouble("TPZA"));
              ensamble.setNumtrabajador(rs.getDouble("NUMTR"));
                ensambles.add(ensamble);
             }

          }

       } catch (SQLException e) {
          System.err.println("Error de conexion "+e.getMessage());
       }finally{
          if (cn!=null) {
             try {
                cn.close();
             } catch (Exception e) {
             }
          }
       }

        return ensambles;
    }

    @Override
    public ProdOpeMaq mostrarId(ProdOpeMaq o) {
        ProdOpeMaq ensamble = new ProdOpeMaq();
       Connection cn= null;

       try {
          cn= Conexion.conectar();
          if (cn!=null) {
             String sql = "SELECT A.DESCRIABREV PRODUCTO, D.DESCRIABREV OPE, B.DESCRIABREV MAQ, TIEMPOCONFIG
TCONF, TIEMPOPIEZA TPZA, NUMTRABAJADOR NUMTR"
                    +" FROM PRODOPEMAQ C"
                   +" INNER JOIN PRODUCTO A ON (A.ID=C.IDPRODUCTO)"
                  +" INNER JOIN MAQUINA B ON (B.ID=C.IDMAQUINA)"
                 +" INNER JOIN OPERACION D ON (D.ID=C.IDOPERACION)"
                +" WHERE C.IDPRODUCTO=? AND C.ESTADO='1'";

             PreparedStatement pstm = cn.prepareStatement(sql);

             pstm.setLong(1,o.getProducto().getId());

             ResultSet rs = pstm.executeQuery();
             while (rs.next()) {

                Producto producto = new Producto();
                Maquina maquina = new Maquina();
                Operacion operacion = new Operacion();
               producto.setDescriabrev(rs.getString("PRODUCTO"));
               maquina.setDescriabrev(rs.getString("MAQ"));
               operacion.setDescriabrev(rs.getString("OPE"));
               ensamble.setProducto(producto);
```

```
                ensamble.setMaquina(maquina);
                ensamble.setOperacion(operacion);
              ensamble.setTiempoconfig(rs.getDouble("TCONF"));
                ensamble.setTiempopieza(rs.getDouble("TPZA"));
               ensamble.setNumtrabajador(rs.getDouble("NUMTR"));

              }

            }


        } catch (SQLException e) {
            System.err.println("Error de conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                } catch (Exception e) {
                }
            }
        }

     return ensamble;
    }

    @Override
    public ProdOpeMaq encuentraID(String nombre) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public boolean validar(String nombre) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public boolean eliminar() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public List<ProdOpeMaq> mostrarCriteriolike(String nombre) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }



}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import pe.produccion.clases.ClaseProducto;
import pe.produccion.clases.Producto;
import pe.produccion.clases.TipoProducto;
import pe.produccion.clases.UnidMed;
import pe.produccion.conexion.Conexion;
/**
 *
 * @author Rooll
 */
public class ProductoImple implements BDInterf<Producto>{
    @Override
    public List<Producto> mostrar() {
        List<Producto> productos = new ArrayList<Producto>();
        Connection cn= null;
        try {
            cn= Conexion.conectar();
            if (cn!=null) {
                String sql = "SELECT A.ID ID, A.DESCRIABREV PRODUCTO, B.DESCRIABREV TIPO,"
                        +" C.DESCRIABREV CLASE, D.DESCRIABREV MEDIDA FROM PRODUCTO A, TIPOPRODUCTO B,"
                        +" CLASEPRODUCTO C, UNIDMED D WHERE ((A.TIPOPRODUCTO=B.ID AND A.CLASEPRODUCTO = C.ID) AND"
                        +" A.ESTADO='1' AND (A.MEDIDA=D.ID)) ORDER BY A.ID";

                Statement st = cn.createStatement();
```

```
                    ResultSet rs = st.executeQuery(sql);
                    while (rs.next()) {
                        Producto producto = new Producto();
                        TipoProducto tipoproducto = new TipoProducto();
                        ClaseProducto claseproducto = new ClaseProducto();
                        producto.setId(rs.getLong("ID"));
                        producto.setDescriabrev(rs.getString("PRODUCTO"));
                        tipoproducto.setDescriabrev(rs.getString("TIPO"));
                        producto.setTipoproducto(tipoproducto);
                        claseproducto.setDescriabrev(rs.getString("CLASE"));
                        producto.setClaseproducto(claseproducto);
                        UnidMed unimed= new UnidMed();
                        unimed.setDescriabrev(rs.getString("MEDIDA"));
                        producto.setUnidmed(unimed);
                        productos.add(producto);
                    }

                }
                    } catch (SQLException e) {
                System.err.println("Error de conexion "+e.getMessage());
            }finally{
                if (cn!=null) {
                    try {
                        cn.close();
                    } catch (Exception e) {
                    }
                }
            }

             return productos;
        }

        @Override
        public List<Producto> mostrarCriterio(String nombre) {
            List<Producto> productos = new ArrayList<Producto>();
            Connection cn= null;

            try {
                cn= Conexion.conectar();

                if (cn!=null) {

                    String sql="SELECT A.ID ID, A.DESCRIABREV ABREV, A.DESCRIEXTEN EXTEN, B.DESCRIABREV TIPO, C.DESCRIABREV
CLASE, D.DESCRIABREV MEDIDA"
                            +" FROM PRODUCTO A"
                            +" INNER JOIN TIPOPRODUCTO B ON (B.ID=A.TIPOPRODUCTO)"
                            +" INNER JOIN CLASEPRODUCTO C ON (C.ID=A.CLASEPRODUCTO)"
                             +" INNER JOIN UNIDMED D ON (D.ID=A.MEDIDA)"
                            +" WHERE (UPPER(A.DESCRIABREV) LIKE UPPER(?) OR UPPER(A.DESCRIEXTEN) LIKE UPPER(?))"
                             + " AND A.ESTADO='1' ORDER BY A.ID";
                    PreparedStatement pstm= cn.prepareStatement(sql);
                    pstm.setString(1, "%"+nombre+"%");
                    pstm.setString(2, "%"+nombre+"%");
                    ResultSet rs= pstm.executeQuery();

                    while (rs.next()) {
                        Producto producto = new Producto();
                        ClaseProducto claseproducto = new ClaseProducto();
                        TipoProducto tipoproducto = new TipoProducto();
                        producto.setId(rs.getLong("ID"));
                        producto.setDescriabrev(rs.getString("ABREV"));
                        producto.setDescriexten(rs.getString("EXTEN"));
                        claseproducto.setDescriabrev(rs.getString("CLASE"));
                        tipoproducto.setDescriabrev(rs.getString("TIPO"));
                        producto.setClaseproducto(claseproducto);
                        producto.setTipoproducto(tipoproducto);
                         UnidMed unimed= new UnidMed();
                        unimed.setDescriabrev(rs.getString("MEDIDA"));
                        producto.setUnidmed(unimed);
                        productos.add(producto);
                    }

                }

            } catch (SQLException e) {
                System.err.println("Error en la conexion "+e.getMessage());
            }finally{
                if (cn!=null) {
                    try {
                        cn.close();
                    } catch (Exception e) {
                    }
                }
            }

         return productos;
```

```
    }

    @Override
    public Producto mostrarId(Producto o) {
        Producto producto = new Producto();
        Connection cn= null;

        try {
            cn= Conexion.conectar();

            if (cn!=null) {

                String sql="SELECT A.ID ID, A.DESCRIABREV ABREV, A.DESCRIEXTEN EXTEN, B.DESCRIABREV TIPO, C.DESCRIABREV
CLASE, D.DESCRIABREV MEDIDA"
                        +" FROM PRODUCTO A"
                        +" INNER JOIN TIPOPRODUCTO B ON (B.ID=A.TIPOPRODUCTO)"
                        +" INNER JOIN CLASEPRODUCTO C ON (C.ID=A.CLASEPRODUCTO)"
                          +" INNER JOIN UNIDMED D ON (D.ID=A.MEDIDA)"
                        +" WHERE A.ID=? AND A.ESTADO='1'";
                    PreparedStatement pstm= cn.prepareStatement(sql);
                    pstm.setLong(1, o.getId());

                    ResultSet rs= pstm.executeQuery();

                    while (rs.next()) {

                        ClaseProducto claseproducto = new ClaseProducto();
                        TipoProducto tipoproducto = new TipoProducto();
                        producto.setId(rs.getLong("ID"));
                        producto.setDescriabrev(rs.getString("ABREV"));
                        producto.setDescriexten(rs.getString("EXTEN"));
                        claseproducto.setDescriabrev(rs.getString("CLASE"));
                        tipoproducto.setDescriabrev(rs.getString("TIPO"));
                        producto.setClaseproducto(claseproducto);
                        producto.setTipoproducto(tipoproducto);
                         UnidMed unimed= new UnidMed();
                        unimed.setDescriabrev(rs.getString("MEDIDA"));
                        producto.setUnidmed(unimed);

                    }

                }

        } catch (SQLException e) {
            System.err.println("Error en la conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                     Thread.sleep(150);
                } catch (Exception e) {
                }
            }
        }

         return producto;
    }

    @Override
    public boolean insertar(Producto o) {

        boolean sw;
        Connection cn= null;

        try {
            cn= Conexion.conectar();

            if (cn!=null) {
                String sqlmax="SELECT MAX(ID) max from PRODUCTO";
                Statement st = cn.createStatement();
                ResultSet rs = st.executeQuery(sqlmax);
                while (rs.next()) {
                    o.setId(rs.getLong("max")+1);
                }

                String sql= "INSERT INTO PRODUCTO VALUES (?,?,?,?,?,?,'1',?)";
                PreparedStatement pstm= cn.prepareStatement(sql);
                pstm.setLong(1, o.getId());
                pstm.setString(2, o.getDescriabrev());
                pstm.setString(3, o.getDescriexten());
                pstm.setLong(4, o.getTipoproducto().getId());
                pstm.setLong(5, o.getClaseproducto().getId());
                pstm.setLong(6, o.getUnidmed().getId());
                pstm.executeUpdate();

            }
```

```
     sw=true;
    } catch (SQLException e) {
       sw=false;
       System.err.println("Error en la conexion "+e.getMessage());
    }finally{
       if (cn!=null) {
          try {
             cn.close();
          } catch (Exception e) {

          }
       }
    }

  return sw;
}

@Override
public boolean actualizar(Producto o) {
     boolean sw;
    Connection cn= null;

    try {
       cn= Conexion.conectar();

       if (cn!=null) {

          String sql= "UPDATE PRODUCTO SET DESCRIABREV=?, DESCRIEXTEN=?,"
                +" TIPOPRODUCTO=?, CLASEPRODUCTO=?, MEDIDA=? WHERE ID=?";

          PreparedStatement pstm= cn.prepareStatement(sql);
          pstm.setLong(6, o.getId());
          pstm.setString(1, o.getDescriabrev());
          pstm.setString(2, o.getDescriexten());
          pstm.setLong(3, o.getTipoproducto().getId());
          pstm.setLong(4, o.getClaseproducto().getId());
          pstm.setLong(5, o.getUnidmed().getId());
          pstm.executeUpdate();

       }
      sw=true;
    } catch (SQLException e) {
       sw=false;
       System.err.println("Error en la conexion "+e.getMessage());
    }finally{
       if (cn!=null) {
          try {
             cn.close();
          } catch (Exception e) {

          }
       }
    }

  return sw;
}

@Override
public boolean eliminar(Producto o) {
      boolean sw;
    Connection cn= null;

    try {
       cn= Conexion.conectar();

       if (cn!=null) {

          String sql= "UPDATE PRODUCTO SET ESTADO='0' WHERE ID=?";

