ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

APLICACIONES DE SOFTWARE EDUCATIVOS Y EMPRESARIAL PROTOTIPO DE SOFTWARE QUE FACILITE LA AUTOMATIZACIÓN DE TAREAS DE RECOPILACIÓN, PROCESAMIENTO Y ANÁLISIS DE INFORMACIÓN DE EQUIPOS DE RED UTILIZANDO PYTHON SCRIPTING EN LA EMPRESA LIKATELEC

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

CHRISTIAN OCTAVIO OYANA GUACHAMÍN
chr_oyana@hotmail.com

DIRECTOR: DRA. ANA MARÍA ZAMBRANO VIZUETE
ana.zambrano@epn.edu.ec

DMQ, Abril 2023

CERTIFICACIONES

Yo, Christian Octavio Oyana Guachamín declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Christian Octavio Oyana Guachamín

Certifico que el presente trabajo de integración curricular fue desarrollado por Christian Octavio Oyana Guachamín, bajo mi supervisión.

Dra. Ana María Zambrano Vizuete
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

SR. CHRISTIAN OCTAVIO OYANA GUACHAMÍN

DRA. ANA MARÍA ZAMBRANO VIZUETE

DEDICATORIA

A mi madre, pues gracias a su apoyo y amor incondicional no permitió que me rinda en este largo camino universitario. A mi padre, pues gracias a sus enseñanzas y consejos me han llevado a ser un mejor ser humano. A mis hermanas, sobrinas y sobrinos a quienes llevo en mi corazón, nunca dejen de luchar por sus sueños.

A mis amigos que han sido parte integral de mi desenvolvimiento personal y profesional.

A todos quienes de alguna manera aportaron en el desarrollo de este trabajo, sepan que les estaré eternamente agradecido.

Christian Octavio Oyana Guachamín

AGRADECIMIENTO

Brindo un agradecimiento especial a la Dra. Ana María Zambrano, directora de este trabajo, por darme las directrices correctas, brindarme su apoyo y permitir que este trabajo se terminara a buen recaudo.

Agradezco a Dios y a mi familia, quienes siempre han creído en mí, dándome ejemplo de superación, humildad y sacrificio; enseñándome a valorar todo lo que tengo y lo que soy.

Christian Octavio Oyana Guachamín

ÍNDICE DE CONTENIDOS

CERTIFICACIONES I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDOS V
RESUMEN VII
ABSTRACT VIII
1 INTRODUCCIÓN 9
1.1 OBJETIVO GENERAL 9
1.2 OBJETIVOS ESPECÍFICOS 9
1.3 ALCANCE 10
1.4 MARCO TEÓRICO 14
1.4.1 *PYTHON SCRIPTING* Y SU APLICACIÓN EN LA PROGRAMABILIDAD DE REDES 14
1.4.2 GESTIÓN DE EQUIPOS DE RED CISCO A TRAVÉS DE CLI 15
1.4.3 METODOLOGÍA DE GESTIÓN DEL CICLO DE VIDA DE APLICACIONES (ALM) 16
1.4.4 TECNOLOGÍAS, ENTORNOS DE DESARROLLO Y LENGUAJES DE PROGRAMACIÓN PARA IMPLEMENTACIÓN DE PROTOTIPO 17
2 METODOLOGÍA 21
2.1 GOBERNANZA 21
2.1.1 PLANTEAMIENTO DEL TABLERO DE GESTIÓN DEL CICLO DE VIDA DE APLICACIONES (ALM) 21
2.1.2 REQUERIMIENTOS FUNCIONALES (RF) 22
2.1.3 REQUERIMIENTOS NO FUNCIONALES (RNF) 25
2.1.4 DISEÑO DE LA CAPA DE NEGOCIO 26
2.1.5 DISEÑO DE LA CAPA DE PRESENTACIÓN 29
2.2 DESARROLLO 32
2.2.1 ACTUALIZACIÓN DEL TABLERO DE GESTIÓN DEL CICLO DE VIDA DE APLICACIONES (ALM) 33
2.2.2 IMPLEMENTACIÓN DE LA CAPA DE NEGOCIO 33
2.2.3 IMPLEMENTACIÓN DE LA CAPA DE PRESENTACIÓN 35
2.3 OPERACIONES 38
2.3.1 ACTUALIZACIÓN DEL TABLERO DE GESTIÓN DEL CICLO DE VIDA DE APLICACIONES (ALM) 39

