**FACULTAD DE INGENIERÍA GEOGRÁFICA, AMBIENTAL Y ECOTURISMO**

# AUTOMATIZACIÓN DEL PROCESO DE VALIDACIÓN Y CARGA DE REDES DE GAS EN LIMA Y CALLAO

**Línea de investigación:**

**Desarrollo urbano-rural, catastro, prevención de riesgos, hidráulica y geotecnia**

Trabajo de suficiencia profesional para optar el título profesional de Ingeniero Geógrafo

**Autor:**

Vasquez Milla, Jaime

**Asesor:**

Sánchez Paredes, César Alberto

(ORCID: 0000-0003-1136-5403)

**Jurado:**

Alva Velasquez, Miguel

Sernaqué Auccahuasi, Fernando Antonio

Paricoto Simón, María Mercedes

**Lima - Perú**

**2023**

# AUTOMATIZACIÓN DEL PROCESO DE VALIDACIÓN Y CARGA DE REDES DE GAS EN LIMA Y CALLAO

INFORME DE ORIGINALIDAD

| 8% | 7% | 1% | 2% |
|---|---|---|---|
| INDICE DE SIMILITUD | FUENTES DE INTERNET | PUBLICACIONES | TRABAJOS DEL ESTUDIANTE |

FUENTES PRIMARIAS

| | | |
|---|---|---|
| 1 | repositorio.unfv.edu.pe<br>Fuente de Internet | 1% |
| 2 | Submitted to Universidad San Ignacio de Loyola<br>Trabajo del estudiante | 1% |
| 3 | pingpdf.com<br>Fuente de Internet | <1% |
| 4 | www.coursehero.com<br>Fuente de Internet | <1% |
| 5 | es.readkong.com<br>Fuente de Internet | <1% |
| 6 | empiezoinformatica.wordpress.com<br>Fuente de Internet | <1% |
| 7 | issuu.com<br>Fuente de Internet | <1% |
| 8 | cdn.www.gob.pe<br>Fuente de Internet | <1% |

FACULTAD DE INGENIERÍA GEOGRÁFICA, AMBIENTAL Y ECOTURISMO

# AUTOMATIZACIÓN DEL PROCESO DE VALIDACIÓN Y CARGA DE REDES DE GAS EN LIMA Y CALLAO

Línea de Investigación: Desarrollo Urbano-Rural, Catastro, Prevención de Riesgos, Hidráulica y Geotécnica

Informe de suficiencia profesional para optar el título profesional de Ingeniero Geógrafo

**Autor:**

Vasquez Milla Jaime

**Asesor:**

Sánchez Paredes, César Alberto

ORCID: 0000-0003-1136-5403

**Jurado:**

Alva Velasquez, Miguel

Sernaqué Auccahuasi, Fernando Antonio

Paricoto Simón, María Mercedes

Lima – Peru

2023

Índice de Contenido

## Índice de Figuras

Índice de Tablas

**Resumen**

La empresa concesionaria de la distribución de gas natural en Lima y Callao tiene la necesidad de agregar las nuevas redes de gas construidas a su base de datos geoespacial, a fin de gestionar las conexiones instaladas, mantener los procesos de venta y cumplir con los plazos establecidos por las instituciones reguladoras. El problema radica en la cantidad de información que se genera diariamente, producto de la habilitación de siete kilómetros de red por día como media; dichos volúmenes de datos dificultan su validación de acuerdo con los estándares de calidad y tiempos requeridos. En base a esta necesidad, el proyecto tiene el objetivo principal de automatizar los procesos de validación y carga de las nuevas redes a la base de datos geoespacial, mediante la creación de herramientas personalizadas. Para cumplir con ello, se usó las ventajas que brinda el software ArcGIS y el lenguaje de programación Python, a fin de aprovechar ambas tecnologías y en base a ellos crear scripts personalizados de uso recurrente que ayuden con las tareas que involucran la carga de redes. Como resultado del proyecto, se crearon 18 herramientas que automatizaron el 80% de tareas que se requieren para la validación de las nuevas redes, además producto de la automatización se redujeron los recursos de personal, hardware y software destinados al cumplimiento de todas las actividades y finalmente, pero no menos importante se cumple con los tiempos estipulados. En conclusión, en base a los resultados de los scripts creados, esta iniciativa ha demostrado ser eficiente para validar las nuevas redes de gas creadas, mejorando los tiempos y estándares de calidad requeridos.

Palabras clave: *sistemas de información geográfica, redes de gas, arcpy, arcmap*

**Abstrac**

The concessionary company responsible for the distribution of natural gas in Lima and Callao needs to add the newly constructed gas networks to its geospatial database, in order to manage the installed connections, maintain sales processes, and comply with the deadlines set by regulatory institutions. The problem lies in the amount of information generated daily, resulting from the activation of an average of seven kilometers of network per day; such volumes of data make it difficult to validate them according to quality standards and required times. Based on this need, the main objective of the project is to automate the validation processes and loading of the new networks into the geospatial database, by creating customized tools. To achieve this, the advantages offered by ArcGIS software and the Python programming language were used, in order to take advantage of both technologies and create custom, recurrent use scripts that assist with tasks involving network loading. As a result of the project, 18 tools were created that automated 80% of the tasks required for the validation of the new networks, and as a result of automation, the resources for personnel, hardware, and software allocated to the fulfillment of all activities were reduced, and last but not least, the stipulated times were met. In conclusion, based on the results of the scripts created, this initiative has proven to be efficient in validating the newly created gas networks, improving the times and quality standards required.

# I. Introducción

El presente informe describe el proceso de automatización de validación y carga de nuevas redes a la base de datos de la empresa concesionaría de gas natural en Lima Metropolitana. Poniendo énfasis en la creación de algoritmos computacionales basados en las librerías que brinda el software ArcGIS en combinación con el lenguaje de programación Python.

Para gestionar eficazmente las redes de gas, la empresa ha implementado una base de datos geoespacial, facilitando así el manejo de la información necesaria para cumplir con los mandatos de las entidades reguladoras. Dado que la incorporación de nuevas redes, que promedia los siete kilómetros diarios, genera un volumen considerable de datos, lo que representa un desafío significativo para mantener la calidad y eficiencia requeridas en la actualización de la base de datos geoespacial. Por consiguiente, se ha vuelto imprescindible la automatización de los procesos de validación y carga de las redes de gas para satisfacer estas exigencias operativas.

En ese sentido, el objetivo principal del proyecto es automatizar el proceso de validación y carga de información de las nuevas redes a la base de datos espacial, mediante la creación de herramientas personalizadas usando el software ArcGIS en combinación del lenguaje de programación Python. La automatización del proceso debe asegurar la calidad de los datos subidos a la base de datos espacial, respetando las directrices del cliente y además de cumplir con los tiempos estipulados. Se debe contemplar la optimización de los recursos destinados para el cumplimiento de las actividades de carga, dentro de los recursos a optimizar son los del personal técnico, hardware y software.

Las nuevas redes vienen en distintos formatos, siendo los más comunes archivos geodatabase no esquematizados, dicha información debe ser convertida a los esquemas requeridos por las reglas de Network Analyst utilizados por el cliente, después debe heredar

datos de diversas fuentes, entre ellos datos espaciales de ubicación y datos alfanuméricos relacionados con la construcción, para ser finalmente cargado al sistema de redes de gas del cliente. Para desarrollar todas estas tareas, fue necesario crear herramientas de doble propósito, el primero es garantizar la calidad de los datos y el segundo es reducir recursos para el desarrollo del trabajo.

Los resultados obtenidos del proyecto son la creación de 18 herramientas personalizadas que cubren el 80% del proceso de validación de las nuevas redes, así como el cumplimento de la calidad de datos subidos y estar dentro de los márgenes de tiempo exigidos por el cliente. Además, las herramientas creadas ayudaron a reducir los recursos destinados al cumplimiento de las tareas necesarias. Otro indicador de los buenos resultados es la obtención de nuevos servicios donde la automatización de procesos es el principal actor para tener precios competitivos.

## 1.1 Trayectoria del autor

Bachiller desde el año 2012 en la carrera de Ingeniería Geográfica de la Facultad de Ingeniería Geográfica Ambiental y Ecoturismo. Además, además tiene el título de Máster en Sistemas de Información Geográfica otorgado por la Universidad Politécnica de Cataluña de España el año 2020.

En el ámbito laboral, mi experiencia se basa en el uso de las herramientas de los Sistemas de Información Geográfica para realizar análisis espacial en entornos urbanos y administración de información de flujos de gas domiciliario.

**Ministerio de Cultura**

Análisis y revisión de la planimetría de obras civiles de proyectos de inversión pública y privada con el fin de verificar la no afectación de sitios arqueológicos a nivel nacional. Además de la creación y administración de la base de datos espacial de los sitios arqueológicos del Ministerio de Cultura.

**Proyecto Barrio Mio**

Participación en la elaboración y organización de una base de datos espacial de los barrios Huaycán y Mariscales Cáceres en los distritos de Ate y San Juan de Lurigancho, respectivamente, con el objetivo de analizar las dinámicas espaciales que suceden en los barrios referidos. El conocimiento generado sirvió de base para la planificación urbana de dichos lugares con un enfoque de la gestión de los peligros y vulnerabilidades presentes.

**Proyecto RIMACDRR**

En conjunto con las ONG CARE, COOPI y la Municipalidad Distrital del Rímac, se participó en la elaboración y organización de una base de datos espacial a partir de información recopilada de instituciones estatales, privadas, organismos no gubernamentales y del territorio. Posteriormente, se analizó y gestionó los datos con un enfoque centrado en el barrio con el propósito de mejorar la gestión urbana y la gestión de riesgos en el distrito del Rímac. En el

marco del proyecto, el autor del presente informe logró llevar con éxito la Meta 16 del Programa de Incentivos para la Mejora de la gestión Municipal del año 2016 Organizada por el Ministerio de Vivienda.

**Banco Mundial y Banco Interamericano de Desarrollo, Project Perú Lima Line 2**

Se asistió técnicamente como analista en Sistemas de Información Geográfica al equipo de investigación en la automatización de la geolocalización de base de datos de inmuebles y de las encuestas de origen destino del área metropolitana de Lima a fin de crear un plan de movilidad general de la ciudad.

**ITIC Perú**

Se lidera los proyectos SIG referente a la administración de datos espaciales relacionados con las redes de gas usando herramientas de los Sistemas de Información Geográfica para el cliente CALIDDA. Las tareas principales dentro los proyectos de gas es la automatización de procesos y creación de herramientas personalizadas usando librerías de automatización como ArcPy de ArcGIS, base de datos de Microsoft SQL Server y PostGIS con el objetivo de la administración y carga de redes internas y externas a la base de datos espacial de CALIDDA.

## 1.2 Descripción de la empresa

ITIC PERU es una empresa que se enfoca en aspectos sociales y ambientales, contando con profesionales altamente capacitados en técnicas de estándares internacionales. Estos expertos están comprometidos con los valores empresariales de la compañía, buscando constantemente la excelencia operativa y la integridad en la prestación de sus servicios. Inició sus operaciones hace trece años como una filial de TÜV Rheinland y en el año 2023 se consolidó como una empresa de origen nacional, expandiendo su presencia a diversos países de Suramérica y Centroamérica. La empresa aprecia y prioriza la transformación digital como una herramienta fundamental para alcanzar sus metas, permitiéndoles ser más eficaces,

innovadores y competitivos en un entorno global cada vez más interconectado. A través de la automatización de procesos, la adopción de tecnologías de vanguardia y el aprovechamiento de datos, están elevando la calidad de la experiencia de sus clientes y fortaleciendo su capacidad para tomar decisiones informadas.

La misión es proporcionar servicios de inspección, certificación y ensayos con los más altos estándares de calidad para garantizar la seguridad, calidad y cumplimiento de las normas en diversos sectores, transformando la industria, brindando agilidad, protección y valor agregado a sus clientes.

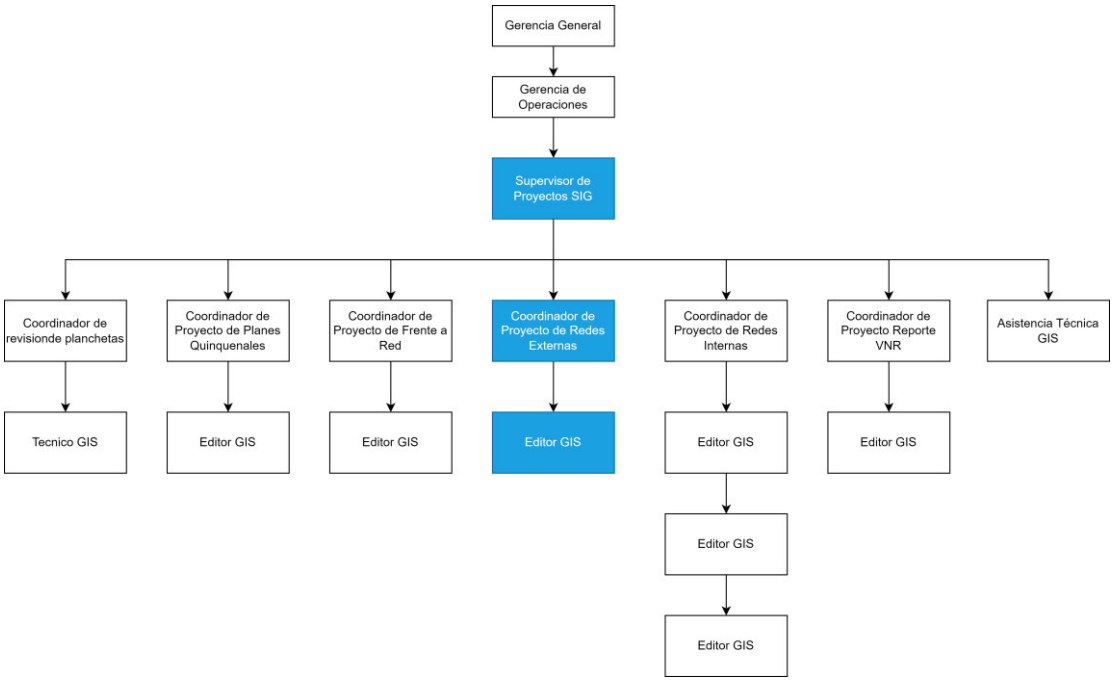
Como visión para el 2025, ser reconocidos como transformadores de la industria de certificación, inspección y ensayos a través de la innovación y el desarrollo de tecnologías avanzadas para brindar soluciones personalizadas y agregar valor a sus clientes, siendo una empresa ética y socialmente responsable que contribuye al desarrollo sostenible de la región. (ITICPERU, 2023)

## 1.3 Organigrama de la empresa

La estructura organizativa de la empresa, en relación con el proyecto específico en cuestión, está delineada en el organigrama mostrado en la figura uno, donde el papel del autor de este informe cumple el rol de supervisor de proyectos GIS. Esta posición actúa como un pivote central para la supervisión y el manejo de todas las actividades de la organización relacionadas a redes de gas. Bajo la Gerencia General y la Gerencia de Operaciones, como supervisor de proyectos GIS se coordina una serie de roles especializados, cada uno enfocado en un aspecto clave de la gestión de las redes de gas. Estos incluyen la coordinación de proyectos de revisión de actas de redes, planificación a largo plazo con planes quinquenales, gestión de proyectos de gas tanto internas como externas, y la elaboración de reportes críticos como el Reporte VNR.

**Figura 1**

*Organigrama de la empresa*




Por políticas de privacidad no se puede mostrar el organigrama de los proyectos no relacionados.

## 1.4 Áreas y funciones desempeñadas

Dentro de la empresa la función principal es la de supervisión del correcto funcionamiento de los proyectos SIG vigentes, brindando asistencia técnica en la automatización de procesos y coordinación de actividades entre los diferentes proyectos al equipo interno de la empresa como al cliente.

En cuanto a la supervisión el autor de este informe es responsable de todas las actividades y productos frente al cliente, supervisando que todos los productos de los proyectos cumplan con los estándares de calidad y tiempos exigidos. Referente a la asistencia técnica se basa en la creación de algoritmos o scripts personalizados para las diversas actividades usando generalmente, por pedido del cliente, la librería ArcPy, el Software ArcGIS en conjunto con Microsoft SQL Server y PostGIS/PostgreSQL. El objetivo de esta asistencia es reducir al

máximo las tareas repetitivas y asegurar que el producto cumpla con las directrices requeridas por el cliente.

Adicionalmente se brinda asistencia técnica en temas relacionados con geoprocesos al Área de Oficina Técnica de del cliente brindándole apoyo técnico en el procesamiento de la información relacionado con la redes internas y externas de gas.

# II. Descripción de una actividad específica

## 2.1 Introducción a la actividad especifica

El proyecto tiene el fin de diseñar herramientas de automatización del proceso de validación y carga de nuevas redes de gas. Los procesos que se automatizan principalmente son los que identifican errores en la geometría, los que asignan la dirección de la red y los que heredan información de otras fuentes. La automatización se realiza mediante la creación de herramientas que son diseñadas o escritas en el lenguaje de programación Python usando principalmente la librería ArcPy de ArcGIS. Los elementos creados son vinculados a Pop-Up con el fin de ser usados dentro del entorno de ArcMap.

### *2.1.1 Alcance de la automatización en la validación de redes de gas*

El proyecto se limita a analizar las nuevas redes que se deben agregar a la base de datos del cliente, es decir, no se realizan análisis de los proyectos ya enlistados anteriormente. Tampoco abarca los procesos referentes a errores de diseño en campo, estos se descartan inicialmente y no forman parte de este proyecto. El proyecto solo incluye a las tuberías de polietileno, quedando excluidas las tuberías de acero que se manejan en un proyecto paralelo.

El proceso de validación se realiza a todas las actividades que involucran la validación topológica de la geometría de la nueva red. Además, se debe validar que la nueva red este conectada a la red existente. También abarca los procesos de validación de datos asignados y herencia de datos de otras fuentes. No se considera los procesos de validación de documentos asociados.

### *2.1.2 Contexto y relevancia del proyecto*

El proyecto se desarrolla en el contexto de la distribución de ganas natural a las zonas residenciales, zonas de comercio e industria en Lima y Callao. El cliente firmo contrato el año 2022 con el Estado Peruano para ser la empresa transportadora de gas del Perú dándole los derechos y obligaciones del contrato BOOT de Concesión para la distribución de gas natural

por red de ductos en el Departamento de Lima y Callao. El contrato tiene por 33 años prorrogables previa anticipación de cuatros años. (Mejia Del Carpio, 2022).

## 2.2 Otros casos

Previamente al ingreso del proyecto, el proceso de validación y carga de redes se realizaba usando herramientas del software ArcGIS, pero con la salvedad que no se automatizaba los procesos, lo que repercutía en la calidad del producto, recursos necesarios e incumplimientos de tiempos.

Por tal motivo para poder automatizar los procesos se revisó la bibliografía disponible con el objetivo de analizar la viabilidad de la automatización. Encontrando casuísticas relativamente parecidas que resuelven sus problemas mediante la creación de herramientas personalizadas SIG. Tal es el caso de la automatización de la búsqueda de códigos prediales, cruzando distintas fuentes, usando el software ArcGIS (Molina Herrera & Rodríguez Vivas, 2019). De igual forma se automatizo los procesos de control calidad para la actualización de series cartográficas, donde diseñaron herramientas que combinan softwares CAD y GIS (Torrent Foz, Muñoz-Nieto, Gonzales Aguilera, & Rodriguez Gonzálvez, 2021), De una manera más global, (Cooper Crispin & Chiaradia Alain, 2020), desarrollaron una caja de herramientas para el análisis de redes espaciales tridimensionales, especialmente orientado a calles, caminos y rede urbanas. Se basaron en diversos lenguajes de programación, pero principalmente en el uso de Python y C++.

En base a estas experiencias se concluyó que es posible crear herramientas personalizadas que puedan realizar procesos repetitivos de forma automática dando ventajas en el tiempo de ejecución y la certeza de obtener siempre los mismos resultados. Para la automatización de nuestros procesos se usaron la combinación de las herramientas que da el software ArcGIS mediante su librería ArcPy y el Python. Debido a que combina el poder de la

programación con el uso de herramientas espaciales que en suma permiten el manejo de información georreferenciada (ESRI, 2021).

## 2.3 Red en SIG

Es un sistema de enlaces conectados por nodos, donde los enlaces pueden ser tuberías, calles, ríos y cualquier elemento lineal del territorio y los nodos pueden ser válvulas, intersecciones viales, pozos etc. Se considera una red pura o simplemente red cuando solo se considera su conectividad y su topología, es decir, que cuando red solo tiene elementos conectados entre si por nodos y además se considera su forma de distribución se considera red pura. Pero si se le considera además de su conectividad y topología se le agrega atributos de flujo de le denomina una red de flujo. (Bell & lida, 1977)

Una red de servicio es una representación del transporte de un elemento como puede ser el agua, gas, desagüe por un enlace en determinada dirección, este elemento no puede elegir su dirección ya que está condicionado por fuerzas externas. Las redes de servicios son un elemento importante de los SIG y actualmente están enfocados en el análisis de la ruta más corta, flujo máximo de la red, costes mínimos y máximos. (Curtin, 2013).

La red de gas natural es una red de servicio que transporta este elemento por tuberías de conexión y que esta administrado por válvulas de control y estaciones de regulación de presión. La red tiene como objetivo distribuir el gas a clientes residenciales, clientes comerciales y clientes industriales. En el presente trabajo solo se considera la distribución de gas en la ciudad, no considera el transporte desde la fuente, tampoco se considera la administración propia de la red, solo se considera la agregación de las nuevas redes considerando las reglas configuradas por el cliente.

## 2.4 Automatización en ArcGIS

### *2.4.1 Fundamentos del ArcPy*

Los sistemas de información geográfica, en adelante SIG, desempeñan un papel protagónico en el manejo y análisis de datos espaciales en casi todos los ámbitos de la industria relacionados con el espacio. En ese contexto surge ArcGIS, que fue desarrollado por ESRI, siendo el software más usado en el mundo por su versatilidad y adaptación para diferentes ámbitos de la geografía. Por otra parte, esta Python un lenguaje de programación de alto nivel de propósito general, siendo el lenguaje de programación más usado para la automatización de tareas, tiene acceso a una amplia biblioteca lo que permite interconectar diferentes servicios, incluyendo el procesamiento de datos espaciales en ArcGIS.

ArcGIS unió ambos conceptos el software de escritorio SIG y el lenguaje de programación Python, creando ArcPy que es una biblioteca de Python que proporciona un conjunto de herramientas y módulos para la manipulación, análisis y automatización de datos geoespaciales dentro del entorno de todo el paquete de ArcGIS. (ESRI, 2021). ArcPy está compuesto por módulos, clases y funciones que interactúan entre sí, dentro de los principales módulos están:

- arcpy.analysis: Modulo de herramientas que sirven para el análisis espacial y geoprocesos relacionados.
- arcpy.management: Modulo de herramientas para la administración de datos.
- arcpy.mapping: Modulo de herramientas para manipular mapas dentro de ArcMap.
- arcpy.da: Modulo para la manipulación de los datos de la tabla de atributos a nivel fila y campo, acá se destaca los cursores de búsqueda, de actualización e inserción de datos.
- arcpy.env: Modulo para controlar el entorno de geoprocesamiento.
- arcpy.sa: Modulo de herramientas para el análisis especial avanzado, con la extensión Spatial Analys.

- arcpy.na: Modulo para trabajar las funcionalidades de red con la extensión Spatial Network Analyst.