          PreparedStatement pstm= cn.prepareStatement(sql);
          pstm.setLong(1, o.getId());
          pstm.executeUpdate();
         }
      sw=true;
    } catch (SQLException e) {
       sw=false;
       System.err.println("Error en la conexion "+e.getMessage());
    }finally{
       if (cn!=null) {
          try {
             cn.close();
          } catch (Exception e) {

          }
       }
    }
```

```
 return sw;
}



@Override
public Producto encuentraID(String nombre) {
  Producto producto = new Producto();
    Connection cn= null;

    try {
        cn= Conexion.conectar();

        if (cn!=null) {

            String sql="SELECT A.ID FROM PRODUCTO A"
              +" WHERE (upper(A.DESCRIABREV)=upper(?) OR upper(A.DESCRIEXTEN)=upper(?))"
                  + " AND A.ESTADO='1'";
            PreparedStatement pstm= cn.prepareStatement(sql);
            pstm.setString(1, nombre);
            pstm.setString(2, nombre);
            ResultSet rs= pstm.executeQuery();

            while (rs.next()) {

               producto.setId(rs.getLong("ID"));

            }

        }

    } catch (SQLException e) {
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
               cn.close();
                Thread.sleep(150);
            } catch (Exception e) {
            }
        }
    }

  return producto;
}

@Override
public boolean validar(String nombre) {
   boolean sw = true;
    Connection cn= null;

    try {
        cn= Conexion.conectar();

        if (cn!=null) {

            String sql="SELECT A.ID FROM PRODUCTO A"
                  +" WHERE (upper(A.DESCRIABREV) =upper(trim(?)) OR"
                   +" upper(A.DESCRIEXTEN) =upper(trim(?)))"
                   + " AND A.ESTADO='1'";
            PreparedStatement pstm= cn.prepareStatement(sql);
            pstm.setString(1, nombre);
            pstm.setString(2, nombre);
            ResultSet rs= pstm.executeQuery();

            while (rs.next()) {

               if (rs.getLong("ID")>0) {
                  sw=false;
               }

            }

        }

    } catch (SQLException e) {
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
               cn.close();
            } catch (Exception e) {
            }
```

```
                }
            }

             return sw;
            }

            @Override
            public boolean eliminar() {
                throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
            }

            @Override
            public List<Producto> mostrarCriteriolike(String nombre) {
                throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
            }



        }


        /*
         * To change this license header, choose License Headers in Project Properties.
         * To change this template file, choose Tools | Templates
         * and open the template in the editor.
         */
        package pe.produccion.dao;

        import java.sql.Connection;
        import java.sql.PreparedStatement;
        import java.sql.ResultSet;
        import java.sql.SQLException;
        import java.sql.Statement;
        import java.util.ArrayList;
        import java.util.List;
        import pe.produccion.clases.ClaseProducto;
        import pe.produccion.clases.ConsumoLM;
        import pe.produccion.clases.ControlProduccion;
        import pe.produccion.clases.GanttProceso;
        import pe.produccion.clases.ListaMateriales;
        import pe.produccion.clases.Maquina;
        import pe.produccion.clases.OrdenProduccion;
        import pe.produccion.clases.ProdOpeMaq;
        import pe.produccion.clases.Producto;
        import pe.produccion.clases.SeguiProduccion;
        import pe.produccion.clases.TipoProducto;
        import pe.produccion.conexion.Conexion;

        /**
         *
         * @author Rooll
         */
        public class SeguiProduccionImple {

            public List<SeguiProduccion> mostrar() {
                List<SeguiProduccion> listasegui = new ArrayList<SeguiProduccion>();
                Connection cn= null;
                try {
                    cn= Conexion.conectar();
                    if (cn!=null) {
                        String sql = "SELECT A.IDOP ID, D.DESCRIABREV PRODUCTO, C.DESCRIABREV COMPONENTE, A.CANTIDAD
CANTIDAD,\n" +
                                "A.AVANCE AVANCE, A.SALDO SALDO, A.ESTADO ESTADO\n" +
                                "FROM SEGUIPRODUCCION A\n" +
                                "INNER JOIN ORDENPRODUCCION B ON(B.IDOP=A.IDOP) INNER JOIN PRODUCTO D ON(D.ID =B.IDPRODUCTO)\n"
+
                                "INNER JOIN PRODUCTO C ON(C.ID=A.IDCOMPONENTE)"
                                +" ORDER BY A.IDOP, C.DESCRIABREV";

                        Statement st = cn.createStatement();
                        ResultSet rs = st.executeQuery(sql);
                        while (rs.next()) {
                            SeguiProduccion segui = new SeguiProduccion();
                            segui.setOp(rs.getLong("ID"));
                            Producto producto = new Producto();
                            producto.setDescriabrev(rs.getString("COMPONENTE"));
                            segui.setComponente(producto);
                            Producto producto1 = new Producto();
                            producto1.setDescriabrev(rs.getString("PRODUCTO"));
                            segui.setProducto(producto1);

                            segui.setCantidad(rs.getDouble("CANTIDAD"));
                            segui.setAvance(rs.getDouble("AVANCE"));
                            segui.setSaldo(rs.getDouble("SALDO"));
                            segui.setEstado(rs.getString("ESTADO"));
                             listasegui.add(segui);
```

```
            }

        }

    } catch (SQLException e) {
        System.err.println("Error de conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {
            }
        }
    }

     return listasegui;
}

public boolean insertOP(OrdenProduccion op,String comp, double cantidad, String padre, double cantipadre, double ult){
 boolean sw=false;
    Connection cn= null;
    long count=0;
    try {
        cn= Conexion.conectar();

        if (cn!=null) {

                String sql= "INSERT INTO SEGUIPRODUCCION VALUES (?,?,?,0,?,?,0,?,0,0,?,?,?,?,0,?)";
            PreparedStatement pstm= cn.prepareStatement(sql);
            pstm.setDouble(1, op.getIdop());
            BDInterf<Producto> bd = new ProductoImple();
            Producto prod= bd.encuentraID(comp);
            Producto padr= bd.encuentraID(padre);
            //insertar la busqueda de la maquina y tiempo para el producto
            pstm.setDouble(2,prod.getId());
            pstm.setDouble(3,cantidad);
            pstm.setDouble(4,cantidad);
            pstm.setString(5,"Pendiente");
            pstm.setLong(6,padr.getId());
            pstm.setDouble(7,cantipadre);

             ProdOpeMaq pom= new ProdOpeMaq();
             pom.setProducto(prod);
             BDInterf<ProdOpeMaq> bd1 = new ProdOpeMaqImple();
             ProdOpeMaq pom2= bd1.mostrarId(pom);
             double tconfig=pom2.getTiempoconfig();
             double tpieza=pom2.getTiempopieza();
             double ttotal=tconfig+tpieza*cantidad;

             String maq=pom2.getMaquina().getDescriabrev();
             BDInterf<Maquina> bd2 = new MaquinaImple();
             Maquina maq2= bd2.encuentraID(maq);
             double maq3=maq2.getId();

        pstm.setDouble(8,maq3);
        pstm.setDouble(9,ttotal);
        pstm.setDouble(10,ttotal);
            pstm.setDouble(11,ult);
            pstm.executeUpdate();
             sw=true;
                    }

    } catch (SQLException e) {
        sw=false;
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
                Thread.sleep(150);
            } catch (Exception e) {

            }
        }
    }

  return sw;
}

public boolean validar(OrdenProduccion ope){
    boolean sw=true;
    Connection cn= null;
    try {
        cn=Conexion.conectar();
        if (cn!=null) {
```

```
            String sql="SELECT A.IDOP OP FROM SEGUIPRODUCCION A"
                    +" WHERE A.IDOP=?";
            PreparedStatement pstm= cn.prepareStatement(sql);
            pstm.setDouble(1, ope.getIdop());
            ResultSet rs = pstm.executeQuery();
            while (rs.next()) {
              sw=false;
              break;
            }


        }

    } catch (Exception e) {
        sw=false;
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {

            }
        }
    }
    return sw;
}

  public boolean actualizaTotalControl(){
 boolean sw=false;
    Connection cn= null;
    double can=0;

    try {
        cn= Conexion.conectar();

        if (cn!=null) {

              String sql= "SELECT IDOP OP, IDPRODUCTO PROD ,SUM(PRODUCCION) SUMA\n" +
                      "FROM CONTROLPRODUCCION\n" +
                      "WHERE ESTADO='1'\n" +
                      "GROUP BY IDOP, IDPRODUCTO";

            Statement stm= cn.createStatement();
            ResultSet rs = stm.executeQuery(sql);
            while (rs.next()) {

              String sql1="SELECT CANTIDAD CANT FROM SEGUIPRODUCCION\n" +
                        "WHERE IDOP=? AND IDCOMPONENTE=?";
              PreparedStatement pstm1=cn.prepareStatement(sql1);
              pstm1.setDouble(1, rs.getDouble("OP"));
              pstm1.setDouble(2, rs.getDouble("PROD"));
              ResultSet rs1= pstm1.executeQuery();
              while (rs1.next()) {
                can=rs1.getDouble("CANT");
              }

              String sql2="UPDATE SEGUIPRODUCCION SET AVANCE=?, SALDO=?, ESTADO=?\n" +
                      "WHERE IDOP=? AND IDCOMPONENTE=?";
              PreparedStatement pstm2=cn.prepareStatement(sql2);
              pstm2.setDouble(1, rs.getDouble("SUMA"));
              pstm2.setDouble(2, can-rs.getDouble("SUMA"));
              if ((can-rs.getDouble("SUMA"))<=0) {
                  pstm2.setString(3, "Finalizada");
              }else{
               pstm2.setString(3, "En Proceso");
              }
              pstm2.setDouble(4, rs.getDouble("OP"));
              pstm2.setDouble(5, rs.getDouble("PROD"));
               pstm2.executeUpdate();

            }
             sw=true;
              }

    } catch (SQLException e) {
        sw=false;
        System.err.println("Error en la conexion "+e.getMessage());
    }finally{
        if (cn!=null) {
            try {
                cn.close();
            } catch (Exception e) {

            }
        }
```

```
                }

            return sw;
        }


            public List<SeguiProduccion> mostrarcriterio(SeguiProduccion s) {
            List<SeguiProduccion> listasegui = new ArrayList<SeguiProduccion>();
            Connection cn= null;
            try {
                cn= Conexion.conectar();
                if (cn!=null) {
                    String sql = "SELECT A.IDOP ID, D.DESCRIABREV PRODUCTO, C.DESCRIABREV COMPONENTE, A.CANTIDAD
CANTIDAD,"
                            +" A.AVANCE AVANCE, A.SALDO SALDO, A.ESTADO ESTADO, A.IDPADRE PADRE, A.CANTIPADRE CANTIPADRE"
                            +" FROM SEGUIPRODUCCION A"
                           + " INNER JOIN ORDENPRODUCCION B ON(B.IDOP=A.IDOP) INNER JOIN PRODUCTO D ON(D.ID =B.IDPRODUCTO)"
                           + " INNER JOIN PRODUCTO C ON(C.ID=A.IDCOMPONENTE)"
                            +" WHERE A.IDOP=? AND A.IDCOMPONENTE=? AND (A.ESTADO='Pendiente' or A.ESTADO ='En Proceso')"
                            +" ORDER BY A.IDOP, C.DESCRIABREV";

                    PreparedStatement pstm = cn.prepareStatement(sql);
                    pstm.setLong(1, s.getOp());
                    pstm.setLong(2, s.getComponente().getId());
                    ResultSet rs = pstm.executeQuery();
                    while (rs.next()) {
                        SeguiProduccion segui = new SeguiProduccion();
                        segui.setOp(rs.getLong("ID"));
                        Producto producto = new Producto();
                        producto.setDescriabrev(rs.getString("COMPONENTE"));
                        segui.setComponente(producto);
                        Producto producto1 = new Producto();
                        producto1.setDescriabrev(rs.getString("PRODUCTO"));
                        segui.setProducto(producto1);