2.3.2 DEFINICIÓN DEL ENTORNO DE PRUEBAS 39
2.3.3 VALIDACIÓN DE REQUERIMIENTOS FUNCIONALES 40
2.3.4 VALIDACIÓN DE REQUERIMIENTOS NO FUNCIONALES 41
2.3.5 CIERRE DEL TABLERO DE GESTIÓN DEL CICLO DE VIDA DE APLICACIONES (ALM) 43
3 CONCLUSIONES Y RECOMENDACIONES 44
3.1 CONCLUSIONES 44
3.2 RECOMENDACIONES 45
4 REFERENCIAS BIBLIOGRÁFICA 46
5 ANEXOS 47

RESUMEN

El presente Trabajo de Integración Curricular plantea un prototipo de software que permita a la empresa LIKATELEC la automatización de tareas como: recopilación, procesamiento, análisis de información y presentación de documentación técnica; las cuales se ejecutan durante un mantenimiento preventivo de equipos de red CISCO en sus clientes de manera manual y repetitiva utilizando *Python Scripting*.

Durante el **Capítulo 1**, se explora los pormenores de *Python Scripting* y su aplicación en la programabilidad de redes conjuntamente con la gestión de equipos de red Cisco a través de *Command-Line Interface* (CLI). Siguiente, se detalla la metodología *Application Lifecycle Management* (ALM) que se utiliza a lo largo del presente trabajo. Por último, se explica las tecnologías, entornos de desarrollo y lenguajes de programación para la implementación de prototipo.

A lo largo del **Capítulo 2**, se muestra las particularidades de la construcción de la aplicación de escritorio orientada a la automatización de tareas de mantenimiento de equipos de red: las decisiones de diseño tomadas y los diferentes detalles de desarrollo realizadas. De igual manera, en este Capítulo se sigue las etapas definidas por la metodología ALM que son: Gobernanza, Desarrollo y Operaciones.

En la etapa de Gobernanza, se define las principales pautas para el desarrollo de la aplicación de escritorio, por otra parte, en la etapa de Desarrollo, se explora las decisiones de desarrollo tomadas y, finalmente; en la etapa de Operaciones, se muestra los resultados obtenidos después de finalizar la etapa de implementación del prototipo que ha sido probado sobre una simulación de red (por confidencialidad de datos), replicando el diseño de red total de un cliente determinado de la empresa LIKATELEC.

PALABRAS CLAVE: Automatización, *Python Scripting*, *Command-Line Interface*, CLI, ALM, *Application Lifecycle Management*

ABSTRACT

This Curricular Integration Paper proposes a software prototype that allows the company LIKATELEC to automate tasks such as: collection, processing, information analysis and presentation of technical documentation; which are executed during preventive maintenance of CISCO network equipment in their customers manually and repetitively using Python Scripting.

Chapter 1 explores the details of Python Scripting and its application in network programmability in conjunction with the management of Cisco network equipment through the Command-Line Interface (CLI). Next, the Application Lifecycle Management (ALM) methodology used throughout this work is detailed. Finally, the technologies, development environments and programming languages for the prototype implementation are explained.

Chapter 2 shows the particularities of the construction of the desktop application oriented to the automation of network equipment maintenance tasks: the design decisions taken and the different development details made. Similarly, this chapter follows the stages defined by the ALM methodology, which are: Governance, Development and Operations.

In the Governance stage, the main guidelines for the development of the desktop application are defined, on the other hand, in the Development stage, the development decisions taken are explored and, finally, in the Operations stage, the results obtained after the end of the prototype implementation stage are shown, which has been tested on a network simulation (by data confidentiality), replicating the total network design of a specific client of the LIKATELEC company.