### *2.4.2 Creación de herramientas en ArcGIS con ArcPy*

Para la creación de herramientas personalizas escritas con Python y ejecutarlas sobre el entorno de ArcMap y tener el aspecto de una herramienta de geoprocesamiento del sistema, es necesario crear primero una caja de herramientas de secuencia de comandos. Esta forma de trabajo proporciona ventajas como tener la misma funcionalidad de herramientas de geoprocesamiento nativa, se puede escribir mensajes en el cuadro de dialogo de progreso y también en la ventana de resultados, se puede documentar la herramienta.

Las herramientas creadas se basan en parámetros o variables de entrada, las cuales son creadas y asignadas en el código Python, estos parámetros son de determinado tipo de dato que ya funciona por si misma en el archivo Python, pero si el script debe desea ser usado en una herramienta dentro del entorno de ArcMap, debe ser asignado con la función arcpy.GetParameterAsText(0) donde el numero hace referencia al orden del parámetro, en este caso, el cero hace referencia que es el primer parámetro.

Las variables dentro del script son asignados dependiendo a la secuencia de comandos creados, pueden ser usado fuera o dentro del entorno de ArcMap, para usarlo dentro de la caja de herramientas, estas variables, se convierten en parámetros y puede configurarse de acuerdo a la necesidad, la caja de herramientas ofrece un abanico amplio de tipo de parámetros que cubre todo el espectro de posibilidades.

### *2.4.3 Creación de Add-in Tools*

Python Add-In de ArcGIS es un complemento que permite integrar nuevas herramientas, controles y funcionalidades dentro de ArcMap usando el lenguaje de programación Python, facilitando la automatización de tareas, creación de flujos de trabajo. Permite crear interfaces de usuario personalizado mediante la agregación y configuración de

botones, herramientas, cuadros de dialogo las cuales se les da funcionalidades escritas en Python.

Permite agrupar herramientas creadas mediante scripts asociándolas a tools o botones, esta función es de bastante ayuda, ya que se puede agrupar y empaquetar un grupo de scripts y ser compartida con otros usuarios. Para acceder a ellas solo se necesita ejecutar la primera vez y se instala en el ArcMap del usuario.

## 2.5 Productos Iniciales del Proceso de Validación y Carga de Redes

El objetivo del proyecto es agregar a una base de datos geoespacial las redes construidas en campo, con el fin de poder gestionar dichas redes. Para ello, las empresas constructoras envían los datos de lo construido en cuatro formatos, tres de ellos contiene la información espacial y el cuarto es un documento denominado Acta de Barrido con los metrajes del proyecto. Los formatos con contenido georreferenciado son una geodatabase, un archivo CAD y un plano en formato PDF.

La geodatabase es un archivo digital del software ESRI donde la empresa responsable de la construcción dibuja la red y lo representa mediante líneas y puntos separada por capas, es decir capa elemento de la red va en distinto Feature Class. El archivo que envía es básicamente el diseño de la red, mas no indica los atributos o información relacionada que es necesaria para poder ingresar la red al sistema, también es muy frecuente que contenga errores topológicos y tampoco indica el flujo de la red.

El archivo CAD y el Archivo PDF al igual que la geodatabase contiene información del diseño de la red, pero a diferencia de la geodatabase contiene información adicional, pero en un formato no convertible a GIS. A diferencia de la geodatabase, el archivo CAD y PDF son generados anterior a la construcción y que se valida insitu con lo construido, lo que ocasionalmente puede tener diferencias con la geodatabase.

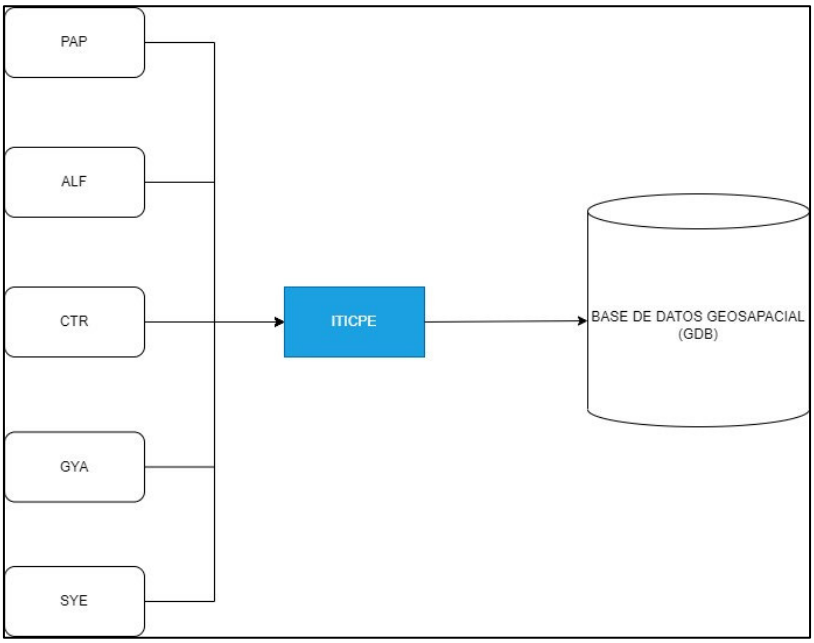
Finalmente, está el acta, denominado Acta de Barrido, es el documento que valida los metrados construidos, este documento se genera posterior a la habilitación y cuenta con la aprobación de la empresa constructora y los supervisores correspondiente. El Acta contiene información de las longitudes construidas por diámetro y debe ser validado con los diseños enviados en los archivos digitales, sirve como fuente de información oficial.

## 2.6 Flujo de entrega de información de las Redes

La construcción de la red esta asignada a cinco empresas contratistas, es decir, la empresa concesionaria subcontrata la construcción de la red a cinco empresas, las cuales son las encargadas de la construcción, habilitación y el reporte de los planos para poder ser subida a la base de datos espacial. El formato en que se envía la información es en un plano de formato geodatabase, plano CAD y archivos en PDF. Las empresas son responsables que la información enviada en los planos corresponda con los construido y sea igual entre los distintos formatos. El flujo general se puede resumir en el siguiente gráfico.

**Figura 2**

*Flujo de entrega de proyectos*



Posterior al envió de la información, inicia nuestras actividades de validación de información geométrica y alfanumérica. Validamos que la información brindada por las empresas constructoras sea correcta y valida. Para luego agregar a la base de datos geoespacial las nuevas redes validadas, respetando las reglas de las herramientas Network del software ArcGIS.

## 2.7 Volumen de trabajo

El volumen de información promedio es de 1500 km anules, repartido a lo largo del año, cabe indicar que la producción no es constante, habiendo picos de envió en las dos últimas semanas de cada mes. Estos picos representan un reto para la carga ya que puede exceder nuestros recursos destinados para el desarrollo de nuestras actividades, es por ello la necesidad de automatizar procesos.

La cantidad estimada para los próximos años tiende a la baja, porqué el territorio del área de concesión ya está en gran parte cubierto, lo que no necesariamente implica menor cantidad de trabajo, debido que ahora se está comenzando a cubrir zonas que por motivos de fuerza mayor fueron denegados en su momento, estas zonas son denominadas difíciles lo cual independientemente del metraje habilitado representa dificultad para su procesamiento.

## 2.8 Geodatabase de la red de gas

La geodatabase con la que se trabaja es una réplica local de la geodatabase principal, está compuesto por un dataset que contiene todas las capas que conforman la red de gas. La información se encuentra dentro de un dataset ya que es impresindible para configurar una red geométrica de ArcGIS. En total son veinte capas, donde una de ellas es de tipo polígono, tres son de tipo líneas y 16 son de tipo punto, además, en total son 1058 campos, la siguiente tabla ofrece un resumen de las estadísticas de las capas.

**Tabla 1**

*Estructura de la geodatabase*

| CAPA | TIPO DE GEOMETRÍA | NUMERO DE CAMPOS |
|---|---|---|
| Feature: NODOVIRTUAL | Punto | 30 |
| Feature: VALVULAEXCESOFLUJO | Punto | 65 |
| Feature: VALVULA | Punto | 79 |
| Feature: TAPON | Punto | 47 |
| Feature: TRAMOGASODUCTO | Línea | 80 |
| Feature: CODO | Punto | 46 |
| Feature: NODOERP | Punto | 49 |
| Feature: TRAMPASCRAPPER | Punto | 45 |
| Feature: LINEAINTERNA | Línea | 46 |
| Feature: OBRAESPECIAL | Punto | 62 |
| Feature: POLIGONOETAPA | Polígono | 17 |
| Feature: UNIONES | Punto | 36 |
| Feature: ODORIZADOR | Punto | 32 |
| Feature: ERM | Punto | 68 |
| Feature: DERIVACION | Punto | 48 |
| Feature: ERP | Punto | 89 |
| Feature: GRM | Punto | 53 |
| Feature: TRANSICION | Punto | 42 |
| Feature: TUBERIACONEXION | Polilínea | 77 |
| Feature: REDUCCION | Punto | 47 |
| **TOTAL** | **20** | **1058** |

La cantidad de capas existentes y la cantidad de filas para evaluar justifica la creación de herramientas que automaticen el trabajo.

## 2.9 Plazos de trabajo

Cada proceso está programado para ser cumplido en plazos determinados, fundamentalmente por dos razones, existen factores de venta de nuevas tuberías de conexión que solo se habilitan cuando la red se encuentra en el sistema gis y la segunda permite hacer simulaciones en caso de accidentes relacionados con fugas de gas.

Hay dos plazos que se deben cumplir en el proceso de carga de redes, la primera son los días que la empresa constructora tiene para enviar los archivos digitales, la segunda son los

días que se tiene para validar y subir al sistema GIS. Los tiempos se muestran en la tabla número dos.

**Tabla 2**

*Días por procesos*

| Proceso | Días |
|---|---|
| Proceso de entrega de la constructora | 3 |
| Proceso de carga al sistema SIG | 2 |

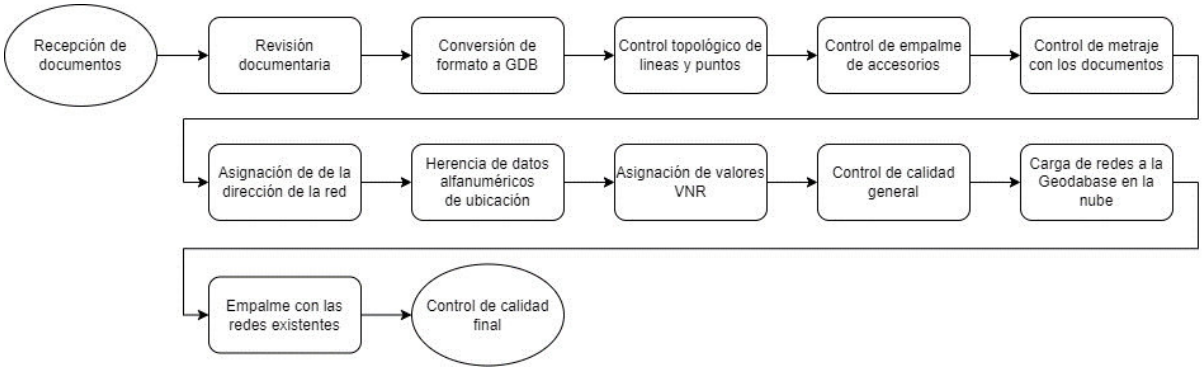
Los plazos que se tiene no solo contemplan el proceso principal, también está contemplado los tiempos que toma levantar observaciones en caso de presentarse.

## 2.10 Análisis de procesos

El proceso de carga de las nuevas redes de gas inicia en la recepción de los proyectos por parte de las contratistas, pasando por el control documentario, topología de los elementos espaciales, asignación de atributos alfanuméricos, control de calidad y carga a la base de datos espacial GIS. Todos los procesos inicialmente se realizaban manualmente lo que repercutía en los recursos asignados tanto como de personal como de software y hardware.

**Figura 3**

*Actividades del proceso*

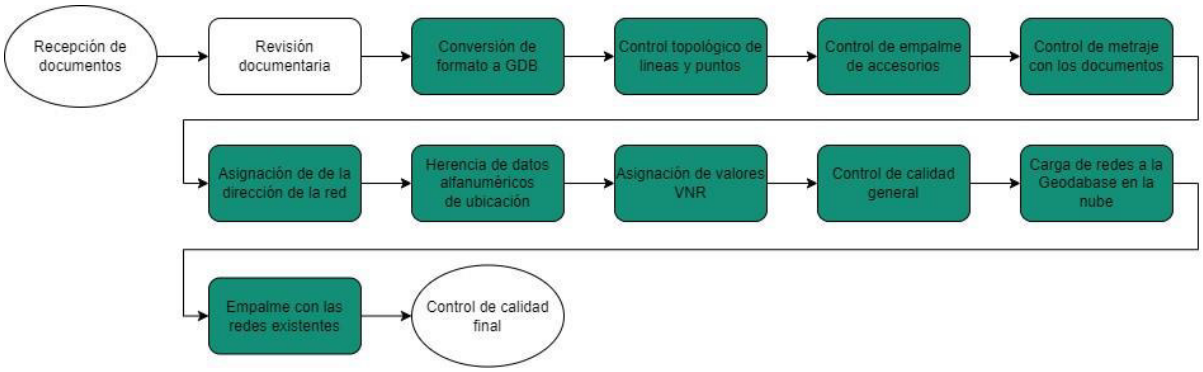



Posterior al análisis de los procesos, se automatizó creando herramientas personalizadas en base al lenguaje de programación Python y la librería de ArcMap Arcpy, los procesos que

se automatizaron representan el 80% del proceso general, lo que se tradujo en menores recursos técnicos de software, hardware y tiempos. Debido a que los requerimientos de la empresa concesionaria son dinámicas, es decir, que puede cambiar las directrices de carga de redes, los scripts creados se adaptan a los requerimientos. Los procesos que se automatizaron son los marcados de color verde en la figura cuatro.

**Figura 4**

*Actividades automatizadas del proceso*

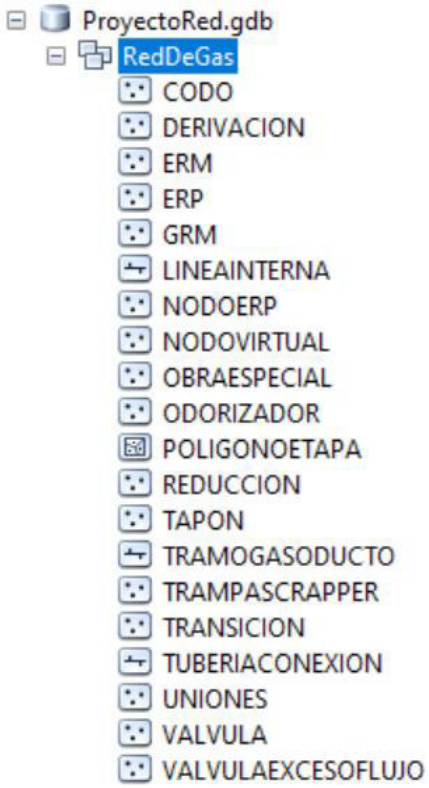



## 2.11 Desarrollo de herramientas personalizadas

El producto inicial o el punto de partida es una de una geodatabase. Esta base de datos comprende la geometría integral de las tuberías y los componentes de la red de distribución. Sin embargo, no está exenta de errores, las capas de datos presentan errores topológicos que incluyen superposiciones de líneas y puntos, elementos desconectados, y tuberías que carecen de una dirección de flujo definida, entre otros problemas técnicos. Adicionalmente, se enfrenta al desafío de que todas las capas llegan carentes de información asociada, lo que requiere una atención meticulosa para su correcta integración y uso. La figura cinco ilustra detalladamente las capas contenidas en la geodatabase, proporcionando una representación visual de la estructura de datos con la que se trabajará.

**Figura 5**

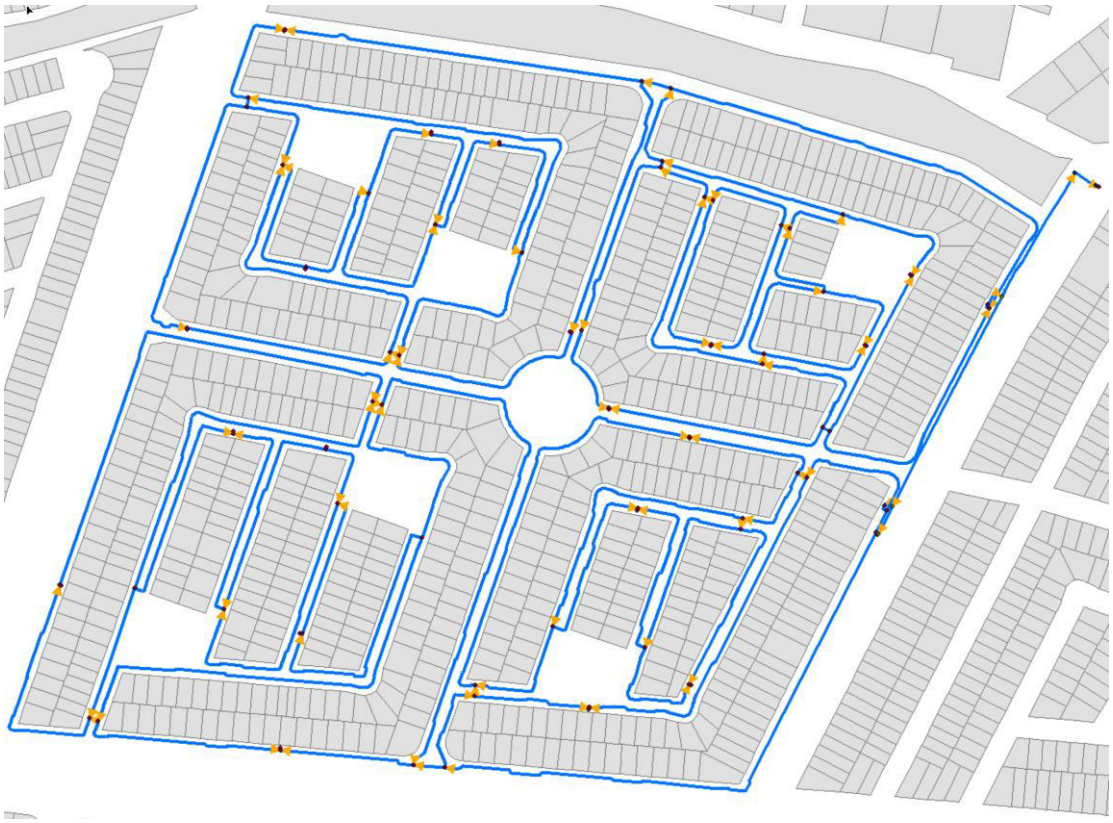
*Capas de la geodatabase*




Un ejemplo de la geometría de una red se muestra en la figura número seis, donde el punto de empalme con la red principal se encuentra en la parte superior derecha de la figura. A partir del punto de empalme se distribuye el gas a través de toda la red de tuberías, por lo tanto, el flujo de la red viene en ese sentido. A la escala de la figura en que se muestra la red, no se puede identificar visualmente los posibles errores topológicos, debido a ello, las herramientas son de gran ayuda para la detección de los errores.

**Figura 6**

*Ejemplo de proyecto de red*



En cuanto a la tabla de atributos de las capas, estas vienen sin información adicional de los elementos a los que representan. Por lo tanto, la tarea de las herramientas creadas es poder completar la información necesaria. El tipo de información que se agregara es de tipo de identificación del elemento, identificación al proyecto al que pertenece, información de la ubicación de la red, información de la función que cumple y de las características físicas del elemento. En la figura siete se muestra un ejemplo de la tabla de atributos inicial sin datos.

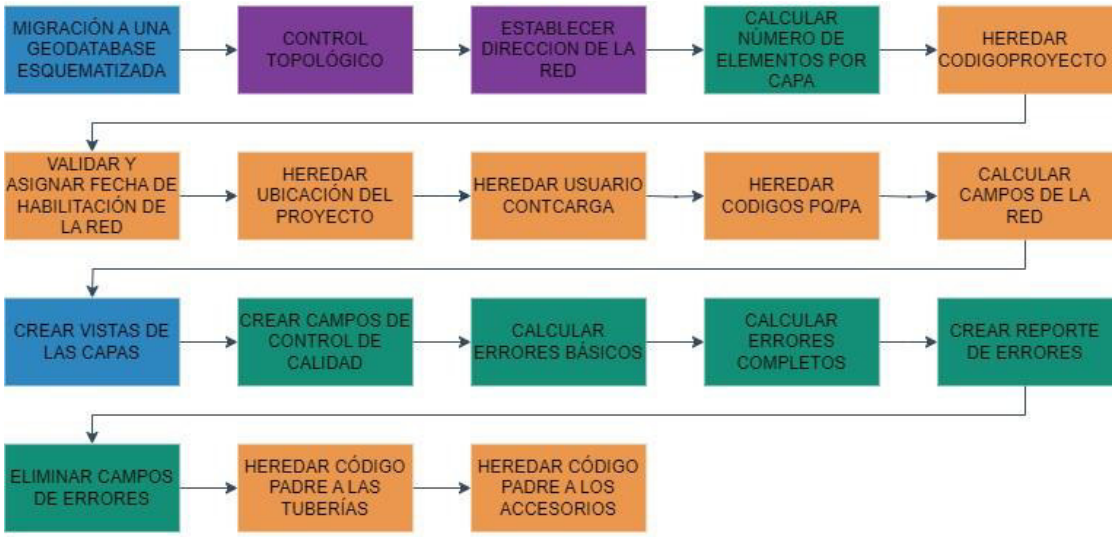
**Figura 7**

*Tabla de atributos vacía*

Se crearon dieciocho herramientas para cubrir las diez actividades que se necesitan para validar las nuevas redes de gas. Están divididos en cuatro tipos, el primer tipo, son procesos intermedios necesario para la ejecución de las demás herramientas, en la figura ocho se representan de color celeste, el segundo tipo son validaciones topológicas geométricas, en la figura ocho se muestran de color morado, el tercero tipo, son validaciones y herencia de datos, en la figura ocho se muestra de color anaranjado, por último, herramientas de control de calidad general en la figura ocho se muestra de color verde.

***Figura 8***

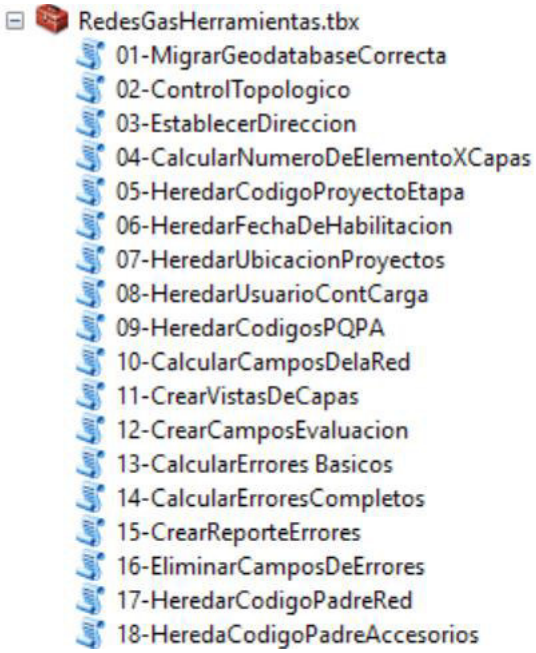
Herramienta creadas por categoría




En la figura número nueve muestro herramientas que se han desarrollado y están almacenadas en una caja de herramientas de ArcGIS. Cada herramienta está diseñada para realizar una actividad especifica que en conjunto se complementan. Están organizados en un orden especifico de ejecución, de manera que se ejecute un tras de otra.

**Figura 9**

*Herramientas desarrolladas*




A continuación, describiremos en términos generales las tareas que realiza cada herramienta desarrollada, detallando el código fuente en las principales herramientas creadas.