                        segui.setCantidad(rs.getDouble("CANTIDAD"));
                        segui.setAvance(rs.getDouble("AVANCE"));
                        segui.setSaldo(rs.getDouble("SALDO"));
                        segui.setEstado(rs.getString("ESTADO"));
                        segui.setIdpadre(rs.getLong("PADRE"));
                        segui.setCantipadre(rs.getDouble("CANTIPADRE"));
                         listasegui.add(segui);
                    }

                }

            } catch (SQLException e) {
                System.err.println("Error de conexion "+e.getMessage());
            }finally{
                if (cn!=null) {
                    try {
                        cn.close();
                    } catch (Exception e) {
                    }
                }
            }

             return listasegui;
        }
          public List<SeguiProduccion> mostrarcriteriointerno(SeguiProduccion s) {
            List<SeguiProduccion> listasegui = new ArrayList<SeguiProduccion>();
            Connection cn= null;
            try {
                cn= Conexion.conectar();
                if (cn!=null) {
                    String sql = "SELECT A.IDOP ID, D.DESCRIABREV PRODUCTO, C.DESCRIABREV COMPONENTE, A.CANTIDAD
CANTIDAD,\n" +
                            "A.AVANCE AVANCE, A.SALDO SALDO, A.ESTADO ESTADO\n" +
                            "FROM SEGUIPRODUCCION A\n" +
                            "INNER JOIN ORDENPRODUCCION B ON(B.IDOP=A.IDOP) INNER JOIN PRODUCTO D ON(D.ID =B.IDPRODUCTO)\n"
+
                            "INNER JOIN PRODUCTO C ON(C.ID=A.IDCOMPONENTE)"
                            +" WHERE A.IDCOMPONENTE=? AND (A.ESTADO='Pendiente' or A.ESTADO ='En Proceso') AND A.IDOP <> ?"
                            +" ORDER BY A.IDOP, C.DESCRIABREV";

                    PreparedStatement pstm = cn.prepareStatement(sql);

                    pstm.setLong(1, s.getComponente().getId());
                    pstm.setLong(2, s.getOp());
                    ResultSet rs = pstm.executeQuery();
                    while (rs.next()) {
                        SeguiProduccion segui = new SeguiProduccion();
                        segui.setOp(rs.getLong("ID"));
                        Producto producto = new Producto();
                        producto.setDescriabrev(rs.getString("COMPONENTE"));
                        segui.setComponente(producto);
```

```
                Producto producto1 = new Producto();
                producto1.setDescriabrev(rs.getString("PRODUCTO"));
                segui.setProducto(producto1);

                segui.setCantidad(rs.getDouble("CANTIDAD"));
                segui.setAvance(rs.getDouble("AVANCE"));
                segui.setSaldo(rs.getDouble("SALDO"));
                segui.setEstado(rs.getString("ESTADO"));
                 listasegui.add(segui);
              }

          }

      } catch (SQLException e) {
          System.err.println("Error de conexion "+e.getMessage());
      }finally{
          if (cn!=null) {
              try {
                  cn.close();
              } catch (Exception e) {
              }
          }
      }

      return listasegui;
  }

  public boolean actualizar(SeguiProduccion s) {
      boolean sw;
      double avance=0;
      double stock=0;
     Connection cn= null;

      try {
          cn= Conexion.conectar();

          if (cn!=null) {
              String sql2="SELECT AVANCE AVAN, STOCK STOCK FROM SEGUIPRODUCCION"
                      +" WHERE IDOP=? AND IDCOMPONENTE=? AND CANTIDAD=?";
              PreparedStatement pstm2= cn.prepareStatement(sql2);
               pstm2.setLong(1, s.getOp());
              pstm2.setLong(2, s.getComponente().getId());
              pstm2.setDouble(3, s.getCantidad());
               ResultSet rs = pstm2.executeQuery();
               while (rs.next()) {
                  avance= rs.getDouble("AVAN");
                  stock= rs.getDouble("STOCK");
               }


              String sql= "UPDATE SEGUIPRODUCCION SET AVANCE=?, SALDO=?,"
                    +" ESTADO=?, STOCK=? WHERE IDOP=? AND IDCOMPONENTE=? AND CANTIDAD=?";

              PreparedStatement pstm= cn.prepareStatement(sql);
              pstm.setDouble(1, avance +s.getAvance());

              pstm.setDouble(2,s.getCantidad()-(avance+s.getAvance()));
               if (s.getCantidad()-(avance+s.getAvance())<=0) {
                   pstm.setString(3, "Finalizada");
               } else {
                   pstm.setString(3, "En Proceso");
               }
              pstm.setDouble(4, stock+s.getAvance());
              pstm.setLong(5, s.getOp());
              pstm.setLong(6, s.getComponente().getId());
              pstm.setDouble(7, s.getCantidad());
              pstm.executeUpdate();

          }
         sw=true;
      } catch (SQLException e) {
          sw=false;
          System.err.println("Error en la conexion "+e.getMessage());
      }finally{
          if (cn!=null) {
              try {
                  cn.close();
              } catch (Exception e) {

              }
          }
      }

     return sw;
  }
      public boolean actualizarunico(SeguiProduccion s) {
```

```
                boolean sw;
                double avance=0;
                double stock=0;
                double cantidad=0;
               Connection cn= null;

               try {
                  cn= Conexion.conectar();

                  if (cn!=null) {
                    String sql2="SELECT AVANCE AVAN, STOCK STOCK, CANTIDAD CANTI FROM SEGUIPRODUCCION"
                                +" WHERE IDOP=? AND IDCOMPONENTE=? AND (ESTADO='Pendiente' or ESTADO ='En Proceso')";
                     PreparedStatement pstm2= cn.prepareStatement(sql2);
                      pstm2.setLong(1, s.getOp());
                     pstm2.setLong(2, s.getComponente().getId());

                     ResultSet rs = pstm2.executeQuery();
                     while (rs.next()) {
                       avance= rs.getDouble("AVAN");
                       stock= rs.getDouble("STOCK");
                       cantidad= rs.getDouble("CANTI");
                     }


                     String sql= "UPDATE SEGUIPRODUCCION SET AVANCE=?, SALDO=?,"
                           +" ESTADO=?, STOCK=? WHERE IDOP=? AND IDCOMPONENTE=? AND (ESTADO='Pendiente' or ESTADO ='En Proceso')";

                     PreparedStatement pstm= cn.prepareStatement(sql);
                     pstm.setDouble(1, avance +s.getAvance());
                     pstm.setDouble(2,cantidad-(avance+s.getAvance()));
                      if (cantidad-(avance+s.getAvance())<=0) {
                         pstm.setString(3, "Finalizada");
                      } else {
                         pstm.setString(3, "En Proceso");
                      }
                     pstm.setDouble(4,stock+s.getAvance());
                     pstm.setLong(5, s.getOp());
                     pstm.setLong(6, s.getComponente().getId());

                     pstm.executeUpdate();

                   }
                  sw=true;
                } catch (SQLException e) {
                   sw=false;
                   System.err.println("Error en la conexion "+e.getMessage());
                }finally{
                   if (cn!=null) {
                      try {
                         cn.close();
                      } catch (Exception e) {

                      }
                   }
                }

              return sw;
             }
             public SeguiProduccion muestraMP (SeguiProduccion s) {
              SeguiProduccion segui = new SeguiProduccion();
               Connection cn= null;
               try {
                  cn= Conexion.conectar();
                  if (cn!=null) {
                     String sql = "SELECT A.IDOP ID, D.DESCRIABREV PRODUCTO, C.DESCRIABREV COMPONENTE, A.CANTIDAD
CANTIDAD,"
                                 +" A.AVANCE AVANCE, A.SALDO SALDO, A.ESTADO ESTADO,"
                                 +" A.IDPADRE PADRE,A.CANTIPADRE CANTIPADRE, A.STOCK STOCK, A.ASIGNADO ASIGNADO"
                                 +" FROM SEGUIPRODUCCION A"
                                +" INNER JOIN ORDENPRODUCCION B ON(B.IDOP=A.IDOP) INNER JOIN PRODUCTO D ON(D.ID =B.IDPRODUCTO)"
                                +" INNER JOIN PRODUCTO C ON(C.ID=A.IDCOMPONENTE)"
                                 +" WHERE A.IDOP=? AND A.IDCOMPONENTE=? AND A.IDPADRE =? AND A.CANTIPADRE=?"
                                 +" ORDER BY C.DESCRIABREV";

                     PreparedStatement pstm = cn.prepareStatement(sql);
                     pstm.setLong(1, s.getOp());
                     pstm.setLong(2, s.getComponente().getId());
                     pstm.setLong(3, s.getIdpadre());
                     pstm.setDouble(4, s.getCantipadre());
                     ResultSet rs = pstm.executeQuery();
                     while (rs.next()) {

                        segui.setOp(rs.getLong("ID"));
                        Producto producto = new Producto();
                        producto.setDescriabrev(rs.getString("COMPONENTE"));
                        segui.setComponente(producto);
```

```
                    Producto producto1 = new Producto();
                    producto1.setDescriabrev(rs.getString("PRODUCTO"));
                    segui.setProducto(producto1);

                    segui.setCantidad(rs.getDouble("CANTIDAD"));
                    segui.setAvance(rs.getDouble("AVANCE"));
                    segui.setSaldo(rs.getDouble("SALDO"));
                    segui.setEstado(rs.getString("ESTADO"));
                    segui.setIdpadre(rs.getLong("PADRE"));
                    segui.setCantipadre(rs.getDouble("CANTIPADRE"));
                    segui.setStock(rs.getDouble("STOCK"));
                    segui.setAsignado(rs.getDouble("ASIGNADO"));

                }

            }

        } catch (SQLException e) {
            System.err.println("Error de conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                } catch (Exception e) {
                }
            }
        }

        return segui;
    }
    public SeguiProduccion muestraMPUnicoPadre (SeguiProduccion s) {
     SeguiProduccion segui = new SeguiProduccion();
      Connection cn= null;
      try {
         cn= Conexion.conectar();
         if (cn!=null) {
            String sql = "SELECT A.IDOP ID, D.DESCRIABREV PRODUCTO, C.DESCRIABREV COMPONENTE, A.CANTIDAD
CANTIDAD,"
                    +" A.AVANCE AVANCE, A.SALDO SALDO, A.ESTADO ESTADO,"
                    +" A.IDPADRE PADRE,A.CANTIPADRE CANTIPADRE, A.STOCK STOCK, A.ASIGNADO ASIGNADO"
                    +" FROM SEGUIPRODUCCION A"
                   +" INNER JOIN ORDENPRODUCCION B ON(B.IDOP=A.IDOP) INNER JOIN PRODUCTO D ON(D.ID =B.IDPRODUCTO)"
                   +" INNER JOIN PRODUCTO C ON(C.ID=A.IDCOMPONENTE)"
                    +" WHERE A.IDOP=? AND A.IDCOMPONENTE=? AND A.IDPADRE =?"
                    +" ORDER BY C.DESCRIABREV";

            PreparedStatement pstm = cn.prepareStatement(sql);
            pstm.setLong(1, s.getOp());
            pstm.setLong(2, s.getComponente().getId());
            pstm.setLong(3, s.getIdpadre());