KEYWORDS: Automation, Python Scripting, Command-Line Interface, CLI, ALM, Application Lifecycle Management

1 INTRODUCCIÓN

Al analizar las tareas que un técnico de la empresa LIKATELEC realiza durante el mantenimiento preventivo de equipos de red en una empresa Cliente, es fácil detectar la existencia de tareas como: recopilación, procesamiento, análisis de información y presentación de documentación técnica; las cuales se ejecutan de manera manual y repetitiva, ocasionando pérdida de tiempo en la ejecución de las mismas (tanto para el cliente como para LIKATELEC). Todo esto repercute en tiempo extenso de entrega de la documentación técnica pertinente, y disponibilidad limitada del personal de le empresa LIKATELEC para atención a otros requerimientos de la empresa.

El presente Trabajo de Integración Curricular propone el desarrollo de un prototipo software que permita a la empresa LIKATELEC automatizar las tareas manuales y repetitivas durante el mantenimiento preventivo de equipos de red en sus clientes utilizando *Python Scripting.*

1.1 OBJETIVO GENERAL

Diseñar e implementar un prototipo de software que facilite la automatización de tareas de mantenimiento de equipos de red utilizando *Python Scripting* en la empresa LIKATELEC.

1.2 OBJETIVOS ESPECÍFICOS

1. Analizar las tecnologías y herramientas necesarias para la realización del prototipo, tal como: *Python Scripting* y su aplicación en la programabilidad de redes, y la metodología ALM (*Application Lifecycle Management*).
2. Analizar el funcionamiento y tareas a realizar en mantenimientos preventivos de equipos clientes en LIKATELEC.
3. Diseñar los módulos y diagramas que comprenden el prototipo.
4. Implementar el prototipo de acuerdo al diseño realizado.
5. Evaluar el funcionamiento del prototipo sobre una simulación de red realizado por un técnico de le empresa LIKATELEC.

1.3 ALCANCE

El presente Proyecto de Integración Curricular propone el desarrollo de un prototipo software que permita a la empresa LIKATELEC automatizar las tareas manuales y repetitivas durante el mantenimiento preventivo de equipos de red en sus clientes utilizando *Python Scripting*.

LIKATELEC es una empresa dedicada a la prestación de servicios tecnológicos como: consultoría, implementación, soporte técnico y mantenimiento preventivo de equipos de red de sus clientes. La misma está conformada por 4 técnicos especialistas en varias áreas tecnológicas, como: Datacenter, Colaboración, Wireless, Seguridad, etc., y como parte del giro de negocio de la empresa, se realiza mantenimiento preventivo de equipos de red de sus clientes, que en su mayoría poseen equipamiento CISCO. Por tanto, este Proyecto de Integración Curricular será limitado para esta marca específica.

El mantenimiento preventivo de equipos de red consiste en la realización de las siguientes tareas:

1. Acceder al CLI *(Command-Line Interface)* del equipo a través la dirección IP de gestión del mismo utilizando un usuario y contraseña.
2. Ejecutar comandos para crear un archivo de respaldo de configuración (*show running-config, show vlan brief, etc.*).
3. Ejecutar comandos para obtener parámetros de operación generales del equipo, como: utilización de CPU, utilización de memoria, utilización de interfaces, parámetros ambientales del equipo, entre otras (*show ip interface brief, show memory, show process cpu history, show environment all, etc.*).
4. Elaborar y entregar un informe técnico que detalla la situación actual del equipo, en conjunto con archivos de respaldo de configuración al cliente.

Por esta razón, el presente Proyecto de Integración Curricular se enfocará principalmente en la automatización de las tareas antes descritas, aprovechando la potencialidad de *Python Scripting* y su aplicación en la programabilidad de redes. En consecuencia, se diseñará e implementará un prototipo de aplicación de escritorio con interfaz gráfica que se ejecutará en el computador de cada técnico de la empresa, por lo que no es necesario autenticar al mismo para utilizar el prototipo.

El prototipo constará de un *script* base (archivo ejecutable diseñado en *Python*), el cual tendrá parámetros configurables para la conexión al CLI de los equipos de red, y un

directorio llamado “Archivos de Configuración”. En este directorio, se encontrarán dos archivos que serán utilizados por el *script* base para su funcionamiento: uno para establecer los parámetros de conexión al CLI de los equipos y los comandos que se podrán ejecutar; y otro que se usará como documento base para la elaboración del informe técnico de cada equipo. Los parámetros de conexión son: dirección IP de gestión, usuario y contraseña. Los archivos de éste directorio serán modificados por el técnico de la empresa según sea el requerimiento del Cliente.