1) Migrar geodatabase correcta: La geodatabase que envía la empresa que construyo la red puede estar desfasada en cuanto a las cantidades de campos en los atributos, subtipos, dominios, es por ello que los elementos deben ser copiados a una geodatabase estandarizada. Debido a la cantidad de capas se desarrolló una herramienta que haga este trabajo automáticamente.

2) Control topológico: La geodatabase que envía la contratista es una transformación simple de un archivo CAD a formato geodatabase, entonces existe la posibilidad que pueda arrastrar errores de tipo líneas duplicadas, puntos superpuestos, elementos desconectados. A fin de poder detectar estos errores se desarrolló una herramienta que recorra cada capa evaluando los errores típicos entre sí mismos y comparara con las demás capas. Los elementos detectados son exportados a una geodatabase temporal para poder ser detectados de manera más simple y puedan ser corregidos manualmente. Debido a la importancia de esta herramienta se procede a explicar detalladamente su desarrollo.

En primer lugar, se establece el espacio de trabajo y el parámetro de entrada para que funcione la herramienta, siendo el parámetro la geodatabase donde se almacena la red. Posteriormente, se establece los nombres de las redes topológicas, tal como muestra la figura nueve.

**Figura 10**

*Código fuente del control topológico*

```
1   import arcpy
2   # Especifica el directorio de trabajo y los nombres de las capas de entrada
3   gdb = arcpy.GetParameterAsText(0)
4   arcpy.env.workspace = gdb
5   dataset = gdb+"\\RedDeGas"
6   topo = dataset+"\\topologia_lineas"
7   topo_puntos = dataset + "\\topologia_puntos"
```

Seguidamente se enlista en una variable todas las capas con el fin de recorrer una a una evaluando sus posibles errores y además se comprueba que la geodatabase con los archivos temporales exista y de ser así la elimina y vuelve a crear otra, esto a fin de volver a ejecutar la herramienta más de una vez, ver figura diez.

**Figura 11**

*Código fuente del control topológico*

```
9   #Enlista todas las capas de toda la geodatabase y hace referencia a la capa de la tuberia
10  lyrs = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")
11  tg = lyrs[15]
12  #Enlista todas las capas que son de tipopunto
13  capas_puntos = [lyr for lyr in lyrs if arcpy.Describe(lyr).shapeType == "Point"]
14  space = gdb[:-19]
15
16  #Verifica que que la gdb que alberga las capas intermedias no exista, si existe, la elimina
17  #Posteriormente la crea
18  tempGDB = space +"\\temp.gdb"
19  if arcpy.Exists(tempGDB):
20      arcpy.Delete_management(tempGDB)
21  arcpy.CreateFileGDB_management(space, "temp")
```

Después se crea la regla topológica para identificar las líneas superpuestas, esto se realiza usando las herramientas topológicas del ArcGIS, pero para facilidad de la identificación en la línea al final del script se exporta a la geodatabase temporal con el fin de poder posteriormente identificarlo y corregir. Luego se hace lo mismo para las capas de tipo punto,

pero al ser muchas se dificulta crear bloques de código por cada una de ellas. Para agilizar este proceso se utiliza un iterador recorriendo cada capa y creando las reglas topológicas por cada una de ellas en un solo bloque de código. Este análisis solo se hace entre los elementos de la misma capa, el código se muestra en la figura once.

**Figura 12**

*Código fuente del control topológico*

```
#Crea una topologia para la capa de entrada
output_topology = "topologia_lineas"
arcpy.CreateTopology_management(dataset, output_topology)
arcpy.AddFeatureClassToTopology_management(topo, tg)
arcpy.AddRuleToTopology_management(topo, "Must Not Overlap (Line)", tg)
arcpy.ValidateTopology_management(topo, "Full_Extent")
arcpy.ExportTopologyErrors_management(topo, tempGDB, "TablaDeSuperposiciones")

#Se Itera sobre cadas de puntos y crea topología para cada una
for capa_puntos in capas_puntos:
    topo_puntos = "{}\\topologia_{}".format(dataset, capa_puntos)
    arcpy.CreateTopology_management(dataset, "topologia_{}".format(capa_puntos))
    arcpy.AddFeatureClassToTopology_management(topo_puntos, capa_puntos)
    arcpy.AddRuleToTopology_management(topo_puntos, "Must Not Overlap (Point)", capa_puntos)
    arcpy.ValidateTopology_management(topo_puntos, "Full_Extent")
    arcpy.ExportTopologyErrors_management
    (topo_puntos, tempGDB, "TablaDeSuperposiciones_{}".format(capa_puntos))
```

Para comparar con otras capas se desarrolló el siguiente bloque, donde lo que hace es tomar una capa de tipo punto y va comparando con todas las capas, una a una, luego hace lo mismo con la siguiente capa. De esta forma se asegura que haya sido analizados todas las capas. el código se muestra en la figura doce.

**Figura 13**

*Código fuente del control topológico*

```
# Verifica la superposición entre todas las capas de puntos
for i in range(len(capas_puntos)):
    for j in range(i + 1, len(capas_puntos)):
        capa1 = capas_puntos[i]
        capa2 = capas_puntos[j]
        topo_capa1 = "{}\\topologia_{}".format(dataset, capa1)
        arcpy.AddFeatureClassToTopology_management(topo_capa1, capa2)
        arcpy.AddRuleToTopology_management(topo_capa1, "Must Not Overlap (Point)", capa2)
        arcpy.ValidateTopology_management(topo_capa1, "Full_Extent")
        arcpy.ExportTopologyErrors_management(topo_capa1, tempGDB, "TablaDeSuperposiciones_{}_{}".format(capa1, capa2))
```

3) Establecer dirección de la red: el flujo de la red tiene una lógica de distribución de gas que se transporta por las tuberías que, desde la troncal hacia la red, tal como se muestra en la

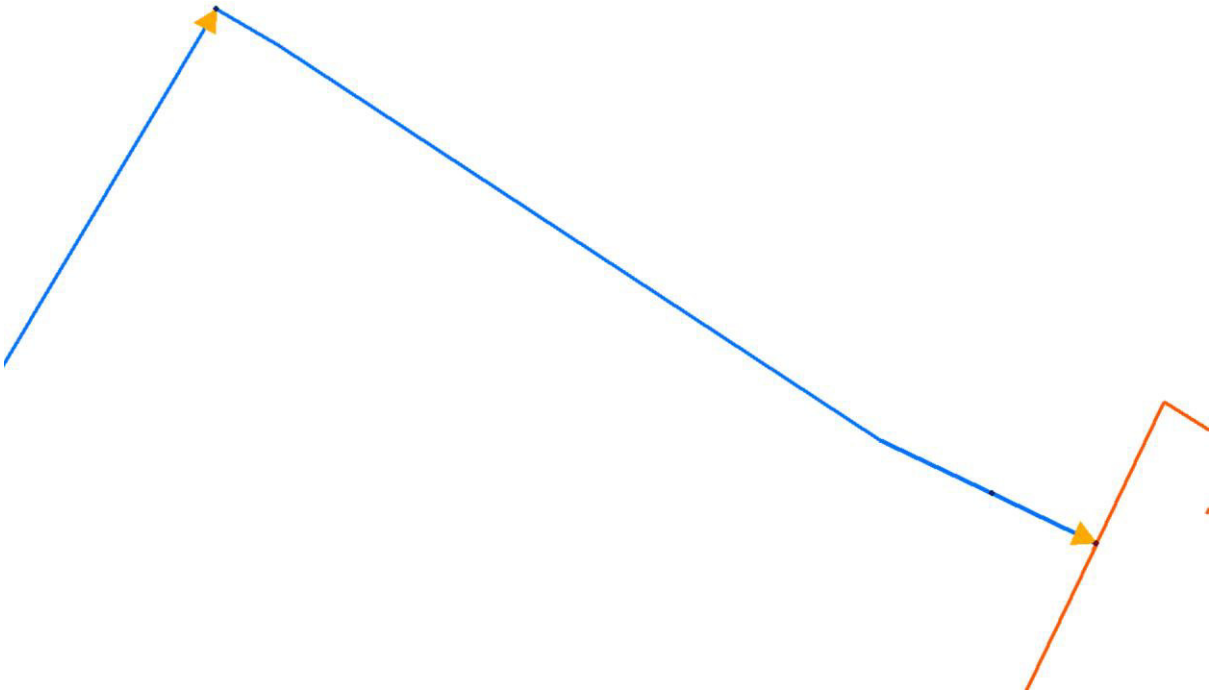
figura catorce, donde las líneas de color anaranjado es la red existente y las líneas de color azul, es la red que se agregara. La dirección del gas va en el sentido de la existente a la nueva red. El sentido del gas lo determina la dirección de digitalización de la red.

**Figura 14**
*Lógica de la dirección de la red*

Si se visualiza el punto de empalme de esta red modelo en la figura quince, se puede ver que la dirección, indicado por las flechas, de las nuevas tuberías está en sentido hacia la red existente, lo cual es un error según la lógica diseñada para el flujo de la misma.

**Figura 15**

*Punto de empalme de la red*



La nueva red que ingresa tiene la dirección de la red desconfigurada, por lo que se debe establecer la dirección manualmente línea por línea, este proceso puede representar el 40% del tiempo estimado del total de las actividades, por lo que automatizar esta actividad fue de gran ayuda, porque se redujo todo este tiempo a cero. Se detalla el código de la herramienta debido a su importancia.

Para desarrollar esta actividad, se necesitará tres parámetros iniciales, la geodatabase que contiene la red, el id del elemento tipo línea que empalma con la red existente, y el id del accesorio que empalma a la red. A partir de este punto se ira estableciendo el flujo de la línea. El script analiza todos los elementos de la capa línea, con cada comando de código puede volver a recorrer la red, ocasionando que se desconfigure si no se implementa un método de control, para evitar ello, se agrega dos columnas temporales donde se registra con un identificador las líneas ya analizadas y se omiten en cada proceso, el código se muestra en la figura quince..

**Figura 16**

*Código fuente Establecer dirección de la red*

```
1   import arcpy
2   # recibe tres parametros, la geodatabase, el id del elemento que se empalma a la red
3   #parametro del accesorio de empalme a la red principal
4   gdb = arcpy.GetParameterAsText(0)
5   idTramo = arcpy.GetParameterAsText(1)
6   idDerivacion = arcpy.GetParameterAsText(2)
7   #Se define el entorno de trabajo
8   arcpy.env.workspace = gdb
9   arcpy.env.overwriteOutput = True
10  capas = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")
11  #Se asigna a variables la capas que se usaran
12  tramo = capas[15]
13  tapon = capas[14]
14  derivacion = capas[3]
15  valvula = capas[0]
16  #Se agrega los campos que ayudaran con el proceso, se asigna como 0 por defecto
17  arcpy.AddField_management(tramo, "flow","SHORT")
18  arcpy.AddField_management(tramo, "work","SHORT")
19  arcpy.CalculateField_management(tramo, "flow", 0)
20  arcpy.CalculateField_management(tramo, "work", 0)
```

Una vez establecido los parámetros generales de la red, se crea funciones que dan flujo a las tuberías, la primera de las funciones se llama flipFirstLine(), esta trabaja con las líneas que empalma a la red, es el punto de partida, recibe como parámetro el tramo inicial, el accesorio inicial, los campos de la red y los campos de los puntos. Se crea inicialmente un cursor que recorre todos los ID de la derivación a fin de identificar el elemento que se conecta a la red, una vez identificado extraer las coordenadas y los almacena en variables, este proceso se muestra desde la línea 38 hasta la línea 48 de la figura once. Seguidamente se crea otro cursor que toma como parámetros la capa de las tuberías y los campos, el script recorre los id hasta detectar el inicial, le calcula los vértices final e inicial, y los compara con las coordenadas del accesorio de empalme almacenado anteriormente, si el accesorio coincide con el punto inicial de la línea no ejecuta ninguna acción, pero si no coincide el script invierte la línea de modo que el punto inicial coincida con el primer accesorio, de esta manera ya tenemos configurada la dirección de la primera línea, podemos ver el resultado en la figura número 16.

**Figura 17**

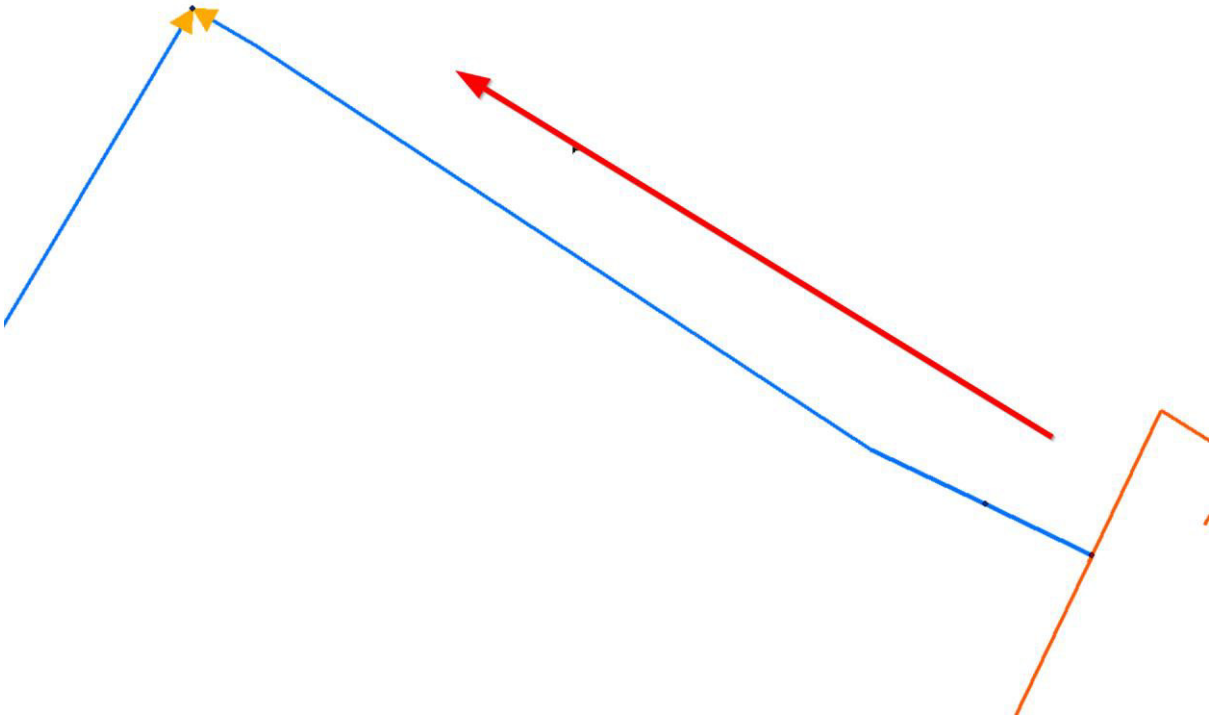
*Código fuente: Establecer dirección de la red*

```
#Se crea la primera funcion que le da la direccion a la primera linea
def flipFirstLine (tramoInicial, fields, derivacionInicial, fieldsPoints):
    startX_tramo = 0
    startY_tramo = 0
    with arcpy.da.SearchCursor(derivacionInicial, fieldsPoints) as cursor_d:
        tablaPointsDerivaciones = []
        for row_d in cursor_d:
            startX_derivacion = row_d[1][0]
            startY_derivacion = row_d[1][1]

            primerPunto_derivacion = arcpy.Point(startX_derivacion, startY_derivacion)
            pg_derivacion = arcpy.PointGeometry(primerPunto_derivacion, arcpy.SpatialReference(32718))
            tablaPointsDerivaciones.append(pg_derivacion)
    with arcpy.da.UpdateCursor(tramoInicial, fields) as cursor:
        for row in cursor:
            startPoint = row[1].firstPoint
            startX_tramo = startPoint.X
            startY_tramo = startPoint.Y
            primerPunto_tramo = arcpy.Point(startX_tramo, startY_tramo)
            pg_tramo_inicial = arcpy.PointGeometry(primerPunto_tramo, arcpy.SpatialReference(32718))
            vDi = 0
            for tpd in tablaPointsDerivaciones:
                if pg_tramo_inicial.equals(tpd) == True:
                    vDi +=1
            if vDi == 0:
                vertices0 = []
                for part in row[1]:
                    for puntos in part:
                        vertices0.append(puntos)
                    row[1] = arcpy.Polyline(arcpy.Array(vertices0[::-1]))
                    row[2] = 1
                    row[3] = 1
                    cursor.updateRow(row)
            else:
                row[2] = 1
                row[3] = 1
                cursor.updateRow(row)
```

El resultado del script es que la primera línea, independientemente de su estado inicial, toma la dirección correcta, es decir de la red existente hacia la nueva red, tal como se muestra en la figura 17. Este es el primer paso para establecer la dirección de todos los elementos de la tubería, ya que es el primer elemento seguro que su dirección es correcta.

**Figura 18**

*Dirección correcta de la primera línea*



Después se establece el flujo de las tuberías que obedecen a reglas especiales como lo es las tuberías que terminan en un accesorio de tipo Tapón. Estos elementos solo admiten una dirección, que va en sentido hacia del tapón, entonces previamente a establecer el flujo a la red, estas redes se deben establecer su dirección. Para lograr esto, se crea una función llamada *flipLineTapon()* que recibe como parámetros el accesorio, los tramos conectados a este accesorio y los campos de cada capa. Mediante un primer cursor se almacena todas las coordenadas de los tapones y con el segundo cursor se calcula los vértices finales de cada línea conectada a cada Tapón, si coincide el vértice final con el tapón no se ejecuta ninguna instrucción, pero si no coincide se invierte la línea, de esta manera todas las tuberías que acaban en tapón ya tienen la dirección correcta. En el grafico numero 18 muestra el código y el 19 se muestra el resultado.

**Figura 19**

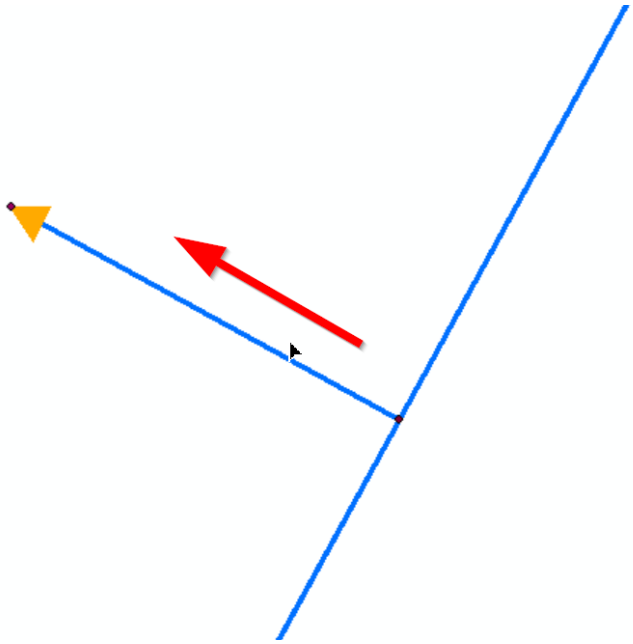
*Código fuente: Establecer dirección de la red*

```
#Se da la direccion a las tuberias que terminan en tapon
def flipLineTapon(tapon, fieldsPoints, tramo_tapon, fields):
    puntoTapon = []
    with arcpy.da.SearchCursor(tapon, fieldsPoints) as cursorTaponS:
        for rowS in cursorTaponS:
            xS = rowS[1][0]
            yS = rowS[1][1]
            xyS = arcpy.Poi (function) PointGeometry: Any
            pg_xyS = arcpy.PointGeometry(xyS, arcpy.SpatialReference(32718))
            puntoTapon.append(pg_xyS)

    with arcpy.da.UpdateCursor(tramo_tapon, fields) as cursorTaponS:
        for rowT in cursorTaponS:
            endPointTapon = rowT[1].lastPoint
            endX_tramoTapon = endPointTapon.X
            endY_tramoTapon = endPointTapon.Y
            puntoFinalTapon = arcpy.Point(endX_tramoTapon, endY_tramoTapon)
            pg_puntoFinalTapon = arcpy.PointGeometry(puntoFinalTapon, arcpy.SpatialReference(32718))
            vT = 0

            for pntTapon in puntoTapon:
                if pg_puntoFinalTapon.equals(pntTapon) == True:
                    vT =+1
            if vT == 0:
                vtx = []
                for partTapon in rowT[1]:
                    for pntsTapon in partTapon:
                        vtx.append(pntsTapon)
                    rowT[1] = arcpy.Polyline(arcpy.Array(vtx[::-1]))
                    rowT[2] = 999
                    rowT[3] = 999
                    cursorTaponS.updateRow(rowT)
            else:
                rowT[2] = 999
                rowT[3] = 999
                cursorTaponS.updateRow(rowT)
```

En la figura veinte se muestra como es el resultado de la ejecución del código descrito, se puede observar que la línea tiene la dirección en el sentido del Tapón.

**Figura 20**

*Flujo del tramo que finaliza en tapón*



Después se debe establecer el flujo de las tuberías contiguas a la tubería que acaba en tapón, ya que ellos obedecen a la regla de que si confluye dos tramos de direcciones opuestas estos deben acabar en tapón. Para dar solución de esta regla se creó una función llamada *flipNearTapon()* que recibe como parámetros las tuberías que confluyen en tapón, las tuberías que acaban en tapón y los campos involucrados. Como lógica del código se calcula primero el punto inicial de los tramos que acaban en Tapón mediante un cursor, estos puntos iniciales deben coincidir con los puntos finales de las tuberías que confluyen en la tubería acaba en tapón, de no coincidir el tapón se invierte y de esta forma se establece la dirección de los elementos. En el grafico numero veinte muestra el código y el 21 se muestra el resultado.

**Figura 21**

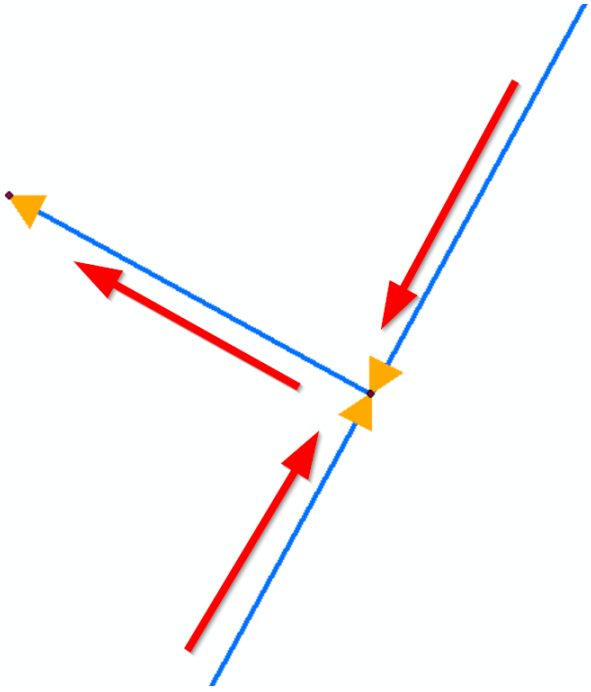
*Código fuente: Establecer dirección de la red*

```
def flipNearTapon(tramo_near_999, fields, tramo_near_tapon_work):
    puntoNearTapon = []
    with arcpy.da.SearchCursor(tramo_near_999, fields) as cursorNearTapon:
        for rowNT in cursorNearTapon:
            startPonintNT = rowNT[1].firstPoint
            x_nearTapon = startPonintNT.X
            y_nearTapon = startPonintNT.Y
            primerPunto_NT = arcpy.Point(x_nearTapon, y_nearTapon)
            pg_tramoNT_start = arcpy.PointGeometry(primerPunto_NT, arcpy.SpatialReference(37218))
            puntoNearTapon.append(pg_tramoNT_start)

    with arcpy.da.UpdateCursor(tramo_near_tapon_work, fields, "work NOT IN (1, 999)") as cursorNearTapon_U:
        for rowNT_U in cursorNearTapon_U:
            lastPonintNT_u = rowNT_U[1].lastPoint
            x_nearTapon_u = lastPonintNT_u.X
            y_nearTapon_u = lastPonintNT_u.Y
            primerPunto_NT_u = arcpy.Point(x_nearTapon_u, y_nearTapon_u)
            pg_tramoNT_start_u = arcpy.PointGeometry(primerPunto_NT_u, arcpy.SpatialReference(37218))
            valor_nt = 0
            for pnt_nt in puntoNearTapon:
                if pg_tramoNT_start_u.equals(pnt_nt) == True:
                    valor_nt += 1
            if valor_nt == 0:
                vtx_nt = []
                for part_nt in rowNT_U[1]:
                    for pnts in part_nt:
                        vtx_nt.append(pnts)
                    rowNT_U[1] = arcpy.Polyline(arcpy.Array(vtx_nt[::-1]))
                    rowNT_U[2] = 888
                    rowNT_U[3] = 888
                    cursorNearTapon_U.updateRow(rowNT_U)
            else:
                    rowNT_U[2] = 888
                    rowNT_U[3] = 888
                    cursorNearTapon_U.updateRow(rowNT_U)
```

En la figura 22 se muestra cómo debería ser la dirección correcta de la red.