            ResultSet rs = pstm.executeQuery();
            while (rs.next()) {

               segui.setOp(rs.getLong("ID"));
               Producto producto = new Producto();
               producto.setDescriabrev(rs.getString("COMPONENTE"));
               segui.setComponente(producto);
               Producto producto1 = new Producto();
               producto1.setDescriabrev(rs.getString("PRODUCTO"));
               segui.setProducto(producto1);

               segui.setCantidad(rs.getDouble("CANTIDAD"));
               segui.setAvance(rs.getDouble("AVANCE"));
               segui.setSaldo(rs.getDouble("SALDO"));
               segui.setEstado(rs.getString("ESTADO"));
               segui.setIdpadre(rs.getLong("PADRE"));
               segui.setCantipadre(rs.getDouble("CANTIPADRE"));
               segui.setStock(rs.getDouble("STOCK"));
               segui.setAsignado(rs.getDouble("ASIGNADO"));

            }

         }

      } catch (SQLException e) {
         System.err.println("Error de conexion "+e.getMessage());
      }finally{
         if (cn!=null) {
            try {
               cn.close();
            } catch (Exception e) {
            }
         }
      }
```

```
        return segui;
    }
    public boolean actualConsumoMP(SeguiProduccion s) {
        boolean sw;
        double cantidad=0;
        double stock=0;
        double asignado=0;
       Connection cn= null;

       try {
          cn= Conexion.conectar();

          if (cn!=null) {
            String sql2="SELECT STOCK STOCK, ASIGNADO ASIG FROM SEGUIPRODUCCION"
                        +" WHERE IDOP=? AND IDCOMPONENTE=? AND IDPADRE=? AND CANTIPADRE=?";
             PreparedStatement pstm2= cn.prepareStatement(sql2);
              pstm2.setLong(1, s.getOp());
             pstm2.setLong(2, s.getComponente().getId());
             pstm2.setDouble(3, s.getIdpadre());
             pstm2.setDouble(4, s.getCantipadre());
              ResultSet rs = pstm2.executeQuery();
              while (rs.next()) {

                stock= rs.getDouble("STOCK");
                asignado= rs.getDouble("ASIG");
              }


             String sql= "UPDATE SEGUIPRODUCCION SET ASIGNADO=?, STOCK=? "
                   +"WHERE IDOP=? AND IDCOMPONENTE=? AND IDPADRE=? AND CANTIPADRE=?";

             PreparedStatement pstm= cn.prepareStatement(sql);
             pstm.setDouble(1, asignado+s.getAsignado());
             pstm.setDouble(2,stock-s.getAsignado());

             pstm.setLong(3, s.getOp());
             pstm.setLong(4, s.getComponente().getId());
             pstm.setDouble(5, s.getIdpadre());
             pstm.setDouble(6, s.getCantipadre());
             pstm.executeUpdate();

           }
         sw=true;
       } catch (SQLException e) {
          sw=false;
          System.err.println("Error en la conexion "+e.getMessage());
       }finally{
          if (cn!=null) {
             try {
                cn.close();
             } catch (Exception e) {

             }
          }
       }

     return sw;
    }

    public List<SeguiProduccion> muestraMPUnicoPadreOI (SeguiProduccion s) {
        List<SeguiProduccion> listasegui = new ArrayList<SeguiProduccion>();

      Connection cn= null;
      try {
         cn= Conexion.conectar();
         if (cn!=null) {
            String sql = "SELECT A.IDOP ID, B.TIPO TIPO, D.DESCRIABREV PRODUCTO, C.DESCRIABREV COMPONENTE,
A.CANTIDAD CANTIDAD,"
                    +" A.AVANCE AVANCE, A.SALDO SALDO, A.ESTADO ESTADO,"
                    +" A.IDPADRE PADRE,A.CANTIPADRE CANTIPADRE, A.STOCK STOCK, A.ASIGNADO ASIGNADO"
                    +" FROM SEGUIPRODUCCION A"
                   +" INNER JOIN ORDENPRODUCCION B ON(B.IDOP=A.IDOP) INNER JOIN PRODUCTO D ON(D.ID =B.IDPRODUCTO)"
                   +" INNER JOIN PRODUCTO C ON(C.ID=A.IDCOMPONENTE)"
                    +" WHERE B.TIPO='Interna' AND A.IDCOMPONENTE=? AND A.IDOP <> ?"
                    +" ORDER BY C.DESCRIABREV";

            PreparedStatement pstm = cn.prepareStatement(sql);
            //pstm.setLong(1, s.getOp());
            pstm.setLong(1, s.getComponente().getId());
            pstm.setLong(2, s.getOp());

            ResultSet rs = pstm.executeQuery();
            while (rs.next()) {
               SeguiProduccion segui = new SeguiProduccion();
               segui.setOp(rs.getLong("ID"));
               Producto producto = new Producto();
```

```
                producto.setDescriabrev(rs.getString("COMPONENTE"));
                segui.setComponente(producto);
                Producto producto1 = new Producto();
                producto1.setDescriabrev(rs.getString("PRODUCTO"));
                segui.setProducto(producto1);

                segui.setCantidad(rs.getDouble("CANTIDAD"));
                segui.setAvance(rs.getDouble("AVANCE"));
                segui.setSaldo(rs.getDouble("SALDO"));
                segui.setEstado(rs.getString("ESTADO"));
                segui.setIdpadre(rs.getLong("PADRE"));
                segui.setCantipadre(rs.getDouble("CANTIPADRE"));
                segui.setStock(rs.getDouble("STOCK"));
                segui.setAsignado(rs.getDouble("ASIGNADO"));
                listasegui.add(segui);
              }

            }

        } catch (SQLException e) {
            System.err.println("Error de conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                } catch (Exception e) {
                }
            }
        }

        return listasegui;
    }

    public boolean actualConsumoMPOI(SeguiProduccion s) {
        boolean sw;
        double cantidad=0;
        double stock=0;
        double asignado=0;
       Connection cn= null;

        try {
            cn= Conexion.conectar();

            if (cn!=null) {
                String sql2="SELECT STOCK STOCK, ASIGNADO ASIG FROM SEGUIPRODUCCION"
                            +" WHERE IDOP=? AND IDCOMPONENTE=?";
                 PreparedStatement pstm2= cn.prepareStatement(sql2);
                  pstm2.setLong(1, s.getOp());
                pstm2.setLong(2, s.getComponente().getId());

                 ResultSet rs = pstm2.executeQuery();
                 while (rs.next()) {

                   stock= rs.getDouble("STOCK");
                   asignado= rs.getDouble("ASIG");
                 }


                String sql= "UPDATE SEGUIPRODUCCION SET ASIGNADO=?, STOCK=? "
                      +"WHERE IDOP=? AND IDCOMPONENTE=?";

                PreparedStatement pstm= cn.prepareStatement(sql);
                pstm.setDouble(1, asignado+s.getAsignado());
                pstm.setDouble(2,stock-s.getAsignado());

                pstm.setLong(3, s.getOp());
                pstm.setLong(4, s.getComponente().getId());

                pstm.executeUpdate();

            }
           sw=true;
        } catch (SQLException e) {
            sw=false;
            System.err.println("Error en la conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                } catch (Exception e) {

                }
            }
        }

       return sw;
```

```
    }
    public List<GanttProceso> mostrarids(double[][] orden) {
        List<GanttProceso> ganttproceso = new ArrayList<GanttProceso>();
        Connection cn= null;
        try {
            cn= Conexion.conectar();
            if (cn!=null) {
                for (int i = 0; i < orden.length; i++) {
                    if (orden[i][0]>0) {



                    String sql = "SELECT A.IDOP ID, A.IDCOMPONENTE COMPONENTE, A.IDPADRE PADRE, A.CANTIPADRE CANTIPADRE
,A.MAQUINA MAQUINA,"
                            +"A.SALDO SALDO, A.TIEMPOSALDO TIEMPO, A.ULTIMO ULTIMO, A.ESTADO ESTADO"
                            +" FROM SEGUIPRODUCCION A"
                            + " WHERE A.IDOP=? AND A.MAQUINA > 0 AND A.ESTADO <> 'Finalizada' ORDER BY A.IDOP, A.IDCOMPONENTE" ;

                    PreparedStatement pstm = cn.prepareStatement(sql);
                    pstm.setDouble(1, orden[i][0]);
                    ResultSet rs = pstm.executeQuery();
                    while (rs.next()) {
                        GanttProceso gant= new GanttProceso();
                        OrdenProduccion op = new OrdenProduccion();
                        op.setIdop(rs.getLong("ID"));
                        gant.setOp(op);
                        Producto comp= new Producto();
                        comp.setId(rs.getLong("COMPONENTE"));
                        gant.setComponente(comp);
                        Producto padre= new Producto();
                        padre.setId(rs.getLong("PADRE"));
                        gant.setPadre(padre);
                        Maquina maq= new Maquina();
                        maq.setId(rs.getLong("MAQUINA"));
                        gant.setMaquina(maq);
                        gant.setCantipadre(rs.getDouble("CANTIPADRE"));
                        gant.setSaldo(rs.getDouble("SALDO"));
                        gant.setTiemposaldo(rs.getDouble("TIEMPO"));
                        gant.setUltimo(rs.getDouble("ULTIMO"));
                        gant.setEstado(rs.getString("ESTADO"));


                         ganttproceso.add(gant);
                    }
                    }
                    }
                }

            } catch (SQLException e) {
                System.err.println("Error de conexion "+e.getMessage());
            }finally{
                if (cn!=null) {
                    try {
                        cn.close();
                    } catch (Exception e) {
                    }
                }
            }