Una vez que el técnico de la empresa haya modificado los archivos antes descritos, el mismo ejecutará el *script* base que le permitirá interactuar con la interfaz gráfica de la aplicación de escritorio, donde se podrá realizar diferentes acciones que permitan la automatización de las tareas de un mantenimiento preventivo de equipos de red. Las acciones que se podrán realizar son: obtener respaldos de configuración, generar informes, visualizar el resultado de cada una de estas y salir de la interfaz gráfica. Adicionalmente, el *script* base creará un directorio llamado “Entregables” en la misma ubicación del *script* para almacenar los resultados de cada acción. Los documentos entregables son: respaldos de configuración e informe técnico de cada equipo. En la Figura 1.1 se muestra un esquema de funcionamiento del prototipo.

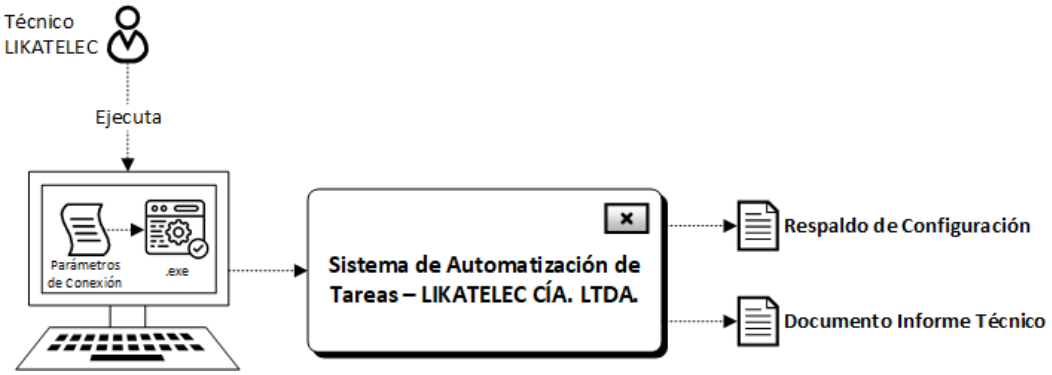

Figura 1.1. Esquema de funcionamiento del prototipo.

Para obtener respaldos, el técnico de la empresa realizará dos acciones: la primera será seleccionar los equipos de los cuales se requiere obtener los respaldos de configuración y la segunda, será seleccionar los comandos a ejecutar en los mismos. Posteriormente, el técnico de la empresa pulsará el botón “Obtener Respaldos” y el *script* base se conectará al CLI de los equipos seleccionados, ejecutará los comandos seleccionados y generará un archivo de respaldo de configuración por cada comando para cada equipo. Estos archivos de respaldo de configuración serán almacenados en el directorio “Entregables”.

Para generar informes, el técnico de la empresa seleccionará qué salidas de comandos se utilizarán para crear un documento de informe técnico. Este documento estará basado en un formato predefinido por la empresa LIKATELEC. Posteriormente, el técnico de la empresa pulsará el botón “Generar Informes” y el *script* base utilizará el contenido de los archivos de respaldo de configuración correspondientes a los comandos seleccionados, y generará un archivo de informe técnico, utilizando el documento base existente en el directorio “Archivos de Configuración” (por cada equipo que haya sido respaldado previamente). Estos archivos de informe técnico serán almacenados en el directorio “Entregables”.

Finalmente, el técnico de la empresa tendrá la opción de salir de la interfaz gráfica, lo cual terminará la ejecución de la aplicación.

El sistema realizará gestión de errores y constará de cuatro módulos:

- **Módulo de Autenticación para Acceso a CLI**

  En este módulo se permitirá el ingreso de los parámetros de conexión al CLI a los equipos de red. Estos parámetros son: dirección IP, usuario y contraseña de acceso a cada equipo.