**Figura 22**

*Fujo de la red de los elementos cercado al tapón*



Una vez de establecido el flujo de los casos especiales, Al igual que en los procesos anteriores, se creó una función llamada *flipOneLine()* que toma como parámetros a la primera línea, y los campos involucrados. Mediante un primer cursor se calcula el punto final de la primera línea, y se almacena en una variable, luego con un segundo cursor se calcula el punto inicial de la línea contigua a la primera línea, se compara si ambos puntos coinciden, de no coincidir se invierte la línea y así con un efecto domino se establece la dirección de todas las líneas de la red. Las líneas de comando se detienen cuando encuentran una línea previamente analizada. En la figura 22 se muestra la secuencia de comandos y en la figura 23 se muestra el resultado final de la herramienta.

Entonces con el uso de esta herramienta se estableció el flujo de toda la red tan solo usando una herramienta que es capaz de cumplir con todas las reglas del flujo de la dirección de la red. Esta herramienta es quizá la más importante de todas ya que resuelve la complejidad de la red y reduce los tiempos de horas de trabajo a segundo de ejecución.

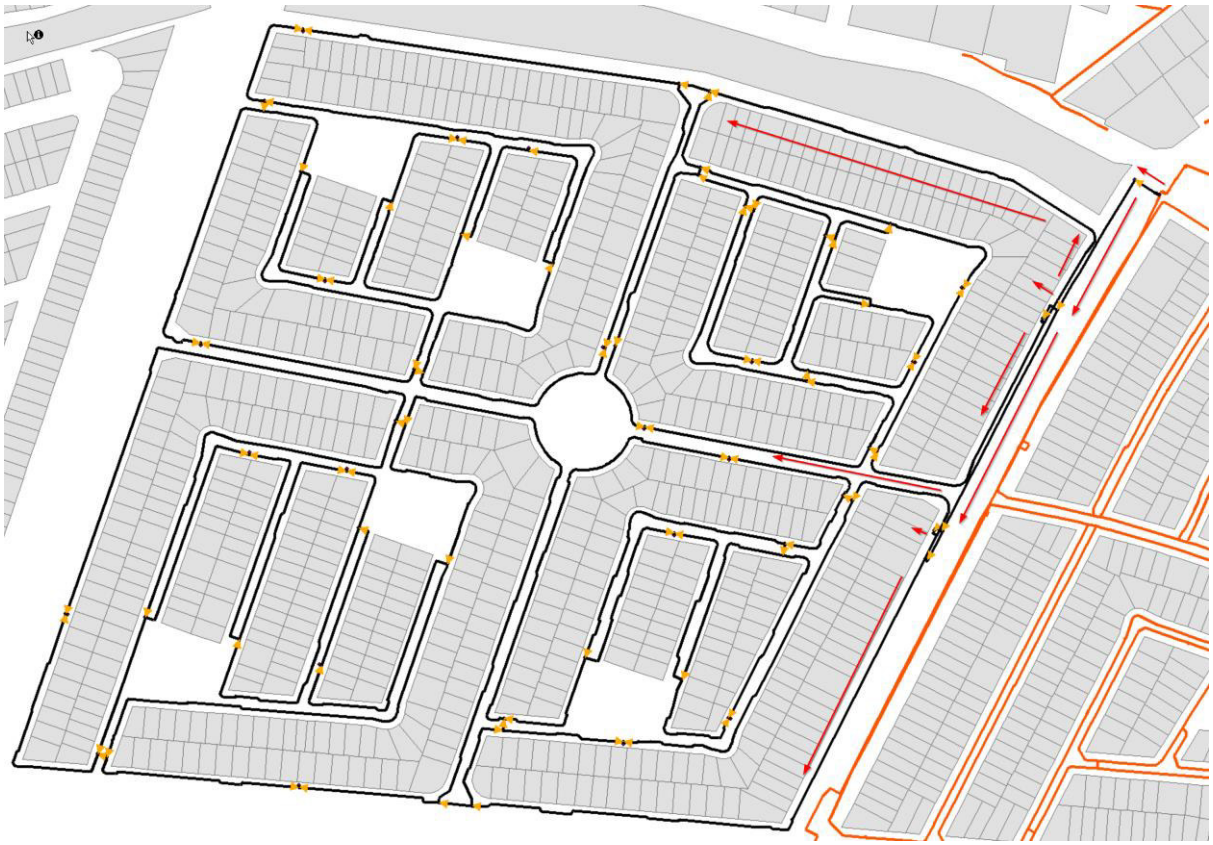
**Figura 23**

*Código fuente: Establecer dirección de la red*

```
def flipOneLine (tramo_trabajo, seleccion, fields):
    puntosFinales =[]
    #crea el curso para todos los elementos que aun no fue establecido la direcion
    with arcpy.da.SearchCursor(tramo_trabajo, fields, where_clause= "flow > 0") as cursor0:
        #Recorre cada linea almacenando los puntos iniciales
        for row0 in cursor0:
            endPoint1 = row0[1].lastPoint
            endX_tramo = endPoint1.X
            endY_tramo = endPoint1.Y
            ultimoPunto_tramo1 = arcpy.Point(endX_tramo, endY_tramo)
            pg_tramo_end0 = arcpy.PointGeometry(ultimoPunto_tramo1, arcpy.SpatialReference(32718))
            puntosFinales.append(pg_tramo_end0)
    with arcpy.da.UpdateCursor(seleccion, fields, where_clause= "flow = 0") as cursor1:
        #recorre los elementos buscando almacenando los puntos finales
        for row1 in cursor1:
            endPoint = row1[1].lastPoint
            endX_tramo = endPoint.X
            endY_tramo = endPoint.Y
            pg_tramo_end1 = arcpy.PointGeometry(arcpy.Point(endX_tramo, endY_tramo), arcpy.SpatialReference(32718))
            #Compara con el punto final de la otra linea
            for pnt in puntosFinales:
                valor1 = pg_tramo_end1.equals(pnt)
                #Sin coincide ambos puntos, se invierte la direccion de la linea
                if pg_tramo_end1.equals(pnt) == True:
                    vertices = []
                    for part in row1[1]:
                        for puntos in part:
                            vertices.append(puntos)
                    row1[1] = arcpy.Polyline(arcpy.Array(vertices[::-1]))
                    row1[2] = flow +1
                    row1[3] = 1
                    cursor1.updateRow(row1)
                else:
                    #De estar correcta la direccion, solo se registra que ya fue analizado
                    row1[2] = flow +1
                    row1[3] = 1
                    cursor1.updateRow(row1)
    puntosFinales =[]
```

**Figura 24**

*Dirección del flujo de la red establecida*



4) Calcular número de elementos por capas: es una herramienta de control que genera una tabla de información que tiene una columna con los nombres de las capas y en otra columna el número de elementos que existen en la capa es una herramienta intermedia de control de calidad. Posteriormente se usa para corroborar el resultado final. En la figura 24 se muestra el resultado de la herramienta:

**Figura 25**

*Tabla de lista de capas*

ListaCapas

| OBJECTID * | CAPA | NUMEROELEMENTOS |
|---|---|---|
| 1 | CODO | 2 |
| 2 | DERIVACION | 71 |
| 3 | ERM | 0 |
| 4 | ERP | 0 |
| 5 | GRM | 0 |
| 6 | LINEAINTERNA | 0 |
| 7 | NODOERP | 0 |
| 8 | NODOVIRTUAL | 5 |
| 9 | OBRAESPECIAL | 0 |
| 10 | ODORIZADOR | 0 |
| 11 | REDUCCION | 1 |
| 12 | TAPON | 25 |
| 13 | TRAMOGASODUCTO | 127 |
| 14 | TRAMPASCRAPPER | 0 |
| 15 | TRANSICION | 0 |
| 16 | TUBERIACONEXION | 0 |
| 17 | UNIONES | 0 |
| 18 | VALVULA | 3 |
| 19 | VALVULAEXCESOFLUJ | 0 |

Las próximas herramientas son procesos de herencia de datos extraídos de fuentes externas que por directrices deben ser validados entre sí, las herramientas creadas deben considerar estas validaciones. Si bien es cierto todas estas herramientas podrían ejecutarse en una misma herramienta, dada su importancia se decidió separarse. Las tareas que se ejecutan en estas herramientas son simples, pero que en la práctica si se hacen de forma manual el tiempo necesario es considerable.

5) Heredar código de proyecto: Hasta el momento el proyecto no tiene los datos identificativos a la etapa y al proyecto al que pertenece y es fundamental que los elementos cuenten con este dato ya que sirve posteriormente para heredar datos de otras fuentes. La información debe ser asignada a las 19 capas que componen la geodatabase de redes. El dato deber ser validado previamente con dos fuentes distintas, el Acta del Proyecto y Registro de Proyecto y debe de ser comparados entre sí, de coincidir se ejecuta los comandos y de no

coincidir se genera una alerta y se debe revisar la casuística. Estos pasos lo realizan la herramienta.

6) Heredar fecha de habilitación: Al igual que el código del proyecto, la fecha de habilitación se debe heredar a todas las capas, con una validación previa de las dos fuentes de información ya mencionadas, al igual que la herramienta pasada, de no pasar a la validación exitosamente, no ejecuta la herramienta, enviando una alerta para revisar la casuística. Estos pasos lo realizan la herramienta.

7) Heredar ubicación del proyecto: Esta herramienta usa procesos de herencia espacial entre información superpuesta. Lo que realiza es un proceso de extracción de información de los polígonos al que se superpone. Los códigos de ubicación que hereda son, el código de la malla, el código del sector y el código del distrito. La herramienta usa todas esas capas como parámetros para poder ejecutarse.

8) Heredar usuario de carga: Esta herramienta hereda códigos generados por cada usuario responsable, esta información debe ir a cada capa de información. Al igual que las herramientas anteriores realiza procesos de asignación de información a todas las capas.

9) Heredar códigos del PQ y PA: La construcción de la red obedece a una planificación que se realiza quinquenal y anualmente. Para hacer un seguimiento de estos planes, las nuevas redes deben heredar dicha información, el análisis se realiza sobre las tuberías y se hace de forma manual, la automatización viene en el proceso de herencia de las tuberías a los accesorios de la red.

10) Calcular campos de la red: La red tiene muchos campos inherentes a la geometría, al tipo de elemento, a la función que cumplen y a campos calculados a partir de otros campos. Son procesos sencillos en muchos de ellos y procesos que necesitan crear funciones para que puedan se ejecutados. El impacto de esta herramienta es significativo ya que ejecuta mas de

500 procesos empaquetados en cursores e iteradores. No se detalla este código debido a lo simple de las herramientas y lo extenso del código. Pero se adjunta en los anexos.

De aquí en adelante se inician los procesos de control de calidad general a los datos agregados. Son herramientas que permiten identificar posibles vicios del proceso.

11) Crear vistas de capas: Este proceso es bastante simple, crea vistas de las capas con los campos principales a revisar, es un proceso que ayuda a validar que haya heredado correctamente los datos a los campos obligatorios. Descarta de la vista los campos no importantes.

12) Crear campos de evaluación: Se crea dos campos a todas las capas donde se almacenará la información en caso haya error en ese registro, es un código bastante siempre, peor bastante útil.

13) Calcular errores básicos: Se recorre las columnas y las filas de todas las capas de la geodatabase con el objetivo de detectar campos vacíos que no hayan heredado datos y también valida que la información creada no tenga caracteres extraños. Hace un barrido simple por la información heredada en la geodatabase.

14) Calcular errores completos: A diferencia con la herramienta anterior que calcula solo los errores básicos, esta herramienta realiza una validación exhaustiva de la información ingresada y sea válida en cuanto a que respete los dominios y subtipos de la geodatabase. El recorrido es genera por cada elemento de la geodatabase.

15) Crear reportes de errores: Esta herramienta al igual que la anterior vuelve a validar la información, pero con el objetivo de generar un reporte a modo de resumen indicando en una columna si el elemento se puede cargar o no. Esta herramienta utiliza como base la tabla creada en la herramienta número cuatro. El resultado de la tabla se muestra en la figura 25:

**Figura 26**

*Tabla de reportes*

| | CAPA | NUMEROELEMENTOS | ELEMENTOSINCOMPLETOS | DOMINIOSINCORRECTOS | CARGAR |
|---|---|---|---|---|---|
| 1 | CODO | 2 | 0 | 0 | Si |
| 2 | DERIVACION | 71 | 0 | 0 | Si |
| 3 | ERM | 0 | 0 | 0 | NO |
| 4 | ERP | 0 | 0 | 0 | NO |
| 5 | GRM | 0 | 0 | 0 | NO |
| 6 | LINEAINTERNA | 0 | 0 | 0 | NO |
| 7 | NODOERP | 0 | 0 | 0 | NO |
| 8 | NODOVIRTUAL | 5 | 0 | 0 | Si |
| 9 | OBRAESPECIAL | 0 | 0 | 0 | NO |
| 10 | ODORIZADOR | 0 | 0 | 0 | NO |
| 11 | REDUCCION | 1 | 0 | 0 | Si |
| 12 | TAPON | 25 | 0 | 0 | Si |
| 13 | TRAMOGASODUCTO | 127 | 0 | 0 | Si |
| 14 | TRAMPASCRAPPER | 0 | 0 | 0 | NO |
| 15 | TRANSICION | 0 | 0 | 0 | NO |
| 16 | TUBERIACONEXION | 0 | 0 | 0 | NO |
| 17 | UNIONES | 0 | 0 | 0 | NO |
| 18 | VALVULA | 3 | 0 | 0 | Si |
| 19 | VALVULAEXCESOFLUJ | 0 | 0 | 0 | NO |

16) Eliminar campo de errores: Es una herramienta intermedia que elimina las columnas temporales de validación de datos.

17) Heredar código padre a la red: Después de añadir redes al sistema del cliente, cada nuevo elemento recibe un identificador único llamado CODIGOELEMENTO. En esta red jerárquica, cada elemento hereda los códigos de sus elementos padre. En resumen, el elemento padre es el que está más arriba en la jerarquía. En términos prácticos, para una nueva red que se conecta a la red existente, el elemento existente se convierte en el 'padre' del punto de conexión de la nueva red. Este punto de conexión, a su vez, se convierte en el 'padre' del siguiente elemento en la nueva red, y así sucesivamente a lo largo de toda la red. Debido a lo difícil que sería el proceso manual de esta actividad se muestra el código que lo soluciona.

Hasta la línea 14 de la figura 26 se establece el entorno de trabajo, se asigna a una variable los parámetros requeridos y se crea una geodatabase intermedia para almacenar los archivos intermedios. A continuación, se determinan el punto inicial y final de cada línea. Desde la línea 18 en adelante, se realizan varios geoprocesos. Estos procesos transfieren el código

CODIGOELEMENTO del punto final de una línea al punto inicial de la siguiente línea, y de ahí a la línea hija subsiguiente.

**Figura 27**

*Código fuente: Heredar código padre de la red*

```
import arcpy
#Configuracion del entorno de trabajo
arcpy.env.overwriteOutput = True
gdb = arcpy.GetParameterAsText(0)
#gdb = r"D:\Villareal\Scripts\ReplicaEsquema.gdb"
arcpy.env.workspace = gdb
capas = arcpy.ListFeatureClasses(feature_dataset='RedDeGas')
space = gdb[:-19]
#Creacion de una geodatabase temporal
tempGDB = space +"\\temp.gdb"
if arcpy.Exists(tempGDB):
    arcpy.Delete_management(tempGDB)
arcpy.CreateFileGDB_management(space, "temp")
tg = gdb+"\\RedDeGas\\TRAMOGASODUCTO"
#Puntos extremos de cada lineas
inicio = arcpy.FeatureVerticesToPoints_management(tg, tempGDB+"\\tg_point_inicio","START")
fin = arcpy.FeatureVerticesToPoints_management(tg, tempGDB+"\\tg_point_fin","END")
#Geoprocesos de herencia de datos
sj = arcpy.SpatialJoin_analysis(inicio, fin, tempGDB+"\\sj", join_operation ="JOIN_ONE_TO_ONE", match_option="INTERSECT")
tg_lyr = arcpy.MakeFeatureLayer_management(tg, "tg_lyr")
join_tg_original = arcpy.AddJoin_management(tg_lyr, "OBJECTID",sj, "ORIG_FID")
arcpy.CalculateField_management(join_tg_original, "TRAMOGASODUCTO.CODIGOPADRE", "[sj.CODIGOELEMENTO_1]")
```

18) Heredar código padre a los accesorios: Con la misma lógica del código padre de las tuberías, se hereda el código padre a los accesorios, siendo el padre de cada accesorio el elemento superior de la red. Se determina si un elemento es superior cuando el punto final de la línea coincide con la ubicación de los accesorios. En la figura 27 se establece el entorno de trabajo, se asigna la geodatabase de la red como parámetro, se almacena en una variable todas las capas, se crea la geodatabase temporal y se crea el punto final de cada línea.

**Figura 28**

*Código fuente: Heredar código padre de los accesorios*

```
import arcpy
#Configuracion del entorno de trabajo
arcpy.env.overwriteOutput = True
gdb = arcpy.GetParameterAsText(0)
#gdb = r"D:\Villareal\Scripts\ReplicaEsquema.gdb"
arcpy.env.workspace = gdb
capas = arcpy.ListFeatureClasses(feature_dataset='RedDeGas')
#Creacion
space = gdb[:-19]
tempGDB = space +"\\temp.gdb"
if arcpy.Exists(tempGDB):
    arcpy.Delete_management(tempGDB)
arcpy.CreateFileGDB_management(space, "temp")
tg = gdb+"\\RedDeGas\\TRAMOGASODUCTO"
end = arcpy.FeatureVerticesToPoints_management(tg, tempGDB+"\\tg_nodo_final", "END")
#Funcion que relaiza la operacion de asignacion del codigo padre a los accesorios
```

Debido a que el proceso por cada accesorio es repetitivo se creó una función llamada heredarPadreAccesorio() que recibe un parámetro una capa de puntos. La función toma esta capa y lo superpone con los puntos finales de la línea creado anteriormente. De coincidir hereda el código elemento en el campo de CODIGOPADRE del accesorio, el código se muestra en la figura 28.

**Figura 29**

*Código fuente: Heredar código padre de los accesorios*

```
#Funcion que relaiza la operacion de asignacion del codigo padre a los accesorios
def heredarPadreAccesorio(layer):
    layer_sj = arcpy.SpatialJoin_analysis(layer, end, tempGDB+"\\sj_nodo_fin", join_operation= "JOIN_ONE_TO_ONE",
    match_option="INTERSECT")
    arcpy.MakeFeatureLayer_management(layer, layer+"_lyr")
    join_lyr = arcpy.AddJoin_management(layer+"_lyr", "OBJECTID", layer_sj, "TARGET_FID")
    arcpy.CalculateField_management(join_lyr, "{}.CODIGOPADRE".format(layer), "[sj_nodo_fin.CODIGOELEMENTO_1]")
    arcpy.CalculateField_management(join_lyr, "{}.TIPOPADRE".format(layer), "TG")
    arcpy.RemoveJoin_management(layer+"_lyr", "sj_nodo_fin" )
```

Al ser una función no se ejecuta por sí misma, para ejecutarla la agregamos dentro de un iterador que recorra cada capa de accesorios siempre y cuando tenga elementos dibujados.

**Figura 30**

*Código fuente: Heredar código padre de los accesorios*

```
#Se itera por cada capa la fubncion, ante svalida que la capa tenga elementos
for capa in capas:
    numElementos = arcpy.GetCount_management(capa)
    arcpy.AddMessage(capa)
    if numElementos > 0 and (capa != "CODO" and capa != "POLIGONOETAPA" and capa != "TRAMPASCRAPPER"):
        arcpy.AddMessage(capa)
        heredarPadreAccesorio(capa)
```

En resumen, estas herramientas personalizadas desempeñan un papel crucial en la gestión eficiente y precisa de las nuevas redes de gas. Al automatizar tareas como la migración de datos, control topológico, establecimiento de direcciones, herencia de códigos y validación de la información, se logra una considerable mejora en la calidad de los datos y se reducen significativamente los tiempos de procesamiento. La capacidad de identificar y corregir errores topológicos y de datos de manera automática no solo mejora la fiabilidad de la red, sino que también proporciona una base sólida para futuras expansiones y mantenimientos. Este enfoque integral y automatizado es fundamental para enfrentar los desafíos de una infraestructura de gas cada vez más compleja y demandante.