            return ganttproceso;
    }
}
```

9.5 ANEXO E Codificación Recursivo

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.repor;

import java.util.ArrayList;
import java.util.List;
import java.util.Vector;
import pe.produccion.clases.ConsumoLM;
import pe.produccion.clases.ListaMateriales;
import pe.produccion.clases.Producto;
import pe.produccion.dao.BDInterf;
```

```
import pe.produccion.dao.ConsumoLMImple;
import pe.produccion.dao.ListaMaterImple;
import pe.produccion.dao.ProductoImple;



/**
 *
 * @author Rooll
 */
public class ReporteLista {

    public static String vec[][]= new String[100000][5];
  public static int i=0;
  public static int j=0;

   public List<String> muestradistin(String mues){
     // System.out.println(mues);
      i=0;
      j=0;
      for (int k = 0; k < 1000; k++) {
          for (int l = 0; l < 4; l++) {
              vec[k][l]="";
          }
      }

      recur(mues);
     int rep=0;
      List<String> lista = new ArrayList<>();
      for (int k = 0; k < 10000; k++) {
          rep=0;
          if (vec[k][0].isEmpty()){
            break;
          }else{
              if (k==0) {
                lista.add(vec[k][0]);
              }else{
              for (int m = k-1; m >= 0; m--) {
                  if (vec[k][0].equals(vec[m][0])) {
                      rep=1;
                    break;

                  }
                }
                  if (rep==0) {
                      lista.add(vec[k][0]);
                  }
              }
           }

      }
      return lista;
   }

  public List<String> muestraglobal(String mues){
     // System.out.println(mues);
      i=0;
      j=0;
      for (int k = 0; k < 1000; k++) {
          for (int l = 0; l < 3; l++) {
              vec[k][l]="";
          }
      }

      recur(mues);

      List<String> lista = new ArrayList<>();
      for (int k = 0; k < 10000; k++) {

          if (vec[k][0].isEmpty()){
            break;
          }else{
              if (k==0) {
                lista.add(vec[k][0]);
              }else{


                      lista.add(vec[k][0]);

              }
           }

      }
      return lista;
   }
```

```
  public String[][] consumo(String mues){
  // System.out.println(mues);

     String[][]lista= new String[1000][3];
    i=0;
    j=0;
    for (int k = 0; k < 1000; k++) {
       for (int l = 0; l < 3; l++) {
          vec[k][l]="";
       }
    }

    recur(mues);


    for (int k = 0; k < 10000; k++) {

         if (vec[k][0].isEmpty()){
         break;
        }else{
            lista[k][0]=vec[k][0];
            lista[k][1]=vec[k][1];
            lista[k][2]=vec[k][2];


        }
     }
    return lista;
}

public String[][] consumocantidad(String mues, Double cant){
  // System.out.println(mues);

     String[][]lista= new String[1000][5];
    i=0;
    j=0;
    for (int k = 0; k < 1000; k++) {
       for (int l = 0; l < 5; l++) {
          vec[k][l]="";
       }
    }

    recur(mues);


    for (int k = 0; k < 10000; k++) {

         if (vec[k][0].isEmpty()){
         break;
        }else{
         lista[k][0]=vec[k][0];
            lista[k][1]=String.valueOf(Double.parseDouble(vec[k][1])*cant);
            lista[k][2]=vec[k][2];
            lista[k][3]=String.valueOf(Double.parseDouble(vec[k][3])*cant);;
            lista[k][4]=vec[k][4];
        }
     }
    return lista;
}

static void recur(String nombre){

     BDInterf bd = new ListaMaterImple();
    List<ListaMateriales> listaMateriales = bd.mostrar();
    if (i==0) {
      vec[i][0]=nombre;
      vec[i][1]="1";
      vec[i][2]="";
      vec[i][3]="1";
      vec[i][4]="0";
      i=i+1;
  }
     for (ListaMateriales listaMateriale1 : listaMateriales) {
        if (listaMateriale1.getProducto().getDescriabrev().equals(nombre)) {
         //  Prueba6.abc =Prueba6.abc +"***"+listaMateriale1.getComponente().getDescriabrev();


           vec[i][0]=listaMateriale1.getComponente().getDescriabrev();
           vec[i][2]=nombre;
            vec[i][4]="0";
           //vec[i][3]=vec[i-1][1];
            if (i>0) {
                for (int k = i-1; k >= 0; k--) {
                    if (vec[k][0].equals(nombre)) {
                        vec[i][1]=String.valueOf(listaMateriale1.getCantidad_comp()*Double.parseDouble(vec[k][1]));
```

```
                    vec[i][3]=vec[k][1];
                    break;
                  }
                }
              }

              i=i+1;
              recur(listaMateriale1.getComponente().getDescriabrev());

            }
          }
          vec[i-1][4]="1";

        }

        public String [][] globalPadre(String mues){
          // System.out.println(mues);
           i=0;
          j=0;
          for (int k = 0; k < 1000; k++) {
              for (int l = 0; l < 3; l++) {
                vec[k][l]="";
              }
          }

          recur(mues);

          String lis[][] = new String[1000][3];
          for (int k = 0; k < 10000; k++) {

            if (vec[k][0].isEmpty()){
              break;
            }else{
              if (k==0) {
                lis[k][0]=vec[k][0];
                lis[k][2]=vec[k][2];
              }else{

                lis[k][0]=vec[k][0];
                lis[k][2]=vec[k][2];

              }
             }

          }
          return lis;
        }
}
```

9.6 ANEXO F Codificación Vistas

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.vistas;

import java.sql.Date;
import java.util.ArrayList;
import java.util.List;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import pe.produccion.clases.Caracteristica;
import pe.produccion.clases.ClaseProducto;
import pe.produccion.clases.ConsumoLM;
import pe.produccion.clases.ListaMateriales;
import pe.produccion.clases.OrdenProduccion;
import pe.produccion.clases.Producto;
import pe.produccion.clases.TipoProducto;
import pe.produccion.dao.BDInterf;
import pe.produccion.dao.CaracterImple;
import pe.produccion.dao.ClaseProdImple;
import pe.produccion.dao.ConsumoLMImple;
import pe.produccion.dao.ListaMaterImple;
import pe.produccion.dao.OrdenProdImple;
import pe.produccion.dao.ProductoImple;
import pe.produccion.dao.TipoProdImple;
```

```
import pe.pruebas.Prueba6;
import static pe.pruebas.Prueba6.i;
import static pe.pruebas.Prueba6.vec;

/**
 *
 * @author Rooll
 */
public class ConsumoLMVIew extends javax.swing.JInternalFrame {

    /**
     * Creates new form Interno1
     */
  public static String abc;
   public static String vec[][]= new String[100000][3];
  public static int i=0;
  public static int j=0;

    public ConsumoLMVIew() {
        initComponents();

        cargaProducto();


    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        txtBusqueda = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        JScrollPane = new javax.swing.JScrollPane();
        tblDatos = new javax.swing.JTable();
        btnBuscar = new javax.swing.JButton();
        jPanel2 = new javax.swing.JPanel();
        btnConsumo = new javax.swing.JButton();
        cbProducto = new javax.swing.JComboBox();
        jLabel5 = new javax.swing.JLabel();

        setClosable(true);
        setMaximizable(true);
        setResizable(true);
        setTitle("CONSUMO LISTA MATERIALES");

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jLabel1.setText("FILTRO");

        tblDatos.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null}
            },
            new String [] {
                "Title 1", "Title 2", "Title 3", "Title 4"
            }
        ));
        tblDatos.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                tblDatosMouseClicked(evt);
            }
        });
        JScrollPane.setViewportView(tblDatos);

        btnBuscar.setText("BUSCAR");
        btnBuscar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnBuscarActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
```

```
                                .addGap(55, 55, 55)
                                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 59, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(46, 46, 46)
                                .addComponent(txtBusqueda, javax.swing.GroupLayout.PREFERRED_SIZE, 372, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(56, 56, 56)
                                .addComponent(btnBuscar, javax.swing.GroupLayout.PREFERRED_SIZE, 101, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel1Layout.createSequentialGroup()
                                .addContainerGap()
                                .addComponent(JScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 736, javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addContainerGap(27, Short.MAX_VALUE))
                );
                jPanel1Layout.setVerticalGroup(
                    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(txtBusqueda,          javax.swing.GroupLayout.PREFERRED_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel1)
                            .addComponent(btnBuscar))
                        .addGap(18, 18, 18)
                        .addComponent(JScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 394, Short.MAX_VALUE)
                        .addContainerGap())
                );

                btnConsumo.setText("CONSUMO");
                btnConsumo.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        btnConsumoActionPerformed(evt);
                    }
                });

                cbProducto.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Item 1", "Item 2", "Item 3", "Item 4" }));
                cbProducto.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        cbProductoActionPerformed(evt);
                    }
                });

                jLabel5.setText("PRODUCTO");

                javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
                jPanel2.setLayout(jPanel2Layout);
                jPanel2Layout.setHorizontalGroup(
                    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(21, 21, 21)
                        .addComponent(jLabel5)
                        .addGap(34, 34, 34)
                        .addComponent(cbProducto, javax.swing.GroupLayout.PREFERRED_SIZE, 282, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(89, 89, 89)
                        .addComponent(btnConsumo, javax.swing.GroupLayout.PREFERRED_SIZE, 109, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                );
                jPanel2Layout.setVerticalGroup(
                    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(btnConsumo,          javax.swing.GroupLayout.DEFAULT_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(cbProducto,          javax.swing.GroupLayout.PREFERRED_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jLabel5)))
                        .addGap(61, 61, 61))
                );

                javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
                getContentPane().setLayout(layout);
                layout.setHorizontalGroup(
                    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jPanel2,                javax.swing.GroupLayout.DEFAULT_SIZE,                javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addContainerGap())
                );
                layout.setVerticalGroup(
                    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jPanel1,                javax.swing.GroupLayout.PREFERRED_SIZE,                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(jPanel2,                javax.swing.GroupLayout.PREFERRED_SIZE,                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addContainerGap(15, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnBuscarActionPerformed
        // TODO add your handling code here:
        String n= txtBusqueda.getText();


    }//GEN-LAST:event_btnBuscarActionPerformed

    private void tblDatosMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_tblDatosMouseClicked
        // TODO add your handling code here:


    }//GEN-LAST:event_tblDatosMouseClicked

    private void btnConsumoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnConsumoActionPerformed
        // TODO add your handling code here:

        if (cbProducto.getSelectedIndex()==-1) {
            return;
        }
        for (int x = 0; x < 10000; x++) {
            for (int y = 0; y < 3; y++) {
                vec[x][y]="";
            }
        }
        i=0;
        j=0;

        if (JOptionPane.showConfirmDialog(this, "Mostrar Consumo?", "AVISO", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE)
                ==JOptionPane.OK_OPTION) {
             BDInterf<ConsumoLM> bdconsumo = new ConsumoLMImple();
            ConsumoLM consumolm= new ConsumoLM();
            if (!bdconsumo.eliminar()) {
                JOptionPane.showMessageDialog(this, "Error de Eliminacion de Valores", "AVISO", JOptionPane.ERROR_MESSAGE);
                return;
            }

            String prod= cbProducto.getSelectedItem().toString();
            recur(prod);
        //cargar la tabla
        String cabecera[]={"PRODUCTO","COMPONENTE","CANTIDAD", "PADRE"};
        DefaultTableModel dtm = new DefaultTableModel(cabecera, 0);
        tblDatos.setModel(dtm);

        // BDInterf bd = new ConsumoLMImple();
        //List<ConsumoLM> consumoLM= bd.mostrar();
            for (int k = 0; k < 10000; k++) {
                if (vec[k][0]=="") {
                    break;
                }else{
                  Vector vector= new Vector();
            vector.add(prod);
            vector.add(vec[k][0]);
            vector.add(vec[k][1]);
            vector.add(vec[k][2]);
            dtm.addRow(vector);
                }

            }


     }

    }//GEN-LAST:event_btnConsumoActionPerformed

    private void cbProductoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_cbProductoActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_cbProductoActionPerformed


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JScrollPane JScrollPane;
    private javax.swing.JButton btnBuscar;
    private javax.swing.JButton btnConsumo;
    private javax.swing.JComboBox cbProducto;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JTable tblDatos;
```

```
    private javax.swing.JTextField txtBusqueda;
    // End of variables declaration//GEN-END:variables




    private void cargaProducto() {
       cbProducto.removeAllItems();
        BDInterf bd= new ProductoImple();
        List<Producto> producto = bd.mostrar();
        for (Producto producto1 : producto) {

            cbProducto.addItem(producto1.getDescriabrev());
        }
        cbProducto.setSelectedIndex(-1);
    }

   static void recur(String nombre){

         BDInterf bd = new ListaMaterImple();
        List<ListaMateriales> listaMateriales = bd.mostrar();
        if (i==0) {
          vec[i][0]=nombre;
          vec[i][1]="1";
          vec[i][2]="";
          i=i+1;
      }
         for (ListaMateriales listaMateriale1 : listaMateriales) {
           if (listaMateriale1.getProducto().getDescriabrev().equals(nombre)) {
            //  Prueba6.abc =Prueba6.abc +"***"+listaMateriale1.getComponente().getDescriabrev();
             vec[i][0]=listaMateriale1.getComponente().getDescriabrev();
             vec[i][2]=nombre;
              if (i>0) {
                  for (int k = i-1; k >= 0; k--) {
                      if (vec[k][0].equals(nombre)) {
                          vec[i][1]=String.valueOf(listaMateriale1.getCantidad_comp()*Double.parseDouble(vec[k][1]));
                          break;
                      }
                  }
              }

              i=i+1;
             recur(listaMateriale1.getComponente().getDescriabrev());

           }
       }
    }
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.vistas;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.List;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import pe.produccion.clases.Dimension;
import pe.produccion.clases.GanttFinal;
import pe.produccion.clases.ProduDimen;
import pe.produccion.clases.Producto;
import pe.produccion.conexion.Conexion;
import pe.produccion.dao.BDInterf;
import pe.produccion.dao.DimenImple;
import pe.produccion.dao.GanttImpleme;

/**
 *
 * @author Rooll
 */
public class GanttDialogo extends javax.swing.JDialog {

    /**
     * Creates new form GanttDialogo
     */
     public static boolean indicadorgantt=false;
```