- **Módulo de Configuración de *Script* en *Python***

  En este módulo se configurará un *script* base que será usado por el **Módulo de Autenticación para Acceso a CLI** de los equipos de red. Este *script* tendrá parámetros configurables los cuales dependerán de cada cliente.

- **Módulo de Procesamiento**

  En este módulo se ejecutará el *script* base para la obtención y procesamiento de la información originada por los comandos ejecutados por el *script* base. Este módulo será transparente al usuario.

- **Módulo de Visualización y Presentación de Resultados**

  En este módulo se visualizará la información obtenida en el **Módulo de Procesamiento** y se dará la opción de definir las secciones de información que serán parte del informe de resultados a través de la utilización de filtros que permitan ajustar el contenido del mismo según los requerimientos del cliente.

De acuerdo a la metodología ALM, se definen las siguientes tres etapas:

A. Gobernanza

Se estudiará las herramientas necesarias para el entendimiento global del prototipo, así como: *Python Scripting* y su aplicación en la programabilidad de redes; el acceso al CLI de CISCO para la ejecución de comandos; y la metodología que se utilizará para el desarrollo del prototipo [1]. Se resumirá la metodología ALM (*Application Lifecycle Management*) [2] para realizar el Diseño e Implementación del prototipo.

Basándose en la metodología ALM, se organizarán tareas y adecuarán tiempos para la resolución de las mismas. Siguiente, se obtendrán los Requerimientos Funcionales y No Funcionales del prototipo.

Para lograr cumplir con los requerimientos del prototipo se realizarán diagramas que faciliten la comprensión de las tareas a realizar, como lo son: Diagrama de Clases y Diagrama de Actividades.

La lógica de funcionamiento del *script* base será determinada a través de un Diagrama de Flujo.

B. Desarrollo

En base a las tareas definidas, y guiándose en el método de trabajo de ALM se procede con la instalación de las herramientas necesarias para proceder a programar cada módulo, siempre respetando los tiempos y el orden predefinido.

Se configurará el *script* base escrito en *Python*, que permitirá la conexión al CLI de los equipos de red y la ejecución de comandos CISCO para la obtención de información en base al Diseño realizado, y se creará el directorio para “Archivos de Configuración” donde se encontrarán los archivos necesarios para el funcionamiento del *script* (parámetros de conexión y documento base).

Se codificará una interfaz gráfica que permita al técnico ejecutar las acciones para obtener los respaldos de configuración y generar el documento de informe técnico por cada equipo previamente respaldado.

C. Operaciones

Una vez finalizada la lista de tareas de Gobernanza y Desarrollo indicadas por la metodología ALM, se procederá a evaluar del funcionamiento del prototipo, verificando que los Requerimientos Funcionales y No Funcionales se cumplan. Estas verificaciones se

realizarán sobre una simulación de red (por confidencialidad de los datos de un cliente), replicando el diseño de red total de un cliente determinado en la empresa. En el caso que se requiera, se realizará las correcciones necesarias para cumplir a cabalidad con los objetivos planteados para el prototipo.

1.4 MARCO TEÓRICO

En el **Apartado 1.4.1**, se estudiará *Python Scripting* y su aplicación en la programabilidad de redes. En el **Apartado 1.4.2** se estudiará la Gestión de Equipos de Red Cisco a través de CLI. Siguiente, en el **Apartado 1.4.3** la Metodología ALM y, por último, en el **Apartado 1.4.4**, las Tecnologías, Entornos de Desarrollo y Lenguajes de Programación para la Implementación de Prototipo.

1.4.1 *PYTHON SCRIPTING* Y SU APLICACIÓN EN LA PROGRAMABILIDAD DE REDES

La administración, operación y mantenimiento de las redes de información ha tenido un cambio significativo en los últimos tiempos. Según la referencia [3], las redes de información tradicionales no pueden responder de una manera ágil y eficiente a cambios repentinos e impredecibles que una organización y/o empresa requiera en un tiempo determinado. En este contexto, es donde la programabilidad de redes ha adquirido una aplicación cada vez más significativa para llevar a cabo proceso de automatización en las infraestructuras de redes tradicionales.