## 2.12 Resultados de la automatización

### *2.12.1 Productos obtenidos de la automatización*

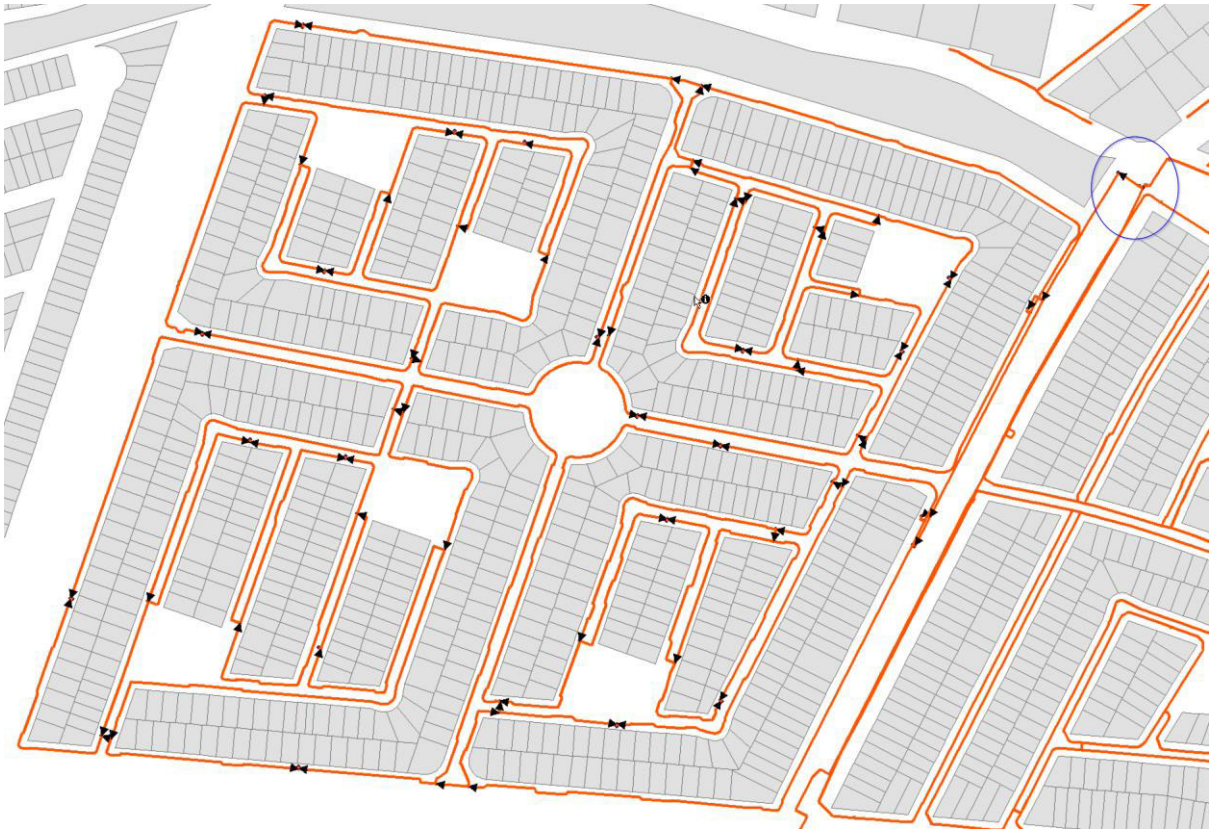
El resultado del uso de las herramientas principalmente son tres, el primer resultado es una geometría libre de errores topológicos, la segunda es la dirección de las tuberías correctas según las reglas establecidas y la herencia de los datos atributivos correspondiente. Para el primer caso se usó las dos primeras herramientas que primero estandarizan la geodatabase y la segunda detecta los errores de superposición y de elementos desconectado. Este proceso es fundamental ya que la dirección del flujo de la red depende de una correcta topología.

El segundo resultado es una red con los flujos correctos, es decir que la representación de lo construido simula la distribución del gas en lo real, en la imagen se puede observar que

el flujo se abastece de la red existente con un flujo correcto. La correcta asignación del flujo permite que el sistema jerárquico de la red padre-hijo se pueda asignar de manera correcta.

**Figura 31**

*Red agregada a la base de datos del cliente*



Finalmente se tiene asignado los datos completos a todas las capas de información de la red, por temas prácticos y privacidad de los datos, solo se muestra una figura con los datos completos de una de las capas. La información que se muestra en la imagen número 31 fue asignada por las herramientas de heredarían de datos, ningún dato de los 1058 campos fue asignado de manera manual.

**Figura 32**

*Tabla de atributos completa*

| CODIGOELEMENTO | CODIGOPROYECTO | CODIGOETAPA | TIPOELEMENTO | CODIGOPADRE | TIPOPADRE | DIAMETRO | MATERIAL | ESPESOR | RESISTENCIA | TARIFA | PROPIEDAD | TIPOTERRENO | MAPO | TIPOPAVIMENTO | CODIGODISTRITO | ESTADO | FECHAPRUEBA | FECHAPUESTASERVICIO | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 190180 | P/10/267 | P/10/267/00 | Tuberia de Conexion | 204233 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Rigido | SAN JUAN DE LURIGANCHO | En servicio | <Null> | 24/05/2011 | <N |
| 229979 | P/10/273 | P/10/273/00 | Tuberia de Conexion | 229958 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Rigido | VILLA MARIA DEL TRIUNFO | En servicio | 2/12/2011 | 6/12/2011 | <N |
| 230014 | P/10/273 | P/10/273/00 | Tuberia de Conexion | 229958 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Rigido | VILLA MARIA DEL TRIUNFO | En servicio | 2/12/2011 | 7/06/2012 | <N |
| 230015 | P/10/273 | P/10/273/00 | Tuberia de Conexion | 229958 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Rigido | VILLA MARIA DEL TRIUNFO | En servicio | 2/12/2011 | 6/12/2011 | <N |
| 230017 | P/10/273 | P/10/273/00 | Tuberia de Conexion | 229958 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Rigido | VILLA MARIA DEL TRIUNFO | En servicio | 2/12/2011 | 6/12/2011 | <N |
| 230018 | P/10/273 | P/10/273/00 | Tuberia de Conexion | 229958 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Rigido | VILLA MARIA DEL TRIUNFO | En servicio | 2/12/2011 | 6/12/2011 | <N |
| 230052 | P/10/273 | P/10/273/00 | Tuberia de Conexion | 229958 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Rigido | VILLA MARIA DEL TRIUNFO | En servicio | 2/12/2011 | 6/12/2011 | <N |
| 230056 | P/10/273 | P/10/273/00 | Tuberia de Conexion | 229958 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Rigido | VILLA MARIA DEL TRIUNFO | En servicio | 2/12/2011 | 6/12/2011 | <N |
| 245760 | P/09/254 | P/09/254/00 | Tuberia de Conexion | 199040 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Rigido | SAN JUAN DE LURIGANCHO | En servicio | 31/10/2011 | 19/10/2011 | <N |
| 245761 | P/09/254 | P/09/254/00 | Tuberia de Conexion | 199040 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Rigido | SAN JUAN DE LURIGANCHO | En servicio | 31/10/2011 | 24/08/2011 | <N |
| 267061 | P/09/246 | P/09/246/01 | Tuberia de Conexion | 207979 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Flexible | SAN JUAN DE LURIGANCHO | En servicio | 31/12/2011 | 8/11/2011 | <N |
| 267165 | P/09/248 | P/09/248/01 | Tuberia de Conexion | 207979 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Flexible | SAN JUAN DE LURIGANCHO | En servicio | 31/12/2011 | 19/12/2011 | <N |
| 276571 | P/10/275 | P/10/275/00 | Tuberia de Conexion | 1974712 | Tramo Gasoducto | 20 mm | Polietileno | 3.00 mm | PE80 | Tarifa Única | Distribuidora | Semirocoso | 5 Bar | Rigido | VILLA MARIA DEL TRIUNFO | En servicio | <Null> | 16/12/2011 | <N |

### *2.12.2 Tiempos y recursos*

Dos de los resultados claves del proceso de automatización es la disminución en el tiempo y los recursos requeridos para realizar las actividades específicas. Medir estas dos variables nos dan una idea clara y tangible de cuan efectivas llegan a ser las herramientas de automatización creadas. Para medir estos indicadores se parte de la información obtenida previo al proceso de creación de las herramientas y se mide con los resultados posterior al uso de la herramienta.

Para medir los tiempos se calculó en base a un proyecto promedio tomando las medidas cuando se hacían las actividades de manera manual, y los tiempos que se tomaron después de la creación de las herramientas de automatización, haciendo una simple diferencia se obtiene el resultado. En la tabla tres, se muestra dos columnas comparativas con los tiempos en minutos y por actividad, notando que en promedio se reduce el 91.5% del tiempo por cada actividad. En la figura de 32 se muestra de una forma más visual la disminución de tiempo por actividad.

**Tabla 3**

***Tiempos antes y después de la automatización de los procesos***

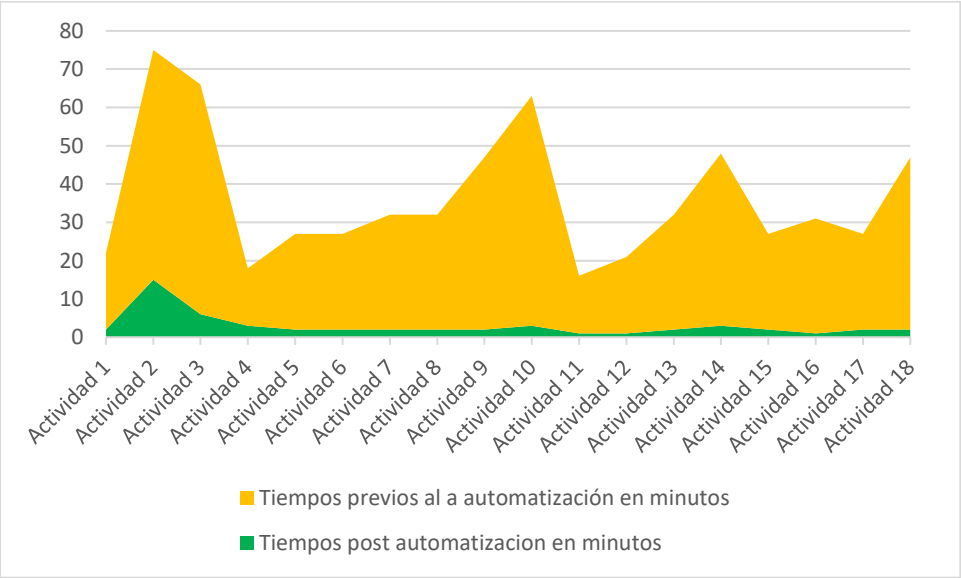
| Denominación | Actividad | Tiempos previos al a automatización en minutos | Tiempos post automatización en minutos | Reducción en % |
|---|---|---|---|---|
| Actividad 1 | Migrar geodatabase correcta | 20 | 2 | 90.0 |
| Actividad 2 | Control topológico | 60 | 15 | 75.0 |
| Actividad 3 | Establecer dirección | 60 | 6 | 90.0 |
| Actividad 4 | Calcular número de elementos por capas | 15 | 3 | 80.0 |
| Actividad 5 | Heredar código de proyecto | 25 | 2 | 92.0 |
| Actividad 6 | Heredar fecha de habilitación | 25 | 2 | 92.0 |
| Actividad 7 | Heredar ubicación del proyecto | 30 | 2 | 93.3 |
| Actividad 8 | Heredar usuario de carga | 30 | 2 | 93.3 |
| Actividad 9 | Heredar códigos del PQ y PA | 45 | 2 | 95.6 |

| | | | | |
|---|---|---|---|---|
| Actividad 10 | Calcular campos de la red | 60 | 3 | 95.0 |
| Actividad 11 | Crear vistas de capas | 15 | 1 | 93.3 |
| Actividad 12 | Crear campos de evaluación | 20 | 1 | 95.0 |
| Actividad 13 | Calcular errores básicos | 30 | 2 | 93.3 |
| Actividad 14 | Calcular errores completos | 45 | 3 | 93.3 |
| Actividad 15 | Crear reportes de errores | 25 | 2 | 92.0 |
| Actividad 16 | Eliminar campo de errores | 30 | 1 | 96.7 |
| Actividad 17 | Heredar código padre a la red | 25 | 2 | 92.0 |
| Actividad 18 | Heredar código padre a los accesorios | 45 | 2 | 95.6 |
| | Totales | 605 | 53 | 91.5 |

En la figura 33 se muestra de color amarillo los tiempos previos a la automatización de los procesos y de color verde los tiempos posteriores a los procesos de automatización.

**Figura 33**

*Tiempos previos y post automatización*




Debido a la confidencialidad de los datos en cuanto a los recursos usado para cumplir con las actividades, no se puede describir la comparación de los datos, pero la proporción de

disminución de los recursos humanos, de hardware y software está en la misma línea que los tiempos estimados, es decir también se redujo en un 90 %.

## III. Aportes destacados a la empresa

El aporte significativo a la empresa se evidencia mediante la ejecución exitosa de todas las tareas de carga de información vinculadas a las redes de gas, tal como lo ha requerido el cliente, en este caso, la empresa encargada de la distribución del gas natural en Lima Metropolitana. No obstante, el valor más destacado reside en la automatización de los procesos SIG, abarcando no solo el proyecto detallado en este informe, sino también diversas iniciativas asignadas a la empresa.

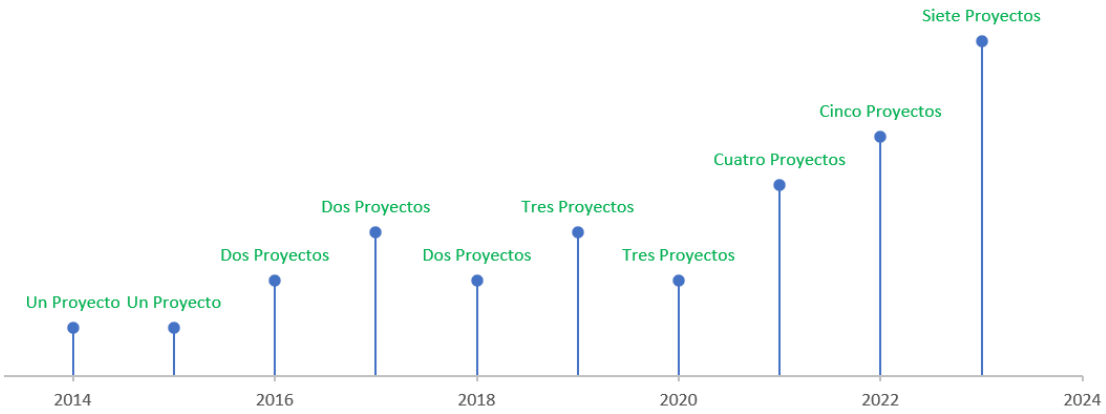
La automatización de los procesos SIG es el principal aporte a la empresa debido a que impacto significativamente en la mejora operativa y la eficiencia organizacional, lo que repercutió positivamente en el rendimiento y competitividad de la empresa. La automatización redujo la carga de trabajo manual al personal técnico, esto no solo los libero de tiempo, si no también minimizo errores humanos, lo que mejoro la precisión y calidad de los resultados.

La reducción de los tiempos es un aporte derivado de la automatización de las actividades, porque acelero los ciclos de trabajo, reduciendo los plazos de entrega de los productos finales, mejorando la capacidad de respuesta frente al cliente. Otro aporte es la reducción de costos ya que disminuyo la necesidad de recursos humanos, de hardware y software optimizando los recursos disponibles permitiendo ser más competitivos frente a otras propuestas en las licitaciones.

Producto de la automatización operativa de los procesos SIG se pudieron adjudicar más proyectos relacionados con procesos SIG y redes de gas con el cliente, en la siguiente figura se muestra la evolución del número de los proyectos asignados al largo de estos nueve años, cabe señalar que todas las adjudicaciones fueron por concurso, siendo la automatización de procesos clave en los precios competitivos. En la figura 34 se muestra el incremento de los proyectos a lo largo del tiempo, evidenciándose el incremento.

**Figura 34**

*Evolución de la cantidad de proyectos por años*




En mi posición actual de Supervisor SIG de todos los proyectos, asumo la responsabilidad del correcto funcionamiento de las actividades. Mi rol abarca mucho más allá de la supervisión de las actividades, se extiende hacia la creación de herramientas para la automatización de las tareas de cada proyecto. En este punto mi aporte tangible es el desarrollo de más de ochenta herramientas que ayudaron a automatizar las actividades de todos los proyectos.

**Tabla 4**

*Herramientas creadas*

| PROYECTO | HERRAMIENTAS |
|---|---|
| Revisión de planchetas | 13 |
| Digitalización de Tuberías de Conexión | 25 |
| Carga de redes | 18 |
| Asignación de frente a red | 5 |
| Reporte VNR | 2 |
| Seguimiento PQ/PA | 5 |
| Asistencia técnica | 12 |
| **TOTAL** | **80** |

# IV. Conclusiones

Se automatizaron las tareas de validación y carga de redes mediante la creación de dieciocho herramientas personalizadas escritas sobre ArcPy y Python. Estas herramientas creadas cubren el 80% de las actividades necesarias para cumplir con el proceso completo. Las herramientas creadas son de uso recurrente que se adaptaron a las actividades requeridas y que a su vez son dinámicas, lo que es valorado ya que las directrices de validación suelen cambiar en el tiempo dependiendo de las necesidades del cliente.

Producto del desarrollo de las herramientas personalizadas se pudo procesar toda la información de manera sistemática lo que ayudo a mejorar la calidad de los datos subidos, excluyendo el factor humado del proceso. Esto se logró debido que en el proceso se agregaron controles de calidad automatizados que recorren las capas validando que la información subida sea completa y valida.

La automatización permitió reducir los tiempos de ejecución de las actividades necesarias para cumplir con el proceso completo de validación de las nuevas redes de gas, pasando de 605 minutos por proyecto a 53 minutos por proyecto, esta disminución del 91.5 % permitió al personal encargado poder incrementar esfuerzos en el control de calidad y de esta forma mejorar la calidad del trabajo entregado. La reducción de tiempos también vino trajo por consiguiente la reducción de los recursos de personal técnico, software y hardware lo que es beneficioso para la empresa debido a que puede mantener precios competitivos y así poder sostener el servicio en los diversos procesos de licitación desde el año 2014 y su vez asignarse nuevos proyectos.

# V. Recomendaciones

Se sugiere realizar una evaluación exhaustiva del rendimiento de las herramientas de automatización que se han desarrollado para la validación y carga de redes. Esto surge de la necesidad de optimizar continuamente su eficiencia ya que se añaden nuevos procesos en el camino. Al analizar y medir el desempeño de estas herramientas, se pueden identificar áreas de mejora que, una vez abordadas, podrían resultar en una mayor eficacia operativa. Este proceso de evaluación no solo busca mantener los estándares de calidad ya alcanzados, sino también explorar maneras de acelerar aún más el tiempo de validación.

Al mejorar continuamente las herramientas de automatización, no solo se consigue una reducción adicional en los tiempos de validación, sino que también se potencia la eficiencia general del proceso. Esto, a su vez, puede conducir a una mayor satisfacción del cliente y a un mejor posicionamiento competitivo en el mercado. En resumen, la evaluación y mejora constante del rendimiento de las herramientas automatizadas es un componente crítico para el éxito y la sostenibilidad a largo plazo de cualquier proyecto de automatización-Es necesario explorar opciones de software libre compatibles con los softwares requeridos por el cliente, que garantice la perfecta compatibilidad con sus sistemas internos, pero que reduzca costos a la empresa en el número de licencias.

Se recomienda explorar alternativas de uso gratuito a los softwares usados actualmente, debido a que los costos de los mismo representan gran parte del presupuesto destinado en la realización de las actividades. Explorar y adoptar software gratuito podría significar una reducción significativa en los gastos operativos.

## VI. Referencia

Bell, M. G., & lida, Y. (1977). *Transportation Network Analysis.*

Cooper Crispin, H. V., & Chiaradia Alain, J. F. (2020). sDNA: 3-d spatial network analysis for GIS, CAD, Command Line & Python. *SoftwareX*. doi:10.1016/j.softx.2020.100525

Curtin, K. M. (2013). Network Analysis in Geographic Information Science:. *Cartography and Geographic Information Science , 34(2),*, 103-111.

ESRI. (2021). *ArcMap*. Obtenido de ¿Qué es ArcPy?: https://desktop.arcgis.com/es/arcmap/latest/analyze/arcpy/what-is-arcpy-.htm

ITICPERU. (2023). *ITICPERU*. Obtenido de ITICPERU: https://iticpe.com/empresa/pilares

Mejia Del Carpio, M. (2022). *Memoria Anual 2022 CALIDDA.* Obtenido de https://www.calidda.com.pe/media/jscegsie/memoria-anual-calidda-2022-vf.pdf

Molina Herrera, D. E., & Rodríguez Vivas, M. C. (2019). *CODE-ID: Herramienta de ArcGis para la automatización del proceso de análisis para la asignación de códigos SIG a predios.*

Torrent Foz, E., Muñoz-Nieto, A. L., Gonzales Aguilera, D., & Rodriguez Gonzálvez, p. (2021). *Implementación de procesos de control de calidad en la actualización de series cartográficas urbanas mediante combinación de CAD y SIG.* Ciudad de México: Revista cartográfica. doi:https://doi.org/10.35424/rcarto.i103.884

# VII. Anexos

Se detalla el código fuente de cada una de las herramientas elaboradas.

a. Migración a una geodatabase: La empresa constructora envía el diseño de la red en una copia de la base de datos principal, pero debido a los cambios constantes que se realiza a la base de datos, esta puede quedar desactualizada, es por ello que se migra a una base de datos estandarizada con el esquema y que sirve de base para que las herramientas creadas en actividades posteriores funcionan.

   El código necesario para que haga la migración es la siguiente:

**Figura 35**

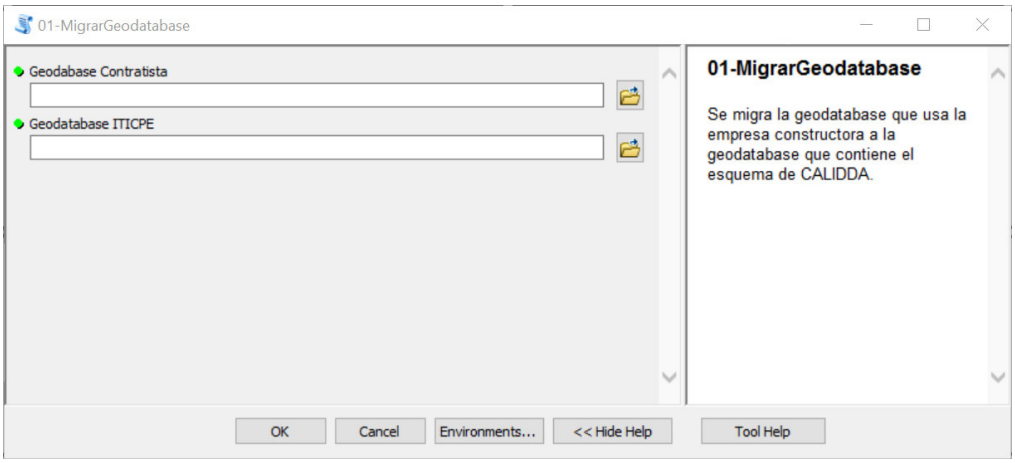
*Código fuente: Migrar de geodatabase*

```
import arcpy
# Migrar GDBs
# La Geodatabase original enviada por la empresa constructora
arcpy.env.workspace = arcpy.GetParameterAsText(0)

gdbDestino = arcpy.GetParameterAsText(1)
# Se enlista las capas que existen en geodabase original
capas = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")

noTest = "NO_TEST"
# bucle que se ejecutara por cada capa que existe dentro de la GDB
for capa in capas:
    gdbObjetivo = gdbDestino+"\\"+capa
    if arcpy.Exists(gdbObjetivo):
        # Se ejecuta la herramienta para agregar una capa a otra, resptando los nombres
        arcpy.Append_management(capa, gdbObjetivo, noTest)
        arcpy.AddMessage("Se migro la capa: "+ capa +"...")
```

La interfaz tiene la siguiente apariencia donde los parámetros que solicita es la ruta de la geodatabase de la contratista y la geodatabase que tiene los esquemas.

**Figura 36:**

***Interfaz: Migrar de Geodatabase***




b. Validación de topología: Se valida que la tubería esté correctamente dibujada, para ello, la herramienta ayuda a identificar los elementos superpuestos mediante la creación de la regla topológica, el script es capaz de alertar si existen errores en la ventana de resultados. Los errores identificados se almacenan en la geodatabase temporal con los elementos que contienen error para su fácil identificación y corrección. Las observaciones se corrigen manualmente.

   En la primera parte del código configuramos el espacio de trabajo, creamos las variables donde se almacenarán las topologías, enlistamos las capas donde se va a validar la topología de los elementos y verificamos que la geodatabase que almacena los elementos temporales no exista, si existe la elimina.