```
private double maximo;
public GanttDialogo(java.awt.Frame parent, boolean modal,double maximo) {
    super(parent, modal);
    this.maximo=maximo;

    initComponents();
    cargatablagantt();
    cargatablaresumen();
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
private void initComponents() {

    jPanel1 = new javax.swing.JPanel();
    jScroll1 = new javax.swing.JScrollPane();
    tblGantt = new javax.swing.JTable();
    jScroll2 = new javax.swing.JScrollPane();
    tblResumen = new javax.swing.JTable();
    jPanel2 = new javax.swing.JPanel();
    btnGuardar = new javax.swing.JButton();
    btnSalir = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    setTitle("PROGHAMA DE ORDENES DE PRODUCCION POR MAQUINA");

    tblGantt.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null}
        },
        new String [] {
            "Title 1", "Title 2", "Title 3", "Title 4"
        }
    ));
    jScroll1.setViewportView(tblGantt);

    tblResumen.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null}
        },
        new String [] {
            "Title 1", "Title 2", "Title 3", "Title 4"
        }
    ));
    jScroll2.setViewportView(tblResumen);

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScroll1, javax.swing.GroupLayout.PREFERRED_SIZE, 578, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addComponent(jScroll2, javax.swing.GroupLayout.PREFERRED_SIZE, 304, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(32, Short.MAX_VALUE))
    );
    jPanel1Layout.setVerticalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jScroll1, javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addComponent(jScroll2, javax.swing.GroupLayout.PREFERRED_SIZE, 288, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(0, 12, Short.MAX_VALUE)))
            .addContainerGap())
    );

    btnGuardar.setText("GUARDAR");
    btnGuardar.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnGuardarActionPerformed(evt);
        }
    });
```

```
                btnSalir.setText("SALIR");
                btnSalir.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        btnSalirActionPerformed(evt);
                    }
                });

                javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
                jPanel2.setLayout(jPanel2Layout);
                jPanel2Layout.setHorizontalGroup(
                    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(216, 216, 216)
                        .addComponent(btnGuardar)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(btnSalir)
                        .addGap(257, 257, 257))
                );
                jPanel2Layout.setVerticalGroup(
                    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(34, 34, 34)
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(btnGuardar)
                            .addComponent(btnSalir))
                        .addContainerGap(43, Short.MAX_VALUE))
                );

                javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
                getContentPane().setLayout(layout);
                layout.setHorizontalGroup(
                    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                );
                layout.setVerticalGroup(
                    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                );

                pack();
            }// </editor-fold>//GEN-END:initComponents

            private void btnSalirActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnSalirActionPerformed
                // TODO add your handling code here:
                GanttImpleme gimple= new GanttImpleme();
                if (gimple.eliminar(maximo)) {
                    System.out.println("Eliminado");
                }
                this.setVisible(false);
            }//GEN-LAST:event_btnSalirActionPerformed

            private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnGuardarActionPerformed
                // TODO add your handling code here:
                 if (JOptionPane.showConfirmDialog(this, "Seguro de Guardar el Gantt ", "AVISO", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE)
                        ==JOptionPane.OK_OPTION) {
                 indicadorgantt=true;
                 this.setVisible(false);
                  }
            }//GEN-LAST:event_btnGuardarActionPerformed

            /**
             * @param args the command line arguments
             */
            public static void main(String args[]) {
                /* Set the Nimbus look and feel */
                //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
                /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
                 * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
                 */
                try {
                    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                        if ("Nimbus".equals(info.getName())) {
                            javax.swing.UIManager.setLookAndFeel(info.getClassName());
                            break;
                        }
                    }
                } catch (ClassNotFoundException ex) {
```

```
            java.util.logging.Logger.getLogger(GanttDialogo.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(GanttDialogo.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(GanttDialogo.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(GanttDialogo.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the dialog */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                GanttDialogo dialog = new GanttDialogo(new javax.swing.JFrame(), true,0);
                dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                    @Override
                    public void windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    }
                });
                dialog.setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnGuardar;
    private javax.swing.JButton btnSalir;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JScrollPane jScroll1;
    private javax.swing.JScrollPane jScroll2;
    private javax.swing.JTable tblGantt;
    private javax.swing.JTable tblResumen;
    // End of variables declaration//GEN-END:variables

    private void cargatablagantt() {
         String cabecera[]={"MAQUINA","OP","COMPONENTE","PADRE","TINICIO","TFINAL"};
        DefaultTableModel dtm = new DefaultTableModel(cabecera, 0);
        tblGantt.setModel(dtm);

         GanttImpleme gimple= new GanttImpleme();

        List<GanttFinal> gfinal= gimple.mostrarultimo(maximo);
        for (GanttFinal gfinal1 : gfinal) {
            Vector vec= new Vector();
            vec.add(gfinal1.getMaq().getDescriabrev());
            vec.add(gfinal1.getOp());
           vec.add(gfinal1.getComp().getDescriabrev());
           vec.add(gfinal1.getPadre().getDescriabrev());
           vec.add(gfinal1.getTinicio());
           vec.add(gfinal1.getTfinal());
           dtm.addRow(vec);
        }



    }

    private void cargatablaresumen() {
        double asig=0;
        double total=0;
        String cabecera[]={"MAQUINA","TIEMPO ASIGNADO","TIEMPO TOTAL","% OCUPACION"};
        DefaultTableModel dtm2 = new DefaultTableModel(cabecera, 0);
        tblResumen.setModel(dtm2);
         Connection cn= null;
        try {
            cn= Conexion.conectar();
            if (cn!=null) {
                String sql = "SELECT MAQ MAQUINA,SUM(TFINAL+-1*TINICIO) ASIGNADO, MAX(TFINAL) TOTAL FROM GANTTFINAL
WHERE IDPROGRAMA=? GROUP BY MAQ ORDER BY MAQ";

                PreparedStatement pstm = cn.prepareStatement(sql);
                pstm.setDouble(1, maximo);
                ResultSet rs = pstm.executeQuery();
                while (rs.next()) {
                    Vector vec= new Vector();
                    vec.add(rs.getLong("MAQUINA"));
                    vec.add(rs.getDouble("ASIGNADO"));
                    asig=asig+rs.getDouble("ASIGNADO");

                    vec.add(rs.getDouble("TOTAL"));
                    total=total+rs.getDouble("TOTAL");
                    vec.add(rs.getDouble("ASIGNADO")/rs.getDouble("TOTAL"));
                      dtm2.addRow(vec);
```

```
                }

            }

        } catch (SQLException e) {
            System.err.println("Error de conexion "+e.getMessage());
        }finally{
            if (cn!=null) {
                try {
                    cn.close();
                } catch (Exception e) {
                }
            }
        }

        Vector vec= new Vector();
        vec.add("Total");
        vec.add(asig);
        vec.add(total);
        vec.add(asig/total);
         dtm2.addRow(vec);
    }
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.vistas;

import java.util.List;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import pe.produccion.clases.Caracteristica;
import pe.produccion.clases.ClaseProducto;
import pe.produccion.clases.ListaMateriales;
import pe.produccion.clases.Maquina;
import pe.produccion.clases.Operacion;
import pe.produccion.clases.ProduProce;
import pe.produccion.clases.Producto;
import pe.produccion.clases.TipoProducto;
import pe.produccion.dao.BDInterf;
import pe.produccion.dao.CaracterImple;
import pe.produccion.dao.ClaseProdImple;
import pe.produccion.dao.ListaMaterImple;
import pe.produccion.dao.MaquinaImple;
import pe.produccion.dao.OperaImple;
import pe.produccion.dao.ProduProceImple;
import pe.produccion.dao.ProductoImple;
import pe.produccion.dao.TipoProdImple;

/**
 *
 * @author Rooll
 */
public class ListMaterView extends javax.swing.JInternalFrame {

    /**
     * Creates new form Interno1
     */
    public ListMaterView() {
        initComponents();
        cargartabla("");
        cargaComponente();
        cargaProducto();

    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        txtBusqueda = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        JScrollPane = new javax.swing.JScrollPane();
        tblDatos = new javax.swing.JTable();
        btnBuscar = new javax.swing.JButton();
        jPanel2 = new javax.swing.JPanel();
```

```
        jLabel3 = new javax.swing.JLabel();
        txtCantidad = new javax.swing.JTextField();
        btnNuevo = new javax.swing.JButton();
        btnInsertar = new javax.swing.JButton();
        btnModificar = new javax.swing.JButton();
        btnEliminar = new javax.swing.JButton();
        cbComponente = new javax.swing.JComboBox();
        jLabel5 = new javax.swing.JLabel();
        cbProducto = new javax.swing.JComboBox();
        jLabel2 = new javax.swing.JLabel();
        txtUm = new javax.swing.JTextField();
        jLabel4 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        txtUmProd = new javax.swing.JTextField();

        setClosable(true);
        setMaximizable(true);
        setResizable(true);
        setTitle("LISTA MATERIALES");

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jLabel1.setText("FILTRO");

        tblDatos.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null}
            },
            new String [] {
                "Title 1", "Title 2", "Title 3", "Title 4"
            }
        ));
        tblDatos.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                tblDatosMouseClicked(evt);
            }
        });
        JScrollPane.setViewportView(tblDatos);

        btnBuscar.setText("BUSCAR");
        btnBuscar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnBuscarActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(55, 55, 55)
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 59, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(46, 46, 46)
                        .addComponent(txtBusqueda, javax.swing.GroupLayout.PREFERRED_SIZE, 372, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(56, 56, 56)
                        .addComponent(btnBuscar, javax.swing.GroupLayout.PREFERRED_SIZE, 101, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(JScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 736, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(27, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(txtBusqueda, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1)
                    .addComponent(btnBuscar))
                .addGap(18, 18, 18)
                .addComponent(JScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 239, Short.MAX_VALUE)
                .addContainerGap())
        );

        jLabel3.setText("CANTIDAD COMP");

        btnNuevo.setText("NUEVO");
        btnNuevo.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnNuevoActionPerformed(evt);
```

```
        }
    });

    btnInsertar.setText("INSERTAR");
    btnInsertar.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnInsertarActionPerformed(evt);
        }
    });

    btnModificar.setText("MODIFICAR");
    btnModificar.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnModificarActionPerformed(evt);
        }
    });

    btnEliminar.setText("ELIMINAR");
    btnEliminar.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnEliminarActionPerformed(evt);
        }
    });

    cbComponente.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Item 1", "Item 2", "Item 3", "Item 4" }));
    cbComponente.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            cbComponenteActionPerformed(evt);
        }
    });

    jLabel5.setText("COMPONENTE");

    cbProducto.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Item 1", "Item 2", "Item 3", "Item 4" }));
    cbProducto.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            cbProductoActionPerformed(evt);
        }
    });

    jLabel2.setText("PRODUCTO");

    txtUm.setEditable(false);
    txtUm.setBackground(new java.awt.Color(204, 204, 255));

    jLabel4.setText("U.M. COMP");

    jLabel6.setText("U.M. PROD");

    txtUmProd.setEditable(false);
    txtUmProd.setBackground(new java.awt.Color(204, 204, 255));

    javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
    jPanel2.setLayout(jPanel2Layout);
    jPanel2Layout.setHorizontalGroup(
        jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addComponent(jLabel2)
                .addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                    .addComponent(jLabel6, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel4, javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 70,
Short.MAX_VALUE)))
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(txtUmProd, javax.swing.GroupLayout.PREFERRED_SIZE, 134, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(txtUm, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 272,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(cbProducto, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE,
272, javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addGroup(jPanel2Layout.createSequentialGroup()
                    .addGap(32, 32, 32)
                    .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(txtCantidad, javax.swing.GroupLayout.PREFERRED_SIZE, 272,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(cbComponente, javax.swing.GroupLayout.PREFERRED_SIZE, 272,
javax.swing.GroupLayout.PREFERRED_SIZE))))
```