La automatización de la red es el proceso que permite automatizar tareas de administración, operación y mantenimiento de una red, por medio de la utilización de herramientas de software y/o librerías para diferentes lenguajes de programación como *Python* [4]. Debido a su facilidad de uso y flexibilidad para crear *scripts*, *Python* es uno de los lenguajes de programación que mejor se ha adaptado para llevar a cabo procesos de automatización en infraestructuras de redes tradicionales [5]. Este proceso tiene como objetivo reducir el tiempo para implementar nuevas configuraciones y/o cambios sobre la infraestructura de red, obtener información de la misma para su posterior análisis y mejoramiento, entre otras; reduciendo significativamente la inversión de tiempo que el Administrador de Red debe realizar en este tipo de tareas que son de carácter manual y repetitivo.

1.4.2 GESTIÓN DE EQUIPOS DE RED CISCO A TRAVÉS DE CLI

Cisco Systems es una empresa estadounidense, con sede matriz en San José, California, fundada en el año de 1984 por Leonard Bosack y Sandy Lerner. Esta empresa se dedica a la fabricación, comercialización y mantenimiento de equipos de Telecomunicaciones, para pequeñas, medianas y grandes empresas a nivel mundial [6].

Para gestionar estos equipos, se tiene un acceso a la línea de comandos (CLI). Este acceso permite ejecutar comandos para configuración y/u obtención de información del equipo.

El acceso CLI de Cisco se divide en varios modos de comandos, como son: ***User EXEC*** **(modo *EXEC* de usuario),** ***Privileged EXEC*** **(modo *EXEC* privilegiado),** ***Global Configuration*** **(modo de configuración global)**, entre otros. En la Tabla 1.1. se muestra un resumen de los modos de comandos.

Tabla 1.1. Resumen Modos de Comandos [7].

Modo	Método de Acceso	Aviso del Sistema	Método de Salida	Acerca de este Modo
Acceso Usuario *(User EXEC)*	Iniciar una sesión mediante Telnet, SSH o consola.	**Switch>**	Ingresar el comando ***logout*** o ***quit***	Usar este modo para: cambiar la configuración del terminal, realizar pruebas básicas y mostrar información del sistema.
Acceso Privilegiado *(Privileged EXEC)*	Mientras se está en el modo EXEC de usuario, ingresar el comando ***enable.***	**Switch#**	Ingresar el comando ***disable*** para salir	Usar este modo para verificar los comandos que se ha ingresado. Utilizar una contraseña para proteger el acceso a este modo.
Configuración Global *(Global Configuration)*	En el modo EXEC privilegiado, ingresar el comando ***configure***.	**Switch(config)#**	Para salir al modo *EXEC* privilegiado, ingresar el comando ***exit*** o ***end,*** o presionar la combinación ***Ctrl+Z***	Utilizar este modo para configurar parámetros que se aplican a todo el equipo.

1.4.3 METODOLOGÍA ALM

La Gestión del Ciclo de Vida de Aplicaciones ALM (*Application Lifecycle Management*) es un proceso de gestión que permite controlar todas las etapas en el desarrollo e implementación de aplicaciones de software, desde su concepción hasta el final de su vida útil. Esto incluye especificar, diseñar, desarrollar, probar y desplegar la aplicación [8].

Este proceso de gestión se puede dividir en tres etapas [9]:

1. **Gobernanza:** En esta etapa se abarca todos los pasos de toma de decisiones y gestión de proyectos que se extienden a lo largo del ciclo de vida de una aplicación. En ésta se incluye gestión de requisitos, gestión de recursos y gestión del sistema, como seguridad de datos, acceso de usuarios, revisión, verificación y control de implementación.
2. **Desarrollo:** En esta etapa se abarca todos los pasos referentes a la creación misma de la aplicación y, adicionalmente, los desarrollos en curso, como revisiones y actualizaciones. En ésta se incluye la identificación de problemas actuales, la planificación, el diseño, la construcción y la prueba de la aplicación.
3. **Operaciones:** En esta etapa se despliega la aplicación desarrollada. Esta etapa comienza poco antes de la implementación e implica ejecutar, monitorear y mantener la aplicación hasta el final de su vida útil.

En la Figura 1.2 se muestran las tres etapas de la Gestión del Ciclo de Vida de Aplicaciones y la relación entre las mismas.