**Figura 37**

*Código fuente: Validación de topología*

```
import arcpy
# Especifica el directorio de trabajo y los nombres de las capas de entrada
gdb = arcpy.GetParameterAsText(0)
arcpy.env.workspace = gdb
dataset = gdb+"\\RedDeGas"
topo = dataset+"\\topologia_lineas"
topo_puntos = dataset + "\\topologia_puntos"

#Enlista todas las capas de toda la geodatabase y hace referencia a la capa de la tuberia
lyrs = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")
tg = lyrs[15]
#Enlista todas las capas que son de tipopunto
capas_puntos = [lyr for lyr in lyrs if arcpy.Describe(lyr).shapeType == "Point"]
space = gdb[:-19]

#Verifica que que la gdb que alberga las capas intermedias no exista, si existe, la elimina
#Posteriormente la crea
tempGDB = space +"\\temp.gdb"
if arcpy.Exists(tempGDB):
    arcpy.Delete_management(tempGDB)
arcpy.CreateFileGDB_management(space, "temp")
```

Seguidamente se crea la regla topológica de superposición de líneas, el código crea, valida la regla topológica y exporta los errores en la geodatabase temporal, para el caso de elementos de tipo punto, al ser muchos, lo que se hace es iterar sobre cada uno de ellos y asignándoles la regla a cada capa. Al igual que con las líneas exporta los resultados en la geodatabase temporal. Esta parte del código solo analiza superposiciones entre los elementos de una misma capa.

**Figura 38**

*Código fuente: Validación de topología*

```
#Crea una topologia para la capa de entrada
output_topology = "topologia_lineas"
arcpy.CreateTopology_management(dataset, output_topology)
arcpy.AddFeatureClassToTopology_management(topo, tg)
arcpy.AddRuleToTopology_management(topo, "Must Not Overlap (Line)", tg)
arcpy.ValidateTopology_management(topo, "Full_Extent")
arcpy.ExportTopologyErrors_management(topo, tempGDB, "TablaDeSuperposiciones")

#Se Itera sobre cadas de puntos y crea topología para cada una
for capa_puntos in capas_puntos:
    topo_puntos = "{}\\topologia_{}".format(dataset, capa_puntos)
    arcpy.CreateTopology_management(dataset, "topologia_{}".format(capa_puntos))
    arcpy.AddFeatureClassToTopology_management(topo_puntos, capa_puntos)
    arcpy.AddRuleToTopology_management(topo_puntos, "Must Not Overlap (Point)", capa_puntos)
    arcpy.ValidateTopology_management(topo_puntos, "Full_Extent")
    arcpy.ExportTopologyErrors_management
    (topo_puntos, tempGDB, "TablaDeSuperposiciones_{}".format(capa_puntos))
```

También se debe validar que no haya superposición de los puntos entre las diferentes capas, para ellos se agrego el siguiente código, que básicamente comparara la ubicación de un punto con el resto de los puntos de la geodatabase, exportando los resultados en la geodatabase temporal.

**Figura 39**

*Código fuente: Validación de topología*

```
# Verifica la superposición entre todas las capas de puntos
for i in range(len(capas_puntos)):
    for j in range(i + 1, len(capas_puntos)):
        capa1 = capas_puntos[i]
        capa2 = capas_puntos[j]
        topo_capa1 = "{}\\topologia_{}".format(dataset, capa1)
        arcpy.AddFeatureClassToTopology_management(topo_capa1, capa2)
        arcpy.AddRuleToTopology_management(topo_capa1, "Must Not Overlap (Point)", capa2)
        arcpy.ValidateTopology_management(topo_capa1, "Full_Extent")
        arcpy.ExportTopolcgyErrors_management(topo_capa1, tempGDB, "TablaDeSuperposiciones_{}_{}".format(capa1, capa2))
```

Finalmente, se añadió código para que, si en caso encuentre errores en el proceso, pueda alertar en la ventana de progreso de la herramienta.

**Figura 40**

*Código fuente: Validación de topología*

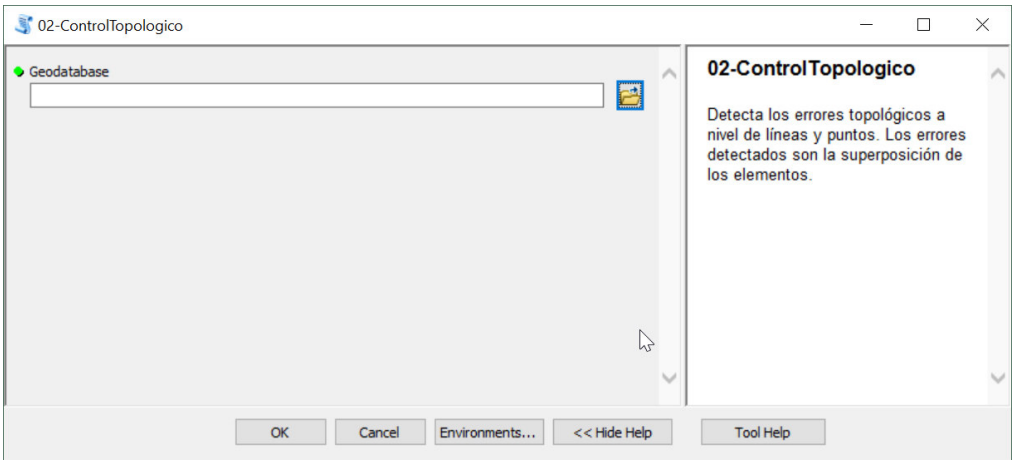
```
#Alerta de los errores en la ventada de progreso de la tuberia
count_lineas = arcpy.GetCount_management(tempGDB + "\\TablaDeSuperposiciones_line")
mensaje_lineas = "Hay {} errores de topología en la capa de líneas.".format(count_lineas)
arcpy.AddMessage(mensaje_lineas)

#Alerta cada caso para las capa lineas
for capa_puntos in capas_puntos:
    count_puntos = arcpy.GetCount_management(tempGDB + "\\TablaDeSuperposiciones_{}".format(capa_puntos))
    mensaje_puntos = "Hay {} errores de topología en la capa de puntos {}.".format(count_puntos, capa_puntos)
    arcpy.AddMessage(mensaje_puntos)
#Se elimina los elementos temporales creados
if arcpy.Exists(topo):
    arcpy.Delete_management(topo)
arcpy.AddMessage("Revisa cada caso")
for capa_puntos in capas_puntos:
    topo_puntos = "{}\\topologia_{}".format(dataset, capa_puntos)
    if arcpy.Exists(topo_puntos):
        arcpy.Delete_management(topo_puntos)
```

(variable) tempGDB: Any

La interfaz tiene la siguiente apariencia donde el único parámetro que solicita es la ruta de la geodatabase.

**Figura 41**

*Interfaz: Validación de topología*




c. Asignación del flujo de la red: La red tiene reglas de dirección, que se puede simplificar en que va de la troncal hacia la red habilitada, los accesorios también definen la dirección de la red, siendo los mas simples codo y tapón, pero por ejemplo las DERIVACINES (accesorios en forma de T) tienen como regla dividir el flujo en dos direcciones a modo de ramificación. Debido a que el código cuenta con 211 líneas de programación, solo se mostrara las partes más importantes, el código completo se adjuntara en los anexos.

En las primeras líneas se definirán los parámetros necesarios para ejecutar las herramientas, en este caso son tres parámetros requeridos, luego se configura el espacio de trabajo, las capas necesarias para estos procesos son asignados en las variables.

**Figura 42**

*Código fuente: Asignación del flujo de la red*

```
import arcpy
# recibe tres parametros, la geodatabase, el id del elemento que se empalma a la red
#parametro del accesorio de empalme ala red principal
gdb = arcpy.GetParameterAsText(0)
idTramo = arcpy.GetParameterAsText(1)
idDerivacion = arcpy.GetParameterAsText(2)
#Se define el entorno de trabajo
arcpy.env.workspace = gdb
arcpy.env.overwriteOutput = True
capas = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")
#Se asigna a variables la capas que se usaran
tramo = capas[15]
tapon = capas[14]
derivacion = capas[3]
valvula = capas[0]
#Se agrega los campos que ayudaran con el proceso, se asigna como 0 por defecto
arcpy.AddField_management(tramo, "flow","SHORT")
arcpy.AddField_management(tramo, "work","SHORT")
arcpy.CalculateField_management(tramo, "flow", 0)
arcpy.CalculateField_management(tramo, "work", 0)
```

Se crearon cuatro funciones que le dan dirección a toda la red, la primera función da la dirección al primero elemento de la red, es necesario este paso porque es el elemento de empalme a la red. La segunda función establece la dirección a todas las tuberías que terminan en tapón, la tercera función establece la dirección a los elementos anteriores a la tubería que finaliza en tapón, la última función asigna la dirección a los elementos restantes.

**Figura 43**

*Código fuente: Asignación del flujo de la red*

```
#Se crea la primera funcion que le da la direccion a la primera linea
def flipFirstLine (tramoInicial, fields, derivacionInicial, fieldsPoints):···

flipFirstLine (tramoInicial, fields, derivacionLayer, fieldsPoints)
#Se da la direccion a las tuberias que terminan en tapon
def flipLineTapon(tapon, fieldsPoints, tramo_tapon, fields):···

tramo_work_tapon = arcpy.MakeFeatureLayer_management(tramo, "tramo_tampor_ly")
tramo_tapon = arcpy.SelectLayerByLocation_management(in_layer=tramo_work_tapon, overlap_type="INTERSECT", select_

flipLineTapon(tapon, fieldsPoints, tramo_tapon, fields)
#Se da la direccion a los elementos anteriores a los tapones
def flipNearTapon(tramo_near_999, fields, tramo_near_tapon_work):···

tramo_near_999 = arcpy.MakeFeatureLayer_management(tramo, "tramo_tapon_n", "work = 999")
tramo_near_tw = arcpy.SelectLayerByLocation_management(tramoLayer, "INTERSECT", tramo_near_999)
tramo_near_tapon_work = arcpy.MakeFeatureLayer_management(tramo_near_tw, "tramo_near_ly", "work NOT IN (1, 999)")

flipNearTapon(tramo_near_999, fields, tramo_near_tapon_work)
#Asigna la direccion para toda la red
def flipOneLine (tramo_trabajo, seleccion, fields):···
```

Las cuatro funciones son muy parecidas entre ellas, pero difieren en los elementos a los que asignan la dirección, por ellos solo se explicara uno para entender el funcionamiento. La función flipOneLine() esta compuesto por dos cursores, el primer cursor almacena en una variable todos los puntos finales de la líneas ya con dirección, el segundo cursor almacena los puntos finales de las líneas que no tienen asignado la dirección, luego comparar si tienen la misma ubicación con los puntos finales de las líneas que ya tienen asignado la dirección, si coinciden estos se invierten de dirección. El análisis es progresivo, y en una columna se va asignando un identificador que precisa que ese elemento ya fue analizado para que no vuelva a ser analizado.

**Figura 44**

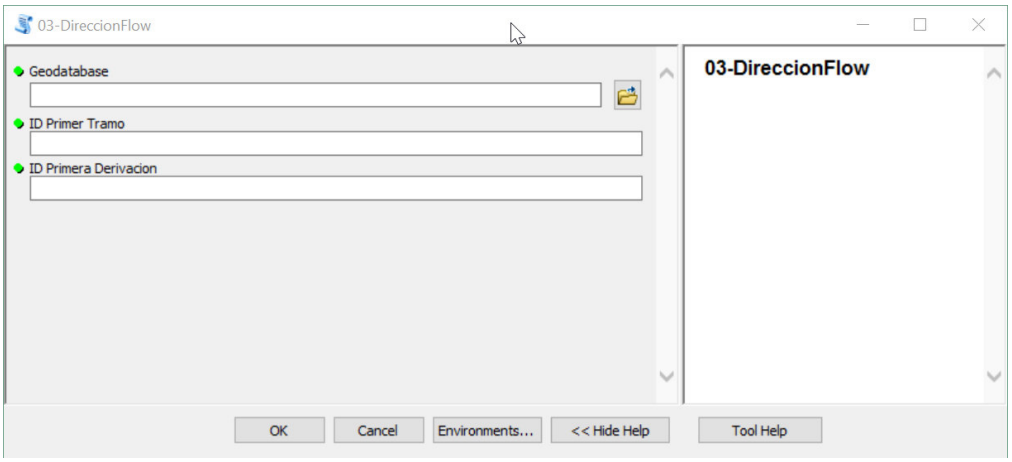
*Código fuente: Asignación del flujo de la red*

```
def flipOneLine (tramo_trabajo, seleccion, fields):
    puntosFinales =[]
    #crea el curso para todos los elementos que aun no fue establecido la direcion
    with arcpy.da.SearchCursor(tramo_trabajo, fields, where_clause= "flow > 0") as cursor0:
        #Recorre cada linea almacenando los puntos iniciales
        for row0 in cursor0:
            endPoint1 = row0[1].lastPoint
            endX_tramo = endPoint1.X
            endY_tramo = endPoint1.Y
            ultimoPunto_tramo1 = arcpy.Point(endX_tramo, endY_tramo)
            pg_tramo_end0 = arcpy.PointGeometry(ultimoPunto_tramo1, arcpy.SpatialReference(32718))
            puntosFinales.append(pg_tramo_end0)
    with arcpy.da.UpdateCursor(seleccion, fields, where_clause= "flow = 0") as cursor1:
        #recorre los elementos buscando almacenando los puntos finales
        for row1 in cursor1:
            endPoint = row1[1].lastPoint
            endX_tramo = endPoint.X
            endY_tramo = endPoint.Y
            pg_tramo_end1 = arcpy.PointGeometry(arcpy.Point(endX_tramo, endY_tramo), arcpy.SpatialReference(32718))
            #Compara con el punto final de la otra linea
            for pnt in puntosFinales:
                valor1 = pg_tramo_end1.equals(pnt)
                #Sin coincide ambos puntos, se invierte la direccion de la linea
                if pg_tramo_end1.equals(pnt) == True:
                    vertices = []
                    for part in row1[1]:
                        for puntos in part:
                            vertices.append(puntos)
                        row1[1] = arcpy.Polyline(arcpy.Array(vertices[::-1]))
                        row1[2] = flow +1
                        row1[3] = 1
                        cursor1.updateRow(row1)
                else:
                    #De estar correcta la direccion, solo se registra que ya fue analizado
                    row1[2] = flow +1
                    row1[3] = 1
                    cursor1.updateRow(row1)
```

La interfaz de la herramienta tiene la siguiente apariencia y pide los tres parámetros requeridos:

**Figura 45**

*Interfaz: Asignación del flujo de la red*



d. validación de número de elementos: Esta herramienta copia una tabla de un repositorio, dicha tabla tiene registrado el nombre de todas las capas y el objetivo es anotar el valor de número de elementos por cada capa.

**Figura 46**

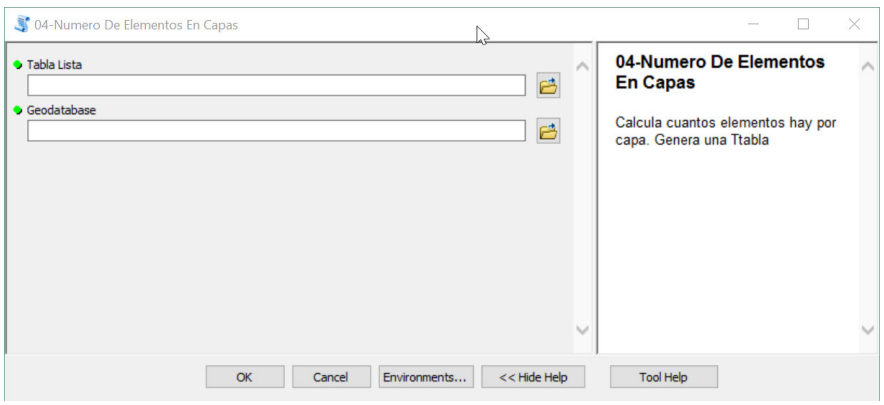
*Código fuente: Validación del número de elementos*

```
import arcpy
(variable) fc_in: Any

fc_in = arcpy.GetParameterAsText(0)
#Campos de la tabla que se trabajaran
fields = ['CAPA', 'NUMEROELEMENTOS', 'ELEMENTOSINCOMPLETOS', 'DOMINIOSINCORRECTOS', 'CARGAR']
#Geodatbase donde se trabajara
gdb = arcpy.GetParameterAsText(1)
arcpy.env.workspace = gdb
#Ruta de la tabla donde se va a copiar
gdb_tabla = gdb + '\\ListaCapas'
#Eliminar la tabla SI existe
if arcpy.Exists(gdb_tabla) == True:
    arcpy.Delete_management(gdb_tabla)
#Se copia la tabla la geodatabase
arcpy.Copy_management(fc_in, gdb_tabla)
#Lista de capas
capas = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")
#Recorre toda la lista
for capa in capas:
    erroresElementos = arcpy.GetCount_management(capa)
    resultElementos = int(erroresElementos.getOutput(0))
    #Actualiza la tabla por cada fila
    with arcpy.da.UpdateCursor(gdb_tabla, fields) as cursor:
        for row in cursor:
            if row[0] == capa:
                row[1] = resultElementos
            cursor.updateRow(row)
```

La interfaz de la herramienta tiene la siguiente apariencia y pide los tres parámetros requeridos:

**Figura 47**

*Interfaz: Validación del número de elementos*



e. Heredar el código de proyecto: Se tienen ## capas de información a los cuales se le tiene que asignar el CODIGOPROYECTO y el CODIGOETAPA, además se valida que el código del proyecto que se registra en el acta de barrido coincida debe coincidir con el registro de proyectos. De no coincidir alertara sobre el error.

**Figura 48**

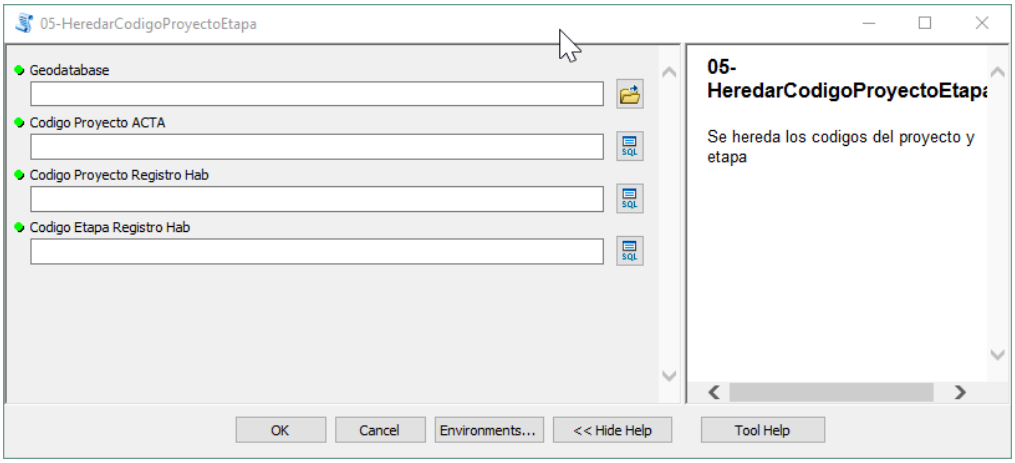
*Código fuente: Heredar el código del proyecto*

```
import arcpy
#Se asigna los parametros necesarios
arcpy.env.workspace = arcpy.GetParameterAsText(0)
gdb = arcpy.env.workspace
proyectoActa = arcpy.GetParameterAsText(1)
proyectoRegistro = arcpy.GetParameterAsText(2)
etapaRegistro = arcpy.GetParameterAsText(3)
arcpy.AddMessage(len(proyectoRegistro))

#Se hace control de calidad a los objetos escritos
pA = proyectoActa.replace(" ", '')
pR = proyectoRegistro.replace(" ", '')
eR = etapaRegistro.replace(" ","")
#Se compara los los diferentes codigis
capas = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")
if len(pR) <= 14 and len(eR) <= 17:
    for capa in capas:
        if pA == pR:
            arcpy.AddMessage("Se esta asignando los codigos de proyecto y etapa a: " + capa + "...")
            arcpy.CalculateField_management(in_table=capa, field="CODIGOPROYECTO", expression=pA,
            expression_type="VB", code_block="")
            arcpy.CalculateField_management(in_table=capa, field="CODIGOETAPA", expression=eR, expression_type="VB",
            code_block="")
        #Si hay diferencias avisa en en la ventana de los resultados
        elif pA != pR:
            arcpy.AddMessage("LOS CODIGOS NO COINCIDEN, POR FAVOR DIGITA BIEN LOS CODIGO")

```

La interfaz de la herramienta tiene la siguiente apariencia y solicita los cuatro parámetros:

**Figura 49**

*Interfaz: Heredar el código del proyecto*



f. Heredar fecha de habilitación: Con la misma lógica que con la herramienta anterior, se valida y hereda la fecha de habilitación de la red, para ello primero compara la fecha de las dos fuentes posibles, no ejecuta si ambas fechas no son iguales, además no todas las capas requieren este dato, por eso se insertó un código que excluyen a dichas capas.

**Figura 50**

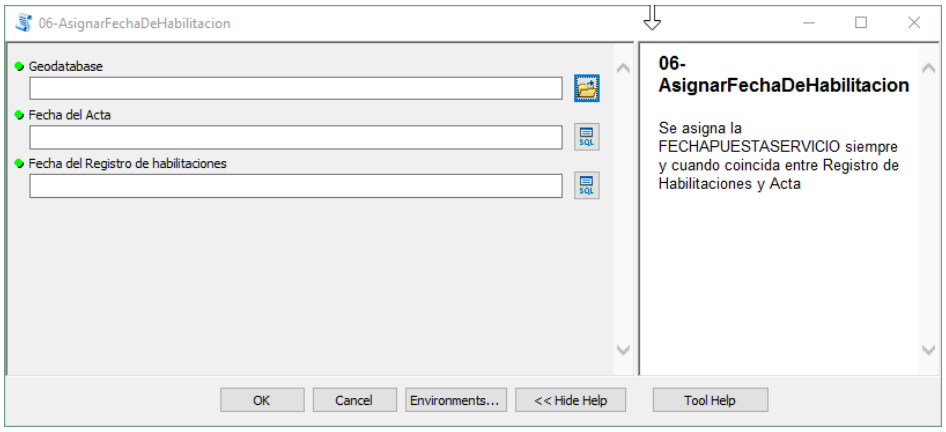
*Código fuente: Heredar fecha de habilitación*

```
import arcpy
#Se configura el entoirno de trabajo, y se asigna los parametros requeridos
arcpy.env.workspace = arcpy.GetParameterAsText(0)
gdb = arcpy.env.workspace
fechaActa = arcpy.GetParameterAsText(1)
fechaRegistro = arcpy.GetParameterAsText(2)
#Se excluye las capas que no requiere ese dato
capas = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")
capas.remove("GRM")
capas.remove("NODOVIRTUAL")
capas.remove("ODORIZADOR")
capas.remove("POLIGONOETAPA")
#Se valida que no haya caracteres no validos
fA = fechaActa.replace(" ", '')
fR = fechaRegistro.replace(" ", '')
#Se hereda el dato a todas las capas
if len(fA) == 12 and len(fR) == 12:
    for capa in capas:
        if fA == fR:
            arcpy.AddMessage("Se esta asigando las fechas a: " + capa)
            arcpy.CalculateField_management(in_table=capa, field="FECHAPUESTASERVICIO", expression=fA,
            expression_type="VB", code_block="")
            arcpy.CalculateField_management(in_table=capa, field="FECHAPUESTASERVICIO", expression=fR,
            expression_type="VB", code_block="")
        #Se alerta con un texto cuando las fechas no coinciden
        elif fA != fR:
            arcpy.AddMessage("Las fechas no coinciden")
#Se alerta con un texto cuando las fechas no coinciden
elif len(fA) != 12:
    arcpy.AddMessage("La fecha del Acta esta mal digitada")
elif len(fR) != 12:
    arcpy.AddMessage("La fecha del Registro esta mal digitada")
```

La interfaz de la herramienta tiene la siguiente apariencia con los tres parámetros requeridos.