```
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,                javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(btnInsertar,          javax.swing.GroupLayout.DEFAULT_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(btnNuevo,          javax.swing.GroupLayout.DEFAULT_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(btnModificar,          javax.swing.GroupLayout.DEFAULT_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(btnEliminar,          javax.swing.GroupLayout.DEFAULT_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                        .addGap(117, 117, 117))
                );
                jPanel2Layout.setVerticalGroup(
                    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel2Layout.createSequentialGroup()
                                .addGap(21, 21, 21)
                                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(btnNuevo)
                                    .addComponent(jLabel5)))
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
                                .addContainerGap()
                                .addComponent(cbComponente,      javax.swing.GroupLayout.PREFERRED_SIZE,      javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel2Layout.createSequentialGroup()
                                .addGap(18, 18, 18)
                                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(txtCantidad,      javax.swing.GroupLayout.PREFERRED_SIZE,      javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabel3)))
                            .addGroup(jPanel2Layout.createSequentialGroup()
                                .addGap(27, 27, 27)
                                .addComponent(btnInsertar)))
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel2Layout.createSequentialGroup()
                                .addGap(24, 24, 24)
                                .addComponent(btnModificar)
                                .addGap(31, 31, 31)
                                .addComponent(btnEliminar))
                            .addGroup(jPanel2Layout.createSequentialGroup()
                                .addGap(13, 13, 13)
                                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jLabel4)
                                    .addComponent(txtUm,          javax.swing.GroupLayout.PREFERRED_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGap(23, 23, 23)
                                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jLabel2)
                                    .addComponent(cbProducto,      javax.swing.GroupLayout.PREFERRED_SIZE,      javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGap(21, 21, 21)
                                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jLabel6)
                                    .addComponent(txtUmProd,      javax.swing.GroupLayout.PREFERRED_SIZE,      javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                        .addContainerGap(24, Short.MAX_VALUE))
                );

                javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
                getContentPane().setLayout(layout);
                layout.setHorizontalGroup(
                    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                );
                layout.setVerticalGroup(
                    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jPanel1,          javax.swing.GroupLayout.PREFERRED_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jPanel2,          javax.swing.GroupLayout.PREFERRED_SIZE,          javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addContainerGap(28, Short.MAX_VALUE))
                );

                pack();
            }// </editor-fold>//GEN-END:initComponents

            private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnBuscarActionPerformed
                // TODO add your handling code here:
                String n= txtBusqueda.getText();
                cargartabla(n);
```

```
    }//GEN-LAST:event_btnBuscarActionPerformed

    private void tblDatosMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_tblDatosMouseClicked
        // TODO add your handling code here:
        int fila=tblDatos.getSelectedRow();
        cbComponente.setSelectedItem(tblDatos.getValueAt(fila, 0).toString());
        txtCantidad.setText(tblDatos.getValueAt(fila, 1).toString());

         cbProducto.setSelectedItem(tblDatos.getValueAt(fila, 2).toString());



    }//GEN-LAST:event_tblDatosMouseClicked

    private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnNuevoActionPerformed
        // TODO add your handling code here:
        cbComponente.setSelectedIndex(-1);
        txtCantidad.setText("");

        cbProducto.setSelectedIndex(-1);
        txtUm.setText("");
        txtUmProd.setText("");
    }//GEN-LAST:event_btnNuevoActionPerformed

    private void btnInsertarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnInsertarActionPerformed
        // TODO add your handling code here:



      if (!validar()) {
        return;
      }
       if (JOptionPane.showConfirmDialog(this, "Seguro de Insertar", "AVISO", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE)
              ==JOptionPane.OK_OPTION) {
           double canti= Double.valueOf(txtCantidad.getText().trim());

           String compon= cbComponente.getSelectedItem().toString();

            String produc= cbProducto.getSelectedItem().toString();

           BDInterf bd = new ListaMaterImple();
           ListaMateriales listaMateriales= new ListaMateriales();
           listaMateriales.setCantidad_comp(canti);


           BDInterf<Producto> bdprod= new ProductoImple();
           Producto componente= bdprod.encuentraID(compon);

           BDInterf<Producto> bdprod1= new ProductoImple();
           Producto producto= bdprod1.encuentraID(produc);

           listaMateriales.setComponente(componente);
           listaMateriales.setProducto(producto);



             if (bd.insertar(listaMateriales)) {

                JOptionPane.showMessageDialog(this, "Insercion Exitosa", "AVISO", JOptionPane.INFORMATION_MESSAGE);
                cargartabla("");
              }else{
                JOptionPane.showMessageDialog(this, "Error de Insercion-Ya existe la combinacion Componente-Producto", "AVISO",
JOptionPane.ERROR_MESSAGE);
              }

       }

    }//GEN-LAST:event_btnInsertarActionPerformed

    private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnModificarActionPerformed
        // TODO add your handling code here:


       if (cbComponente.getSelectedIndex()==-1) {
         JOptionPane.showMessageDialog(this, "Ingrese el Componente","ERROR" , JOptionPane.ERROR_MESSAGE);
              return ;
       }

       if (txtCantidad.getText().trim().length()==0) {
         JOptionPane.showMessageDialog(this, "Ingrese la cantidad","ERROR" , JOptionPane.ERROR_MESSAGE);
        return;
       }
```

```
                if (cbProducto.getSelectedIndex()==-1) {
                 JOptionPane.showMessageDialog(this, "Ingrese el Producto","ERROR" , JOptionPane.ERROR_MESSAGE);
                      return;
                }

                if (JOptionPane.showConfirmDialog(this, "Seguro de Modificar", "AVISO", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE)
                    ==JOptionPane.OK_OPTION) {
                  double canti= Double.valueOf(txtCantidad.getText().trim());

                  String compon= cbComponente.getSelectedItem().toString();

                   String produc= cbProducto.getSelectedItem().toString();

                  BDInterf bd = new ListaMaterImple();
                  ListaMateriales listaMateriales= new ListaMateriales();
                  listaMateriales.setCantidad_comp(canti);


                  BDInterf<Producto> bdprod= new ProductoImple();
                  Producto componente= bdprod.encuentraID(compon);

                  BDInterf<Producto> bdprod1= new ProductoImple();
                  Producto producto= bdprod1.encuentraID(produc);

                  listaMateriales.setComponente(componente);
                  listaMateriales.setProducto(producto);

                    if (bd.actualizar(listaMateriales)) {
                      JOptionPane.showMessageDialog(this, "Modificacion Exitosa", "AVISO", JOptionPane.INFORMATION_MESSAGE);
                     cargartabla("");
                    }else{
                      JOptionPane.showMessageDialog(this, "Error de Modificacion", "AVISO", JOptionPane.ERROR_MESSAGE);
                    }

                }

            }//GEN-LAST:event_btnModificarActionPerformed

            private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnEliminarActionPerformed
                // TODO add your handling code here:

                if (cbProducto.getSelectedIndex()==-1) {

                      return;
                }
                if (cbComponente.getSelectedIndex()==-1) {

                      return;
                }


                if (JOptionPane.showConfirmDialog(this, "Seguro de Eliminar el Producto "+cbProducto.getSelectedItem().toString()+
                     " con su componente "+cbComponente.getSelectedItem().toString()
                     , "AVISO", JOptionPane.OK_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE)
                     ==JOptionPane.OK_OPTION) {

                   String comp= cbComponente.getSelectedItem().toString();
                    String prod= cbProducto.getSelectedItem().toString();
                    BDInterf bd = new ListaMaterImple();
                    ListaMateriales listaMateriales= new ListaMateriales();

                     BDInterf<Producto> bdprod= new ProductoImple();
                    Producto producto= bdprod.encuentraID(comp);
                    Producto producto1= bdprod.encuentraID(prod);

                    listaMateriales.setProducto(producto1);
                    listaMateriales.setComponente(producto);
                      if (bd.eliminar(listaMateriales)) {
                          JOptionPane.showMessageDialog(this, "Eliminacion Exitosa", "AVISO", JOptionPane.INFORMATION_MESSAGE);

                          cbComponente.setSelectedIndex(-1);
                           txtCantidad.setText("");

                           cbProducto.setSelectedIndex(-1);
                           txtUm.setText("");
                           cargartabla("");
                         }else{
                           JOptionPane.showMessageDialog(this, "Error de Eliminacion", "AVISO", JOptionPane.ERROR_MESSAGE);
                         }

                }
            }//GEN-LAST:event_btnEliminarActionPerformed
```

```
private void cbComponenteActionPerformed(java.awt.event.ActionEvent evt) {///GEN-FIRST:event_cbComponenteActionPerformed
    // TODO add your handling code here:

     if (cbComponente.getSelectedIndex()>=0) {

        String prod = cbComponente.getSelectedItem().toString();
        BDInterf<Producto> bdprod= new ProductoImple();
        Producto producto = bdprod.encuentraID(prod);
        Producto producto1=bdprod.mostrarId(producto);

   String umed= producto1.getUnidmed().getDescriabrev();
   txtUm.setText(umed);

      //cargatablamp("",0);
      }
}//GEN-LAST:event_cbComponenteActionPerformed

private void cbProductoActionPerformed(java.awt.event.ActionEvent evt) {///GEN-FIRST:event_cbProductoActionPerformed
    // TODO add your handling code here:
     if (cbProducto.getSelectedIndex()>=0) {

        String prod = cbProducto.getSelectedItem().toString();
        BDInterf<Producto> bdprod= new ProductoImple();
        Producto producto = bdprod.encuentraID(prod);
        Producto producto1=bdprod.mostrarId(producto);

   String umed= producto1.getUnidmed().getDescriabrev();
   txtUmProd.setText(umed);

      //cargatablamp("",0);
      }
}//GEN-LAST:event_cbProductoActionPerformed


// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JScrollPane JScrollPane;
private javax.swing.JButton btnBuscar;
private javax.swing.JButton btnEliminar;
private javax.swing.JButton btnInsertar;
private javax.swing.JButton btnModificar;
private javax.swing.JButton btnNuevo;
private javax.swing.JComboBox cbComponente;
private javax.swing.JComboBox cbProducto;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JTable tblDatos;
private javax.swing.JTextField txtBusqueda;
private javax.swing.JTextField txtCantidad;
private javax.swing.JTextField txtUm;
private javax.swing.JTextField txtUmProd;
// End of variables declaration//GEN-END:variables

private void cargartabla(String nombre) {
    String cabecera[]={"COMPONENTE","CANTIDAD","PRODUCTO"};
    DefaultTableModel dtm = new DefaultTableModel(cabecera, 0);
    tblDatos.setModel(dtm);
    BDInterf bd = new  ListaMaterImple();
    List<ListaMateriales> listaMateriales= bd.mostrarCriteriolike(nombre);
    for (ListaMateriales listaMateriales1 : listaMateriales) {
       Vector vec= new Vector();
       vec.add(listaMateriales1.getComponente().getDescriabrev());
      vec.add(listaMateriales1.getCantidad_comp());

      vec.add(listaMateriales1.getProducto().getDescriabrev());

       dtm.addRow(vec);
        }
}
private boolean validar(){

    if (cbComponente.getSelectedIndex()==-1) {
        JOptionPane.showMessageDialog(this, "Ingrese el Producto IN","ERROR" , JOptionPane.ERROR_MESSAGE);
             return false;
    }

     if (txtCantidad.getText().trim().length()==0) {
        JOptionPane.showMessageDialog(this, "Ingrese la cantidad IN","ERROR" , JOptionPane.ERROR_MESSAGE);
       return false;
    }
```