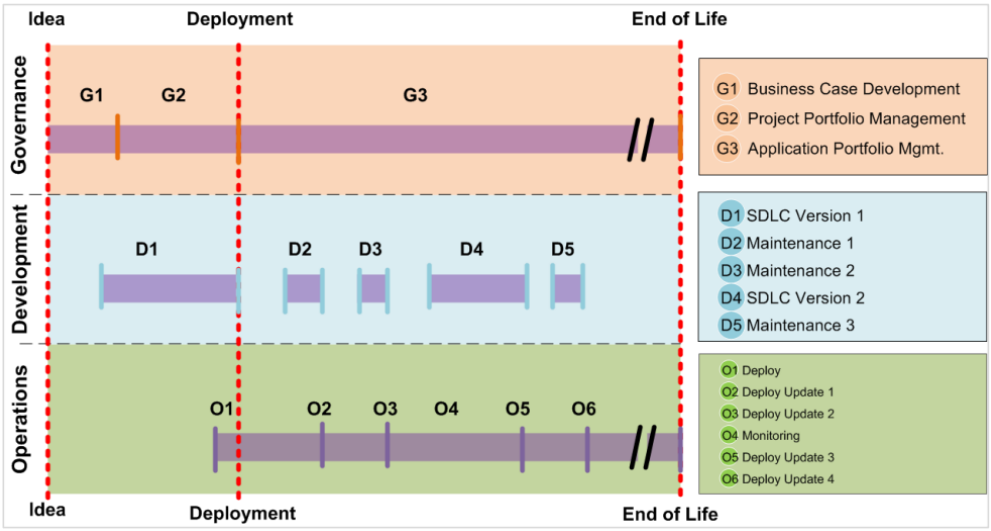

Figura 1.2. Etapas de la Gestión del Ciclo de Vida de Aplicaciones [10].

1.4.4 TECNOLOGÍAS, ENTORNOS DE DESARROLLO Y LENGUAJES DE PROGRAMACIÓN PARA IMPLEMENTACIÓN DE PROTOTIPO

1.4.4.1 Arquitectura en Capas

Esta arquitectura permite separar una aplicación en capas, de tal manera que cada capa tenga una responsabilidad específica. Una capa superior puede utilizar los servicios de una capa inferior, pero una capa inferior no puede utilizar los servicios de una capa superior [11].

Dentro de esta arquitectura se definen generalmente 3 capas que son: **Capa de Presentación, Capa de Negocio** y **Capa de Acceso a Datos,** sin embargo, para la implementación del prototipo se utilizarán únicamente la **Capa de Presentación** y la **Capa de Negocio**, ya que no se necesita gestionar una base de datos.

La **Capa de Presentación** es la interfaz gráfica que permitirá ejecutar las acciones para automatizar las tareas en un mantenimiento de equipos de red. Esta capa se comunica únicamente con la **Capa de Negocio**.

La **Capa de Negocio** define el flujo de trabajo de la aplicación, la misma está conformada por diferentes reglas (*script* base) que se deben cumplir para un correcto funcionamiento del prototipo.

1.4.4.2 Entorno de Desarrollo

El Entorno de Desarrollo Integrado, por sus siglas en inglés IDE *(Integrated Development Environment),* es un software que permite la creación de aplicaciones de escritorio o móviles, utilizando una interfaz gráfica de usuario [12].

Un IDE está formado por al menos los siguientes componentes:

- **Editor de Código Fuente:** Se encarga de proporcionar un editor de texto al desarrollador para escribir el código fuente de una aplicación utilizando algún lenguaje de programación. Este editor de texto también proporciona funcionalidades como: autocompletado y/o sugerencias de código, verificación de posibles errores por sintaxis, sangrado de código para un mejor entendimiento del mismo y ayuda visual para el desarrollador utilizando diferentes colores en el código fuente.

- **Compilador Local:** Se encarga de traducir el código fuente en código de máquina (binario), de tal manera que la máquina pueda entender y ejecute las tareas para las cuales fue desarrollada la aplicación.
- **Depurador:** Se encarga de mostrar la ubicación de un error dentro del código fuente. Adicionalmente, puede recomendar acciones al desarrollador para corregir el error detectado.