**Figura 51**

*Interfaz: Heredar fecha de habilitación*




g. Heredar códigos de ubicación del proyecto: Los proyectos tienen ubicaciones propias creados por el cliente, que se debe agregar a todas las capas. Además, se agrega el código de distritos, en el proceso se valida que los datos ingresados no tengan errores de digitación y se excluye las capas que no soliciten estos datos.

**Figura 52**

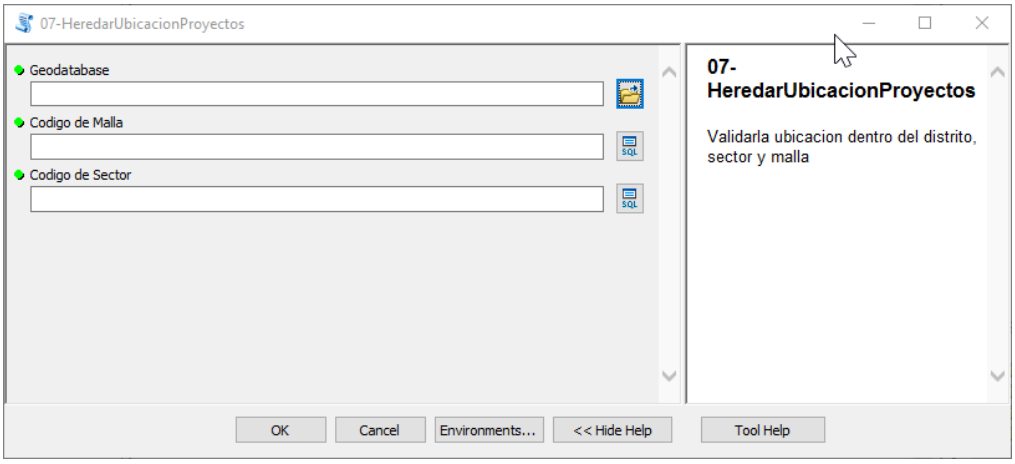
*Código fuente: Heredar códigos de ubicación*

```
import arcpy
#Se configura el entorno de trabajo y se asigna los parametros requeridos
arcpy.env.workspace = arcpy.GetParameterAsText(0)
gdb = arcpy.env.workspace
codMalla = arcpy.GetParameterAsText(1)
codSector = arcpy.GetParameterAsText(2)
#Valida que los codigos requeridos no tengan carcateres no validos
cM = codMalla.replace(" ", "")
cS = codSector.replace(" ", "")
codDistrito = cS[1:7]
#Se excluye las capas que no requieren estos datos
capas = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")
capas.remove("OBRAESPECIAL")
capas.remove("POLIGONOETAPA")
#hereda y valida los codigos a ingresar, de existir un error no ejecuta el comando y alerta
if len(cM) == 29 and len(cS)== 13:
    for capa in capas:
        arcpy.AddMessage("Se esta asignando los codigos de ubicacion a: " + capa + "...")
        arcpy.CalculateField_management(in_table=capa, field="CODIGOMALLA", expression=cM, expression_type="VB",
        code_block="")
        arcpy.CalculateField_management(in_table=capa, field="CODIGOSECTOR", expression=cS, expression_type="VB",
        code_block="")
        arcpy.CalculateField_management(in_table=capa, field="CODIGODISTRITO", expression=codDistrito,
        expression_type="VB", code_block="")
#De no ser valido los codigos alerta en la ventana de progreso
elif len(cM) != 29:
    arcpy.AddMessage("EL CODIGO DE MALLA ESTA ERRADO, POR FAVOR DIGITAR BIEN EL CODIGO")
elif len(cM) != 13:
    arcpy.AddMessage("EL CODIGO DE SECTOR ESTA ERRADO, POR FAVOR DIGITAR BIEN EL CODIGO")
```

La interfaz de las herramientas tiene la siguiente apariencia con los tres parámetros requeridos.

**Figura 53**

*Interfaz: Heredar códigos de ubicación*




h. Crear Vistas: Esta herramienta esta diseñada en Model Builder la cual nos permite seleccionar las columnas importantes para revisar visualmente, esto con el objetivo de hacer un control de calidad intermedio. La lógica de esta herramienta se adjuntará en los anexos.

**Figura 54**

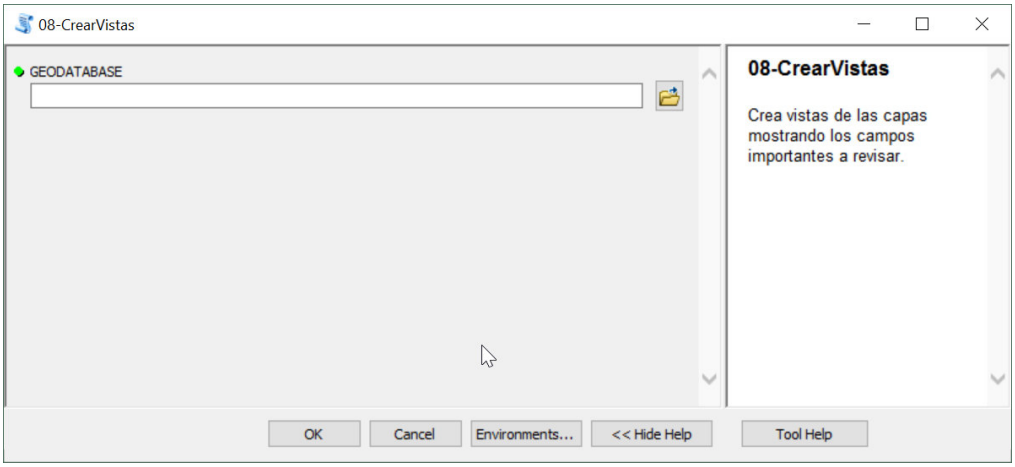
*Código fuente: Crear vistas*

```
import arcpy
#Configuracion del entorno de trabajo
arcpy.env.overwriteOutput = True
gdb = arcpy.GetParameterAsText(0)
#gdb = r"D:\Villareal\Scripts\ReplicaEsquema.gdb"
arcpy.env.workspace = gdb
capas = arcpy.ListFeatureClasses(feature_dataset='RedDeGas')
#Itera todas las capas
for capa in capas:
    if capa == "TRAMOGASODUCTO":
        arcpy.MakeFeatureLayer_management(capa, "TRAMOGASODUCTO_EVALUAR", "", "", "OBJECTID OBJECTID HIDDEN NONE;CODIGOELEMENTO CODIGOELEME
        CODIGOETAPA VISIBLE NONE;TIPOELEMENTO TIPOELEMENTO HIDDEN NONE;CODIGOPADRE CODIGOPADRE HIDDEN NONE;TIPOPADRE TIPOPADRE HIDDEN NONE;
        ESPESOR HIDDEN NONE;RESISTENCIA RESISTENCIA HIDDEN NONE;TARIFA TARIFA HIDDEN NONE;PROPIEDAD PROPIEDAD HIDDEN NONE;TIPOTERRENO TIPOT
        VISIBLE NONE;CODIGODISTRITO CODIGODISTRITO VISIBLE NONE;ESTADO ESTADO HIDDEN NONE;FECHAPRUEBA FECHAPRUEBA VISIBLE NONE;FECHAPUESTAS
        ACCESOREDUCIDO ACCESOREDUCIDO HIDDEN NONE;VALVULAEXCESOFLUJO VALVULAEXCESOFLUJO HIDDEN NONE;LONGITUDREAL LONGITUDREAL HIDDEN NONE;FU
        CODIGOUNDTECNICASAP CODIGOUNDTECNICASAP HIDDEN NONE;MODELOHIDRAULICO MODELOHIDRAULICO HIDDEN NONE;ENABLED ENABLED HIDDEN NONE;CLUSTI
        NONE;SISTEMADISTRIBUCION SISTEMADISTRIBUCION HIDDEN NONE;UTNIVEL3 UTNIVEL3 HIDDEN NONE;UTNIVEL4 UTNIVEL4 HIDDEN NONE;UTNIVEL5 UTNIVI
        NONE;UTNIVEL8 UTNIVEL8 HIDDEN NONE;CODIGOMALLA CODIGOMALLA VISIBLE NONE;CODIGOSECTOR CODIGOSECTOR VISIBLE NONE;USUARIO USUARIO HIDDI
        NONE;TIPOCONSTRUCCION TIPOCONSTRUCCION HIDDEN NONE;CODIGOVNR CODIGOVNR HIDDEN NONE;CODIGOVNRADAPTADO CODIGOVNRADAPTADO HIDDEN NONE;I
        ZANJACOMPARTIDA ZANJACOMPARTIDA HIDDEN NONE;DENOMINACION DENOMINACION HIDDEN NONE;ANHO ANHO HIDDEN NONE;CODIGOEQUIPO CODIGOEQUIPO H
        LONGITUDSAP LONGITUDSAP HIDDEN NONE;USUARIOCREACION USUARIOCREACION HIDDEN NONE;FECHAMODIFICACION FECHAMODIFICACION HIDDEN NONE;USUA
        ANHO2 HIDDEN NONE;ZONA_PQ ZONA_PQ HIDDEN NONE;AÑO_PQ AÑO_PQ HIDDEN NONE;COMENTARIO COMENTARIO HIDDEN NONE;CODPROYECTO_PQ CODPROYECT
        HIDDEN NONE;NOMBRE_PA NOMBRE_PA HIDDEN NONE;ID_REQUERIMIENTO ID_REQUERIMIENTO HIDDEN NONE;CODPROYECTO_PA CODPROYECTO_PA HIDDEN NONE
        created_date created_date HIDDEN NONE;last_edited_user last_edited_user HIDDEN NONE;last_edited_date last_edited_date HIDDEN NONE;SH
        AÑO_PQ14_18 AÑO_PQ14_18 HIDDEN NONE;CODPROYECTO_PQ14_18 CODPROYECTO_PQ14_18 HIDDEN NONE;SHAPE_Length SHAPE_Length HIDDEN NONE")

    if capa == "VALVULA":···
    if capa == "DERIVACION":···
    if capa == "REDUCCION":···
    if capa == "TAPON_EVALUAR":···
    if capa == "TUBERIACONEXION":···
    if capa == "CODO_":···
    if capa == "NODOVIRTUAL":···
    if capa == "OBRAESPECIAL":···
```

La interfaz de las herramientas tiene la siguiente apariencia con los tres parámetros requeridos.

**Figura 55**

*Interfaz: Crear Vistas*




i. Calcular Campos: Existen muchas columnas que se deben calcular en base a la información heredada con las anteriores herramientas. Debido a que el código tiene 251 líneas de código, por tal motivo el código completo se agrega en los anexos. El calculo se hace campo por campo por tal cual se usa un cursor:

**Figura 56**

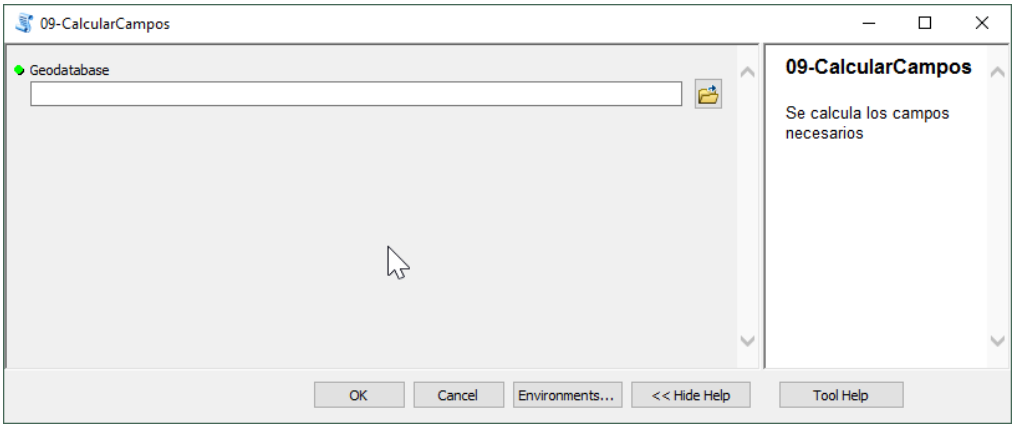
*Código Fuente: Calcular Campos*

```
import (module) arcpy
#Confi                pacio de trabajo
gdb = arcpy.GetParameterAsText(0)
fieldsTG = ['CODIGOMALLA', 'UTNIVEL4', 'UTNIVEL5', 'UTNIVEL6', 'UTNIVEL7', 'DENOMINACION' ]
fieldsDiam = ['DIAMETRO', 'ESPESOR', 'RESISTENCIA', 'MATERIAL', 'UTNIVEL3']
FieldDist = ['CODIGODISTRITO', 'CONCESION','SISTEMADISTRIBUCION']
fieldsAll = ['MATERIAL', 'TIPOELEMENTO', 'TARIFA', 'PROPIEDAD', 'ESTADO', 'FUENTEGRAFICA', 'ENABLED', 'TIPOCONSTRUCCION
fieldsAllD = ['MATERIAL', 'TIPOELEMENTO', 'TARIFA', 'PROPIEDAD', 'ESTADO', 'FUENTEGRAFICA', 'ENABLED', 'ESTADOVNR']
fieldsVNR = ['TIPOELEMENTO', 'MATERIAL', 'DIAMETRO', 'ENABLED','TIPOTERRENO', 'TIPOPAVIMENTO']
fieldsAllV = ['MATERIAL', 'TIPOELEMENTO', 'TIPOVALVULA', 'TARIFA', 'PROPIEDAD', 'ESTADO', 'FUENTEGRAFICA', 'ENABLED', '
fieldsAllCO = ['MATERIAL', 'TIPOELEMENTO', 'TARIFA', 'PROPIEDAD', 'ESTADO', 'FUENTEGRAFICA', 'ENABLED']
fieldsAllR = ['MATERIAL', 'TIPOELEMENTO', 'SUBTIPOELEMENTO', 'TARIFA', 'PROPIEDAD', 'ESTADO', 'FUENTEGRAFICA', 'ENABLED
fieldsAllT = ['MATERIAL', 'TIPOELEMENTO', 'SUBTIPOELEMENTO', 'TARIFA', 'PROPIEDAD', 'ESTADO', 'FUENTEGRAFICA', 'ENABLED
fieldsAllNV = ['ENABLED', 'TIPOELEMENTO']
arcpy.env.workspace = gdb
#Se enlista todas las capas
layers = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")
#Se itera cada capa calculando lso campos correspondientes
for layer in layers:
    arcpy.AddMessage("Se esta calculando los campos para: " + layer)
    if layer == "TRAMOGASODUCTO":
        with arcpy.da.UpdateCursor(layer, fieldsTG) as cursor:
            for row in cursor:
                malla = str(row[0])
                if row[0]== malla:
                    row[1]= malla[9:12]
                    row[2]= malla[13:20]
                    row[3]= malla[21:23]
                    row[4]= malla[24:27]
                    cursor.updateRow(row)
```

La interfaz de las herramientas tiene la siguiente apariencia con los tres parámetros requeridos.

**Figura 57**

*Interfaz: Calcular campos*




j. Heredar datos de usuario y contratista: Se debe heredar el dato de contratista y la identificación del usuario que carga. Para hacer eso se itera todas las capas, el código considera la exclusión de las capas que no requieren este dato.

**Figura 58**

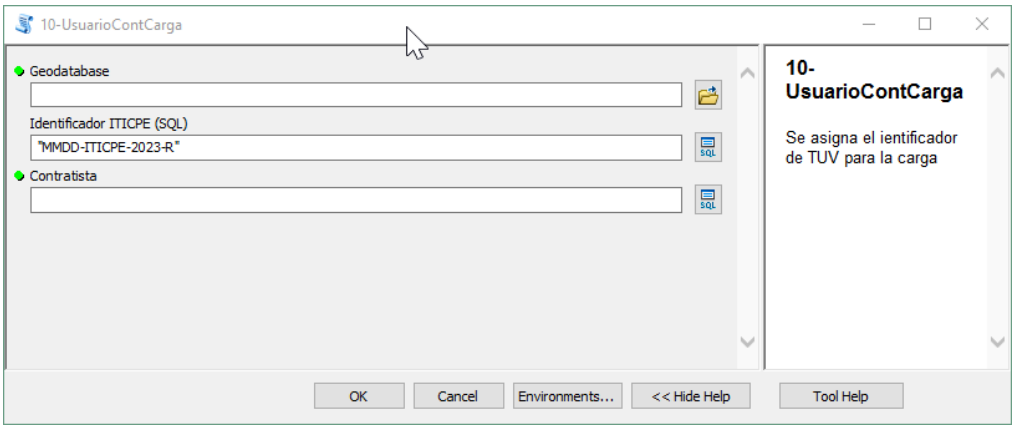
*Código fuente: Heredar usuario y contratista*

```
import arcpy
#Configuracion del espacio de trabajo
gdb = arcpy.GetParameterAsText(0)
arcpy.env.workspace = gdb
idCarga = arcpy.GetParameterAsText(1)
idContratista = arcpy.GetParameterAsText(2)
capas = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")
#Se excluye las capas que no solicitan este datos
capas.remove("POLIGONOETAPA")
capas.remove("UNIONES")
#Se itera cada capa para asignar los datos
for capa in capas:
    arcpy.AddMessage("Se esta calculando la contratista para: " +capa+"...")
    if capa == 'NODOVIRTUAL':
        arcpy.CalculateField_management(capa, "USUARIOCONTCARGA", idCarga, "VB", "")
    else:
        arcpy.CalculateField_management(capa, "USUARIOCONTCARGA", idCarga, "VB", "")
        arcpy.CalculateField_management(capa, "CONSTRUCTORA", idContratista, "VB", "")
```

La interfaz de la herramienta tiene la siguiente experiencia con los parámetros requeridos:

**Figura 59**

*Heredar usuario y contratista*




k. Heredar codigos PQ/PA: El análisis del plan quinquenal y plan anual se realiza manualmente sobre la red, indicando los resultados en seis campos, pero los resultados del análisis deben ser distribuidos a todos los accesorios de la red, teniendo el cuidado respectivo de que algunas capas reciben solo 4 columnas de información y otras capas reciben seis. Se logro eso con el siguiente código:

**Figura 60**

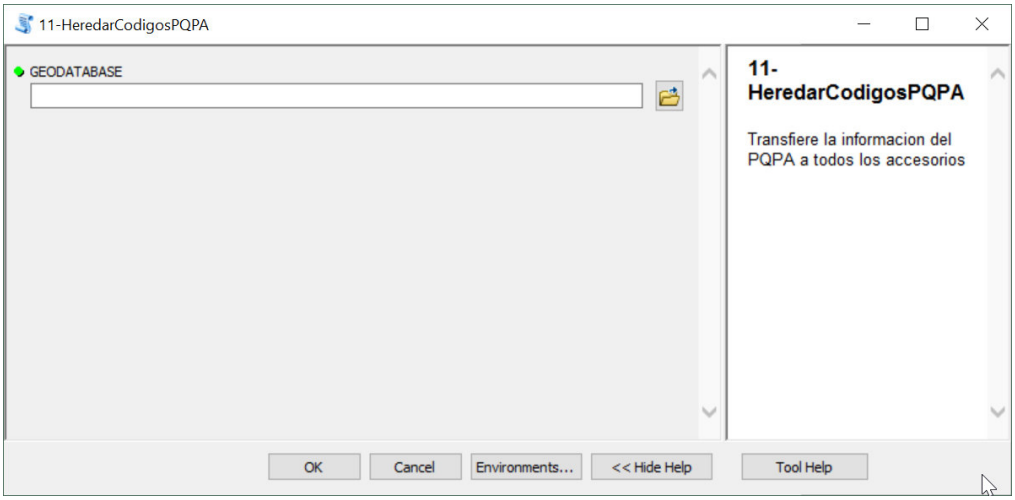
*Código fuente: Heredar códigos PQ y PA*

```
import arcpy
#Configuracion del entorno de trabajo
arcpy.env.overwriteOutput = True
gdb = arcpy.GetParameterAsText(0)
arcpy.env.workspace = gdb
capas = arcpy.ListFeatureClasses(feature_dataset='RedDeGas')
capasAnio = ['VALVULA', 'TUBERICONEXION']
#Creación
space = gdb[:-19]
tempGDB = space +"\\temp.gdb"
if arcpy.Exists(tempGDB):
    arcpy.Delete_management(tempGDB)
arcpy.CreateFileGDB_management(space, "temp")
tg = gdb+"\\RedDeGas\\TRAMOGASODUCTO"
#Se define los campos que se van a copiar en una lista
campos_transferir = ["CODPROYECTO_PA", "CODPROYECTO_PQ", "ANIO_PA", "ANIO_PQ", "NOMBRE_PA", "NOMBRE_PQ"]
campos_transferir_2 = ["CODPROYECTO_PA", "CODPROYECTO_PQ","NOMBRE_PA", "NOMBRE_PQ"]
#Se crea la funcion que va a copiar los campos de una capa a otra
def calcular_campos (layer, campos):
    join_lyr = arcpy.SpatialJoin_analysis(layer, tg, tempGDB+"\\"+layer+"lyr")
    arcpy.MakeFeatureLayer_management(layer, layer+"_lyr")
    capa_join =arcpy.AddJoin_management(layer+"_lyr","OBJECTID", join_lyr, "TARGET_FID")
    for campo in campos:
        arcpy.CalculateField_management(capa_join, "{}.{}".format(layer, campo), "[{}lyr.{}_1]".format(layer, campo), "VB", "")
    arcpy.RemoveJoin_management(layer+"_lyr", layer+"lyr")
#Se recorre todas las capas de la geodatabase, copiando las capas
for capa in capas:
    numElementos = arcpy.GetCount_management(capa)
    arcpy.AddMessage(capa)
    if capa in capasAnio and numElementos > 0:
        calcular_campos(capa, campos_transferir)
    elif capa not in capasAnio and numElementos > 0:
        calcular_campos(capa, campos_transferir_2)
```

La interfaz de la herramienta tiene la siguiente experiencia con los parámetros requeridos:

**Figura 61**

*Código fuente: Heredar códigos PQ y PA*




l. Crear campos de control de calidad: A partir de acá las herramientas que se crearan es para hacer control de calidad. Para hacer el control de calidad se crearán columnas que ayudarán a identificar los posibles errores. Se debe crear para todas las capas, por ello, es que se creó esta herramienta.