```
        if (cbProducto.getSelectedIndex()==-1) {
         JOptionPane.showMessageDialog(this, "Ingrese el Producto OUT","ERROR" , JOptionPane.ERROR_MESSAGE);
              return false;
        }


        return true;
    }



    private void cargaComponente() {
         cbComponente.removeAllItems();
        BDInterf bd= new ProductoImple();
        List<Producto> producto = bd.mostrar();
        for (Producto producto1 : producto) {

            cbComponente.addItem(producto1.getDescriabrev());
        }
        cbComponente.setSelectedIndex(-1);
        txtUm.setText("");
    }

    private void cargaProducto() {
        cbProducto.removeAllItems();
        BDInterf bd= new ProductoImple();
        List<Producto> producto = bd.mostrar();
        for (Producto producto1 : producto) {

            cbProducto.addItem(producto1.getDescriabrev());
        }
        cbProducto.setSelectedIndex(-1);
        txtUmProd.setText("");
    }


}


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pe.produccion.vistas;

import java.util.List;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import pe.produccion.clases.Caracteristica;
import pe.produccion.clases.Maquina;
import pe.produccion.dao.BDInterf;
import pe.produccion.dao.CaracterImple;
import pe.produccion.dao.MaquinaImple;

/**
 *
 * @author Rooll
 */
public class MaqView extends javax.swing.JInternalFrame {

    /**
     * Creates new form Interno1
     */
    public MaqView() {
        initComponents();
        cargartabla("");
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        txtBusqueda = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        JScrollPane = new javax.swing.JScrollPane();
        tblDatos = new javax.swing.JTable();
```

```
        btnBuscar = new javax.swing.JButton();
        jPanel2 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        txtId = new javax.swing.JTextField();
        txtNomAbrev = new javax.swing.JTextField();
        txtNomExten = new javax.swing.JTextField();
        btnNuevo = new javax.swing.JButton();
        btnInsertar = new javax.swing.JButton();
        btnModificar = new javax.swing.JButton();
        btnEliminar = new javax.swing.JButton();

        setClosable(true);
        setMaximizable(true);
        setResizable(true);
        setTitle("INGRESO MAQUINA");

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jLabel1.setText("FILTRO");

        tblDatos.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null}
            },
            new String [] {
                "Title 1", "Title 2", "Title 3", "Title 4"
            }
        ));
        tblDatos.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                tblDatosMouseClicked(evt);
            }
        });
        JScrollPane.setViewportView(tblDatos);

        btnBuscar.setText("BUSCAR");
        btnBuscar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnBuscarActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(55, 55, 55)
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 59, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(46, 46, 46)
                        .addComponent(txtBusqueda, javax.swing.GroupLayout.PREFERRED_SIZE, 372, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(56, 56, 56)
                        .addComponent(btnBuscar, javax.swing.GroupLayout.PREFERRED_SIZE, 101, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(JScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 736, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(27, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(txtBusqueda, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1)
                    .addComponent(btnBuscar))
                .addGap(18, 18, 18)
                .addComponent(JScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 239, Short.MAX_VALUE)
                .addContainerGap())
        );

        jLabel2.setText("ID");

        jLabel3.setText("NOMBRE");

        jLabel4.setText("NOMBRE EXTENDIDO");

        txtId.setEditable(false);
        txtId.setBackground(new java.awt.Color(102, 204, 255));
```

```
        btnNuevo.setText("NUEVO");
        btnNuevo.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnNuevoActionPerformed(evt);
            }
        });

        btnInsertar.setText("INSERTAR");
        btnInsertar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnInsertarActionPerformed(evt);
            }
        });

        btnModificar.setText("MODIFICAR");
        btnModificar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnModificarActionPerformed(evt);
            }
        });

        btnEliminar.setText("ELIMINAR");
        btnEliminar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnEliminarActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel3)
                    .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 42, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 122, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(36, 36, 36)
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(txtNomAbrev, javax.swing.GroupLayout.PREFERRED_SIZE, 229,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(txtId, javax.swing.GroupLayout.PREFERRED_SIZE, 85, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(btnInsertar, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(btnNuevo, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(btnModificar, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(btnEliminar, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                        .addGap(117, 117, 117))
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addComponent(txtNomExten, javax.swing.GroupLayout.PREFERRED_SIZE, 325, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(21, 21, 21)
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel2)
                            .addComponent(btnNuevo)))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(txtId, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(18, 18, 18)
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel3)
                            .addComponent(txtNomAbrev, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(27, 27, 27)
                        .addComponent(btnInsertar)))
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
                        .addGroup(jPanel2Layout.createSequentialGroup()
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel4)
                                .addComponent(txtNomExten, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGroup(jPanel2Layout.createSequentialGroup()
                            .addGap(24, 24, 24)
                            .addComponent(btnModificar)))
                    .addGap(31, 31, 31)
                    .addComponent(btnEliminar)
                    .addContainerGap(37, Short.MAX_VALUE))
                );

                javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
                getContentPane().setLayout(layout);
                layout.setHorizontalGroup(
                    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addContainerGap())
                );
                layout.setVerticalGroup(
                    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                );

                pack();
            }// </editor-fold>//GEN-END:initComponents

            private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnBuscarActionPerformed
                // TODO add your handling code here:
                String n= txtBusqueda.getText();
                cargartabla(n);

            }//GEN-LAST:event_btnBuscarActionPerformed

            private void tblDatosMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_tblDatosMouseClicked
                // TODO add your handling code here:
                int fila=tblDatos.getSelectedRow();
                txtId.setText(tblDatos.getValueAt(fila, 0).toString());
                 txtNomAbrev.setText(tblDatos.getValueAt(fila, 1).toString());
                  txtNomExten.setText(tblDatos.getValueAt(fila, 2).toString());

            }//GEN-LAST:event_tblDatosMouseClicked

            private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnNuevoActionPerformed
                // TODO add your handling code here:
                txtId.setText("");
                txtNomAbrev.setText("");
                txtNomExten.setText("");
            }//GEN-LAST:event_btnNuevoActionPerformed

            private void btnInsertarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnInsertarActionPerformed
                // TODO add your handling code here:
                String nombre= txtNomAbrev.getText().trim();
                String nombreex= txtNomExten.getText().trim();


              if (!validar()) {
                return;
              }
               if (JOptionPane.showConfirmDialog(this, "Seguro de Insertar", "AVISO", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE)
                    ==JOptionPane.OK_OPTION) {
                  BDInterf bd = new MaquinaImple();
                  Maquina maquina= new Maquina();
                  maquina.setDescriabrev(nombre);
                  maquina.setDescriexten(nombreex);

                   if (bd.insertar(maquina)) {
                      txtId.setText(String.valueOf(maquina.getId()));
                      JOptionPane.showMessageDialog(this, "Insercion Exitosa", "AVISO", JOptionPane.INFORMATION_MESSAGE);
                      cargartabla("");
                   }else{
                      JOptionPane.showMessageDialog(this, "Error de Insercion", "AVISO", JOptionPane.ERROR_MESSAGE);
                  }
```

```
        }

    }//GEN-LAST:event_btnInsertarActionPerformed

    private void btnModificarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnModificarActionPerformed
        // TODO add your handling code here:
        String id= txtId.getText();
        String nombre= txtNomAbrev.getText().trim();
        String nombreex= txtNomExten.getText().trim();

        if (id.trim().length()==0) {
            return;
        }
         if (txtNomAbrev.getText().trim().length()==0) {
            JOptionPane.showMessageDialog(this, "Ingrese el Nombre","ERROR" , JOptionPane.ERROR_MESSAGE);
           return;
        }else{
             BDInterf<Maquina> bd = new MaquinaImple();
             Maquina maquina = bd.encuentraID(nombre);
             if ((maquina.getId()!= 0) && (maquina.getId()!= Long.valueOf(id)) ) {
                 JOptionPane.showMessageDialog(this, "Nombre ya existe", "AVISO", JOptionPane.ERROR_MESSAGE);
                return ;
             }
         }
         if (txtNomExten.getText().trim().length()==0) {
                 JOptionPane.showMessageDialog(this, "Ingrese el Nombre Extendido","ERROR" , JOptionPane.ERROR_MESSAGE);
                 return;
              }


        if (JOptionPane.showConfirmDialog(this, "Seguro de Modificar", "AVISO", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE)
            ==JOptionPane.OK_OPTION) {
           BDInterf bd = new MaquinaImple();
           Maquina maquina= new Maquina();
           maquina.setId(Long.valueOf(id));
           maquina.setDescriabrev(nombre);
           maquina.setDescriexten(nombreex);

            if (bd.actualizar(maquina)) {
               JOptionPane.showMessageDialog(this, "Modificacion Exitosa", "AVISO", JOptionPane.INFORMATION_MESSAGE);
              cargartabla("");
             }else{
               JOptionPane.showMessageDialog(this, "Error de Modificacion", "AVISO", JOptionPane.ERROR_MESSAGE);
             }

        }


    }//GEN-LAST:event_btnModificarActionPerformed

    private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnEliminarActionPerformed
        // TODO add your handling code here:
         String id= txtId.getText();


        if (id.trim().length()==0) {
            return;
        }


        if (JOptionPane.showConfirmDialog(this, "Seguro de Eliminar "+id, "AVISO", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE)
            ==JOptionPane.OK_OPTION) {
           BDInterf bd = new MaquinaImple();
           Maquina maquina= new Maquina();
           maquina.setId(Long.valueOf(id));
            if (bd.eliminar(maquina)) {
                JOptionPane.showMessageDialog(this, "Eliminacion Exitosa", "AVISO", JOptionPane.INFORMATION_MESSAGE);
              txtId.setText("");
                txtNomAbrev.setText("");
                txtNomExten.setText("");
                cargartabla("");
             }else{
               JOptionPane.showMessageDialog(this, "Error de Eliminacion", "AVISO", JOptionPane.ERROR_MESSAGE);
             }

        }
    }//GEN-LAST:event_btnEliminarActionPerformed


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JScrollPane JScrollPane;
    private javax.swing.JButton btnBuscar;
    private javax.swing.JButton btnEliminar;
```

```
    private javax.swing.JButton btnInsertar;
    private javax.swing.JButton btnModificar;
    private javax.swing.JButton btnNuevo;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JTable tblDatos;
    private javax.swing.JTextField txtBusqueda;
    private javax.swing.JTextField txtId;
    private javax.swing.JTextField txtNomAbrev;
    private javax.swing.JTextField txtNomExten;
    // End of variables declaration//GEN-END:variables

    private void cargartabla(String nombre) {
        String cabecera[]={"ID","DESCRIPCION","DESCR.EXTENDIDA"};
        DefaultTableModel dtm = new DefaultTableModel(cabecera, 0);
        tblDatos.setModel(dtm);

         BDInterf bd = new MaquinaImple();
        List<Maquina> maquina= bd.mostrarCriterio(nombre);
        for (Maquina maquina1 : maquina) {
            Vector vec= new Vector();
            vec.add(maquina1.getId());
            vec.add(maquina1.getDescriabrev());
           vec.add(maquina1.getDescriexten());
           dtm.addRow(vec);
            }
    }
    private boolean validar(){

        if (txtNomAbrev.getText().trim().length()==0) {
            JOptionPane.showMessageDialog(this, "Ingrese el Nombre","ERROR" , JOptionPane.ERROR_MESSAGE);
          return false;
        }else{
            String nombre= txtNomAbrev.getText().trim();
            BDInterf bd = new MaquinaImple();
            if (!bd.validar(nombre)) {
                JOptionPane.showMessageDialog(this, "Nombre ya existe", "AVISO", JOptionPane.ERROR_MESSAGE);
                return false;
            }
        }

        if (txtNomExten.getText().trim().length()==0) {
                JOptionPane.showMessageDialog(this, "Ingrese el Nombre Extendido","ERROR" , JOptionPane.ERROR_MESSAGE);
                return false;
              }
        return true;
    }
}
```