**Figura 62**

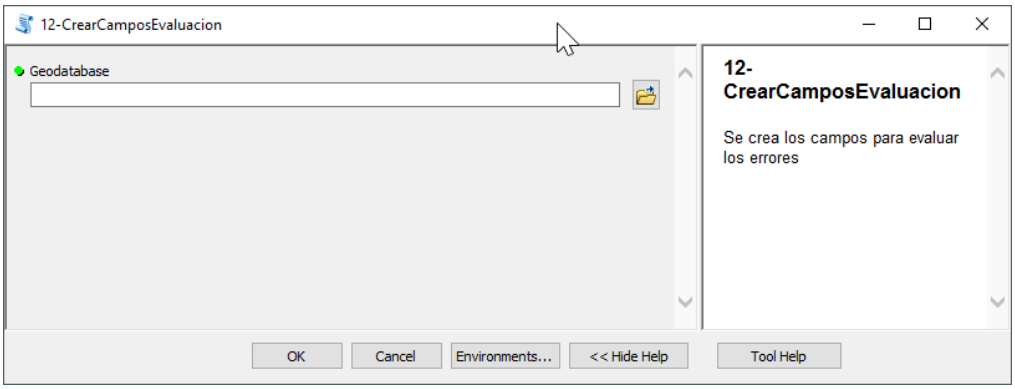
*Código fuente: Crear campos de control de calidad*

```
import arcpy
#Configuracion del espacio de trabajo
arcpy.env.workspace = arcpy.GetParameterAsText(0)
lista = arcpy.ListFeatureClasses(feature_dataset='RedDeGas')
#Se crean los campos para todas las capas
for capa in lista:
    arcpy.AddMessage("Se esta creando los campos para: "+capa+"...")
    arcpy.AddField_management(capa, "DatosCompletos", "TEXT", "", "", "50", "", "NULLABLE", "NON_REQUIRED", "")
    arcpy.AddField_management(capa, "Dominios", "TEXT", "", "", "50", "", "NULLABLE", "NON_REQUIRED", "")
```

La interfaz de la herramienta tiene la siguiente experiencia con los parámetros requeridos:

**Figura 63**

*Interfaz: Crear campos de control de calidad*




m. Errores Básicos: Valida que todos los campos tengan valores y que esos valores sean correctos, es una validación simple para los datos importantes, el código es bastante amplio, agregara a los anexos.

**Figura 64**

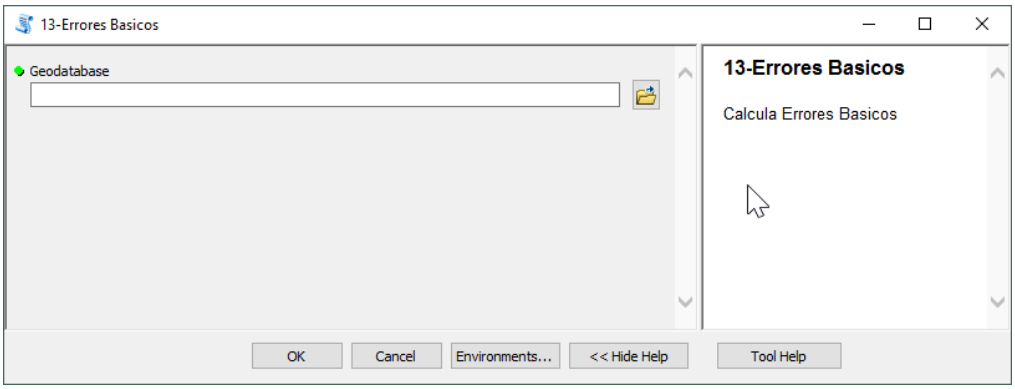
*Código fuente: Errores básicos*

```
import arcpy
#Configuracion del entorno de trabajo
arcpy.env.workspace = arcpy.GetParameterAsText(0)
capas = arcpy.ListFeatureClasses(feature_dataset='RedDeGas')
#iteria por cada capa
for capa in capas:
    lyr = arcpy.Describe(capa)
    fields = lyr.fields
    for field in fields:
        if field.name == "Errores01":
            arcpy.DeleteField_management(capa, "Errores01;ErroresTG")
            arcpy.AddMessage(capa)
    if capa == 'OBRAESPECIAL' or capa == 'POLIGONOETAPA':
        arcpy.CalculateField_management(in_table=capa, field="CODIGOPROYECTO", expression='!CODIGOPROYECTO!.replace("
        ","")', expression_type="PYTHON_9.3", code_block="")
        arcpy.CalculateField_management(in_table=capa, field="CODIGOETAPA", expression='!CODIGOETAPA!.replace(" ","")
        ', expression_type="PYTHON_9.3", code_block="")
        arcpy.AddField_management(capa, "Errores01", "TEXT", field_length= 250)
        arcpy.CalculateField_management(in_table=capa, field="Errores01", expression="cp( !CODIGOPROYECTO!,
        !CODIGOETAPA!)", expression_type="PYTHON_9.3", code_block="def cp(a, b):\n    valor = ''\n    if len(a) !=
        11:\n        valor = valor + 'Error: En la cantidad de caracteres CODIGOPROYECTO'\n    if len(b) !=
        14:\n        valor = valor + ' Error: En la cantidad de caracteres CODIGOETAPA'\n    if valor == '':\n
        valor = 'Todo correcto'\n    return valor")
```

La interfaz de la herramienta tiene la siguiente experiencia con los parámetros requeridos:

**Figura 65**

*Interfaz: Errores básicos*




n. Errores Completos: Con la misma lógica de los errores básicos, se evalúa los todos los campos de cada capa, lo que se valida es que todos los campos tengan los datos necesarios y en determinados casos que tengan un valor valido. Debido a que el código es bastante amplio se mostrara el código completo en anexos.

**Figura 66**

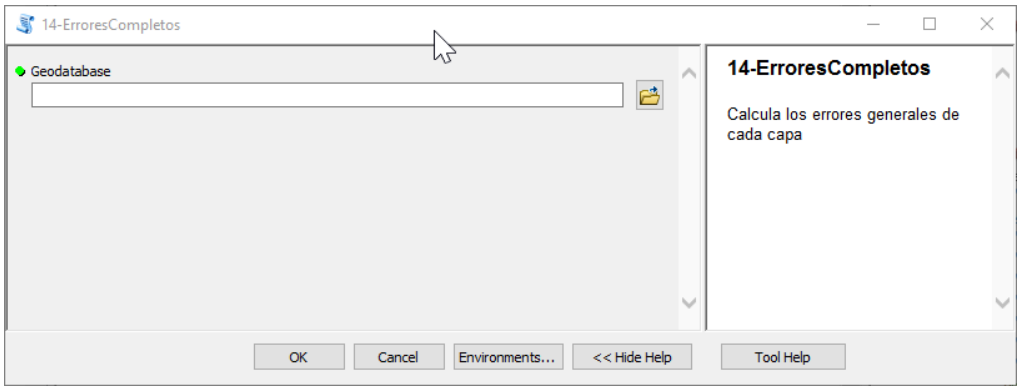
*Código fuente: Errores completos*

```
import arcpy

#Geodatbase donde se trabajara
gdb = arcpy.GetParameterAsText(0)
arcpy.env.workspace = gdb

capas = arcpy.ListFeatureClasses(feature_dataset="RedDeGas")

for capa in capas:
    arcpy.AddMessage(capa)
    if capa == 'VALVULA':
        arcpy.CalculateField_management(in_table=capa, field="CODIGOVNR", expression="codigovnr( !TIPOELEMENTO! , !MATERIAL! ,
        !DIAMETRO! , '01', !MAPO! ,  !TIPOVALVULA! )", expression_type="PYTHON_9.3", code_block="def codigovnr (campo1, campo2, campo3,
        campo4, campo5, campo6):\n    codigo = ''\n    if campo1 == 'VAL':\n        codigo = codigo + '04'\n    if campo2 ==
        'PE':\n        codigo = codigo + '02'\n    if campo3 == 20.1:\n        codigo = codigo + '01'\n    elif campo3 == 25:\n
        codigo = codigo + '02'\n    elif campo3 == 32:\n        codigo = codigo + '03'\n    elif campo3 == 40:\n        codigo = codigo +
        '04'\n    elif campo3 == 50:\n        codigo = codigo + '05'\n    elif campo3 == 63:\n        codigo = codigo + '06'\n    elif
        campo3 == 90:\n        codigo = codigo + '07'\n    elif campo3 == 110:\n        codigo = codigo + '08'\n    elif campo3 ==
        160:\n        codigo = codigo + '09'\n    elif campo3 == 200:\n        codigo = codigo + '10'\n    if campo4 == '01':\n
        codigo = codigo + '01'\n    if campo5 == 5:\n        codigo = codigo + '04'\n    elif campo5 == 10:\n        codigo = codigo +
        '05'\n    elif campo5 == 19:\n        codigo = codigo + '06'\n    elif campo5 == 50:\n        codigo = codigo + '08'\n    elif
        campo5 == 100:\n        codigo = codigo + '10'\n    elif campo5 == 150:\n        codigo = codigo + '04'\n    elif campo5 ==
        153:\n        codigo = codigo + '11'\n    if campo6 == 'Ball':\n        codigo = codigo + '01'\n    return codigo\n")
        arcpy.CalculateField_management(capa, "DatosCompletos", "campos( !CODIGOPROYECTO! , !CODIGOETAPA! , !TIPOELEMENTO! ,
        !TIPOVALVULA! , !SUBTIPOELEMENTO!  , !MATERIAL!, !TARIFA!, !CLASE!, !MAPO!, !PROPIEDAD!, !TIPOTERRENO!, !ESTADO!, !CONSTRUCTORA!,
        !FECHAPRUEBA!, !FECHAPUESTASERVICIO!, !FUENTEGRAFICA!, !CODIGODISTRITO!, !ENABLED!, !DIRECCION!, !CODIGOMALLA!, !CODIGOSECTOR!,
        !CONCESION!, !SISTEMADISTRIBUCION!, !UTNIVEL3!, !UTNIVEL4!, !UTNIVEL5!, !UTNIVEL6!, !UTNIVEL7!, !DENOMINACION!, !ESTADOVNR!,
        !USOVALVULA!, !CODIGOVNR!, !PROFUNDIDAD!, !USUARIOCONTCARGA!, !DIAMETRO! )", "PYTHON_9.3", code_block="def campos(*args):\n
        valor = ''\n    for arg in args:\n        if arg == None or arg == '':\n            valor = 'Error: Hay campos obligatorios
        vacios'\n            return valor\n    if valor == '':\n        valor = 'Todo esta correcto'\n        return valor")
        arcpy.CalculateField_management(capa, "Dominios", "dominios( !TIPOELEMENTO!, !TIPOVALVULA!, !SUBTIPOELEMENTO!,!MATERIAL!,
        !TARIFA!, !CLASE!, !MAPO!, !PROPIEDAD!, !TIPOTERRENO!, !ESTADO!, !CONSTRUCTORA!, !ENABLED!, !CONCESION!, !SISTEMADISTRIBUCION!,
        !UTNIVEL3!, !ESTADOVNR!, !USOVALVULA!, !DIAMETRO! )", "PYTHON_9.3", code_block="def dominios(*args):\n    valor = ''\n    for arg
        in args:\n        argText = str(arg)\n        if len(argText) > 5 or argText == '':\n            valor = 'Error: Hay un error en
        los dominios'\n            return valor\n    if valor == '':\n        valor = 'Todo esta correcto'\n        return valor")
```

La interfaz de la herramienta tiene la siguiente experiencia con los parámetros requeridos:

**Figura 67**

*Interfaz: Errores completos*




o. Crear reporte de errores: Con el objetivo de generar un reporte con el resumen de análisis de los posibles errores detectados por cada capa, se creó un script donde indica si hay errores detectados en las capas. Para ello se usa la tabla creada en la herramienta 4.

**Figura 68**

*Código fuente: Crear reporte de errores*

```
import arcpy

#Configuracion del entorno de trabajo
fc_in = arcpy.GetParameterAsText(0)
fields = ['CAPA', 'NUMEROELEMENTOS', 'ELEMENTOSINCOMPLETOS', 'DOMINIOSINCORRECTOS', 'CARGAR']
gdb = arcpy.GetParameterAsText(1)
gdb_tabla = gdb + '\\TablaReportes'
lyrfile_text = gdb + '\\RedDeGas\\'

#Eliminar la tabla SI existe
if arcpy.Exists(gdb_tabla) == True:
    arcpy.Delete_management(gdb_tabla)

#Se copia la tabla la geodata
arcpy.Copy_management(fc_in, gdb_tabla)
capas = ["VALVULA", "TAPON", "CODO", "REDUCCION", "DERIVACION", "TRAMOGASODUCTO", "NODOVIRTUAL", "ERM", "ERP", "GRM", "NODOERP"
"VALVULAEXCESOFLUJO", "TUBERIACONEXION", "LINEAINTERNA"]
```

(variable) gdb_tabla: Any

Luego de establecer el entorno de trabajo se recorre todas las capas con el objetivo de detectar posibles errores, al detectar un error lo registra en la tabla de reporte. De esta manera se tiene un resumen con los errores detectas y es más fácil poder corregir.

**Figura 69**

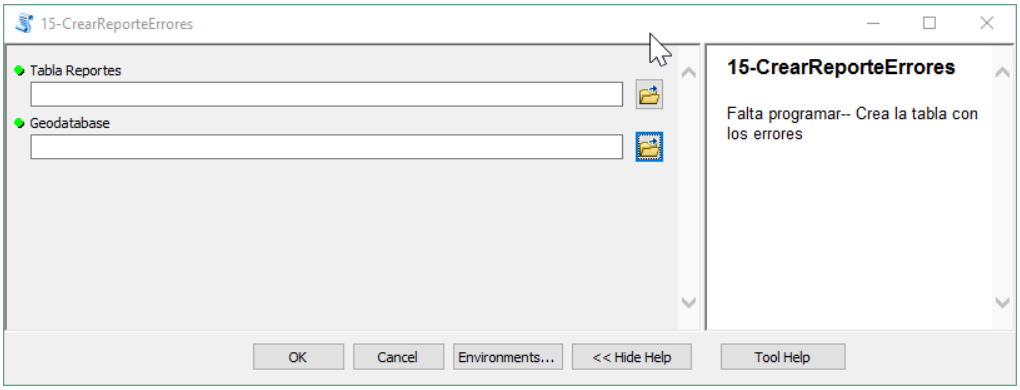
*Código Fuente: Crear reporte de errores*

```
for capa in capas:
    #Se recorre toda la GDB
    layer = lyrfile_text + str(capa)
    #Se cuenta el número de elementos que hay en la capa
    erroresElementos = arcpy.GetCount_management(layer)
    resultElementos = int(erroresElementos.getOutput(0))
    #Se caclcula el número de campos vacios que hay en la capa
    capa_layer = capa + 'layer'
    errorVacios = arcpy.MakeFeatureLayer_management(in_features=layer, out_layer= capa_layer, where_clause="DatosCompletos = 'Error: Hay campos obligatorios vacios'")
    cuentaVacios = arcpy.GetCount_management(errorVacios)
    resultVacios = int(cuentaVacios.getOutput(0))
    #Se caclula el número de campos vacios que en la capa
    capa2_layer = capa + 'layer' + str(1)
    errorDominios = arcpy.MakeFeatureLayer_management(in_features=layer, out_layer= capa2_layer, where_clause="Dominios = 'Error: Hay un error en los dominios'")
    cuentaDominios = arcpy.GetCount_management(errorDominios)
    resultDominios= int(cuentaDominios.getOutput(0))
    #Actualiza la tabla por cada fila
    with arcpy.da.UpdateCursor(gdb_tabla, fields) as cursor:
        for row in cursor:
            if row[0] == capa:
                row[1] = resultElementos
                row[2] = resultVacios
                row[3] = resultDominios
                if row[2] + row[3] > 1:
                    row[4] = "NO"
                else:
                    row[4] = "Si"
                if row[1] == 0:
                    row[4] = "NO"
            cursor.updateRow(row)
```

La interfaz de la herramienta tiene la siguiente experiencia con los parámetros requeridos:

**Figura 70**

*Interfaz: Crear reporte de errores*




p. Eliminar campos de errores: Después de analizar y corregir todos los posibles de errores generados al largo del proceso, se debe eliminar todos los campos creados para este fin, para ellos ejecutamos el siguiente código:

**Figura 71**

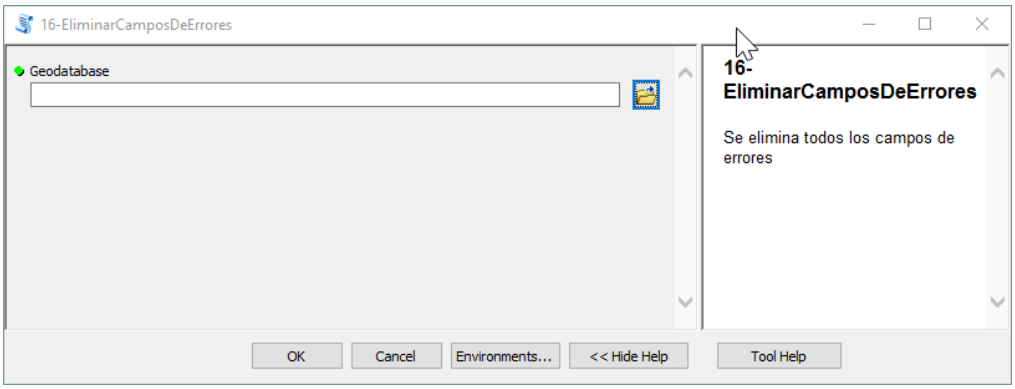
*Código fuente: Eliminar campos de errores*

```
import arcpy
#Configuracion del entorno de trabajo
arcpy.env.workspace = arcpy.GetParameterAsText(0)
capas = arcpy.ListFeatureClasses(feature_dataset='RedDeGas')

#Recorre toda la lista
for capa in capas:
    #Se recorre toda la GDB
    # Elimina los campos de revisión
    msg = 'Se esta eliminando en: ' + capa
    arcpy.AddMessage(msg)
    arcpy.DeleteField_management(capa, "DatosCompletos;Dominios;Errores01;ErroresTG;PavimentoPQ;PavimentoPA")
```

La interfaz de la herramienta tiene la siguiente experiencia con los parámetros requeridos:

**Figura 72**

*Interfaz: Eliminar campos de errores*




q. Heredar códigos padres a la red: Luego de cargar las redes, específicamente las tuberías, a la base de datos del cliente, este genera un código automático correlativo denominado código elemento, pero este código elemento es el código padre del elemento que le sigue en la lógica de la red. Por ejemplo, el primer elemento creado se asigna el código de elemento 1, pero el segundo elemento creado se le asigna el código elemento 2, pero su código padre viene a ser el 1 debido a que es el elemento que le antecede. Para asignar este código se utiliza el siguiente modelo:

**Figura 73**

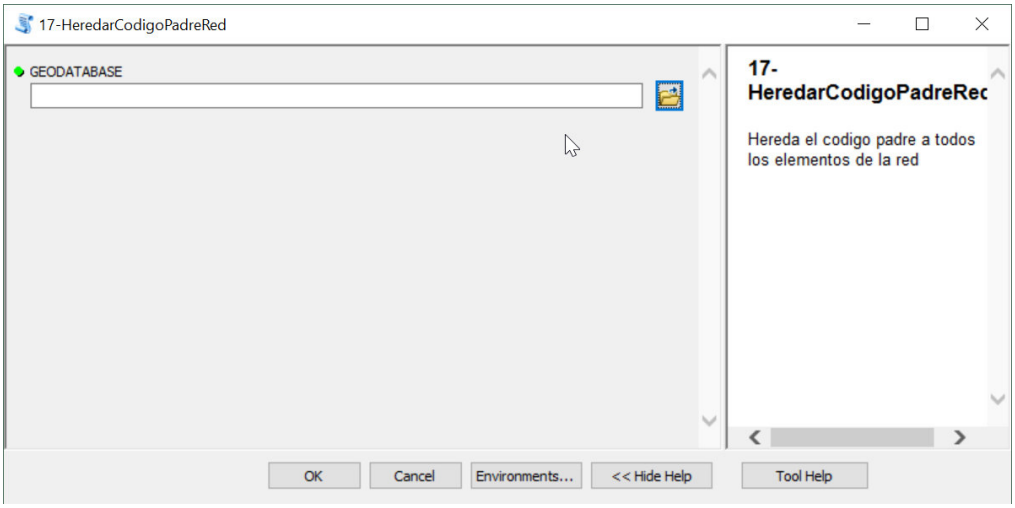
*Código fuente: Heredar códigos padres a la red*

```
import arcpy
#Configuracion del entorno de trabajo
arcpy.env.overwriteOutput = True
gdb = arcpy.GetParameterAsText(0)
#gdb = r"D:\Villareal\Scripts\ReplicaEsquema.gdb"
arcpy.env.workspace = gdb
capas = arcpy.ListFeatureClasses(feature_dataset='RedDeGas')
#Creacion
space = gdb[:-19]
tempGDB = space +"\\temp.gdb"
if arcpy.Exists(tempGDB):
    arcpy.Delete_management(tempGDB)
arcpy.CreateFileGDB_management(space, "temp")
tg = gdb+"\\RedDeGas\\TRAMOGASODUCTO"

inicio = arcpy.FeatureVerticesToPoints_management(tg, tempGDB+"\\tg_point_inicio","START")
fin = arcpy.FeatureVerticesToPoints_management(tg, tempGDB+"\\tg_point_fin","END")

sj = arcpy.SpatialJoin_analysis(inicio, fin, tempGDB+"\\sj", join_operation ="JOIN_ONE_TO_ONE", match_option="INTERSECT")
tg_lyr = arcpy.MakeFeatureLayer_management(tg, "tg_lyr")
join_tg_original = arcpy.AddJoin_management(tg_lyr, "OBJECTID",sj, "ORIG_FID")
arcpy.CalculateField_management(join_tg_original, "TRAMOGASODUCTO.CODIGOPADRE", "[sj.CODIGOELEMENTO_1]")
```

La interfaz de la herramienta tiene la siguiente experiencia con los parámetros requeridos:

**Figura 74**

*Interfaz: heredar código padre a la red*




r. Heredar código padre a los accesorios: Después de asignar el código padre a las tuberías, también se debe heredar el código padre a los accesorios de la red, se sigue la misma lógica, pero para los accesorios el código padre viene a ser el elemento anterior en la lógica de la dirección de la red, para ellos se ejecuta el siguiente modelo:

**Figura 75**

*Código fuente: Heredar código padre a los accesorios*

```
import arcpy
#Configuracion del entorno de trabajo
arcpy.env.overwriteOutput = True
gdb = arcpy.GetParameterAsText(0)
#gdb = r"D:\Villareal\Scripts\ReplicaEsquema.gdb"
arcpy.env.workspace = gdb
capas = arcpy.ListFeatureClasses(feature_dataset='RedDeGas')
#Creacion
space = gdb[:-19]
tempGDB = space +"\\temp.gdb"
if arcpy.Exists(tempGDB):
    arcpy.Delete_management(tempGDB)
arcpy.CreateFileGDB_management(space, "temp")
tg = gdb+"\\RedDeGas\\TRAMOGASODUCTO"
end = arcpy.FeatureVerticesToPoints_management(tg, tempGDB+"\\tg_nodo_final", "END")
def heredarPadreAccesorio(layer):
    layer_sj = arcpy.SpatialJoin_analysis(layer, end, tempGDB+"\\sj_nodo_fin", join_operation= "JOIN_ONE_TO_ONE", match_option="INTERSECT")
    arcpy.MakeFeatureLayer_management(layer, layer+"_lyr")
    join_lyr = arcpy.AddJoin_management(layer+"_lyr", "OBJECTID", layer_sj, "TARGET_FID")
    arcpy.CalculateField_management(join_lyr, "{}.CODIGOPADRE".format(layer), "[sj_nodo_fin.CODIGOELEMENTO_1]")
    arcpy.CalculateField_management(join_lyr, "{}.TIPOPADRE".format(layer), "TG")
    arcpy.RemoveJoin_management(layer+"_lyr", "sj_nodo_fin" )

for capa in capas:
    numElementos = arcpy.GetCount_management(capa)
    arcpy.AddMessage(capa)
    if numElementos > 0 and (capa != "CODO" and capa != "POLIGONOETAPA" and capa != "TRAMPASCRAPPER"):
        arcpy.AddMessage(capa)
        heredarPadreAccesorio(capa)
```

La interfaz de la herramienta tiene la siguiente experiencia con los parámetros requeridos:

**Figura 76**

*Interfaz: Heredar código padre a los accesorios*