1.4.4.3 Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente gratuito desarrollado por Microsoft, el cual está disponible para sistemas operativos Windows, macOS y Linux. Según la referencia [13], en una encuesta realizada por *Stack Overflow* en mayo de 2022 a más de 70,000 desarrolladores, con un 74.48% *VS Code* se ha convertido en el Entorno de Desarrollo preferido por los mismos, debido a sus facilidades para escribir, depurar y probar código.

Entre sus principales características y funcionalidades se tienen [14]:

- **Multiplataforma:** Esta característica permite ejecutar *VS Code* en sistemas operativos Windows, macOS y Linux.
- ***IntelliSense:*** Esta característica permite proporcionar sugerencias de código para el desarrollador, así como las funcionalidades de autocompletado y resaltado de sintaxis al momento de escribir el código fuente de una aplicación.
- **Depuración de Código:** Esta característica permite detectar errores en el código fuente de la aplicación, indicando en qué línea se encuentra el error y una pequeña descripción del mismo.
- **Extensiones:** Esta característica permite personalizar y agregar funcionalidades de forma modular y aislada a *VS Code*. Permite instalar extensiones desarrolladas por terceros para admitir casi cualquier lenguaje de programación como *Python*, C#, C++, entre otros. Adicionalmente, permite agregar nuevos temas al editor de código fuente y conectar *VS Code* con otros servicios.

1.4.4.4 Microsoft Visual Studio

Microsoft Visual Studio es un IDE para desarrolladores de .NET y C++ en Windows. Permite el desarrollo de aplicaciones multiplataforma para dispositivos móviles y de escritorio, y está disponible para sistemas operativos Windows y macOS. Tiene una versión gratuita para estudiantes, colaboradores de código abierto y usuarios particulares [15].

Entre sus principales características y funcionalidades se tienen [16]:

- **Multiplataforma:** Esta característica permite ejecutar ***Microsoft Visual Studio*** en sistemas operativos Windows y macOS.
- **Subrayados Ondulados y Acciones Rápidas:** Es una característica visual que permite alertar al desarrollador sobre posibles errores en el código fuente a medida que se escribe, de tal manera, que el desarrollador pueda corregir los mismos antes de la compilación o tiempo de ejecución del código fuente.
- **Limpieza de Código:** Esta característica permite dar formato al código fuente, como, por ejemplo, el sangrado que debe tener el mismo a medida que se escribe; de tal manera que sea legible y fácil de entender.
- **Refactorización:** Esta característica permite cambiar el nombre de una variable en todo el código fuente, extraer una o más líneas de código en un nuevo método y cambiar el orden de los parámetros de método.
- ***IntelliSense*:** Esta característica permite proporcionar sugerencias de código para el desarrollador, así como las funcionalidades de autocompletado y resaltado de sintaxis al momento de escribir el código fuente de una aplicación.

1.4.4.5 Python

Python es un lenguaje de programación de código abierto, interpretado de alto nivel y orientado a objetos. Fue creado por Guido van Rossum en 1991. Su sintaxis es simple y fácil de entender, lo que permite a los desarrolladores crear aplicaciones con un código fuente legible, de tal manera que el mantenimiento de la aplicación no sea costoso [17].

Dentro de sus principales aplicaciones se tienen [18]:

- ***Scripting* y Automatización:** *Python* se utiliza para *scripting* (ejecución de una secuencia de comandos) y automatización de tareas en diferentes áreas como:

redes de información, análisis de datos, etc., optimizando el tiempo utilizado para la ejecución de estas tareas en las diferentes áreas.

- **Desarrollo de Software:** *Python* se utiliza para el desarrollo de software en diferentes plataformas, ya que es compatible con muchos ámbitos de operación, sistemas operativos para computadoras y dispositivos móviles.
- **Análisis de Datos:** *Python* se utiliza para el análisis de datos, ya que permite crear representaciones visuales legibles de bloques de datos complejos.
- ***Machine Learning* e Inteligencia Artificial:** *Python* se utiliza en inteligencia artificial debido a su potencialidad en la automatización de tareas, lo cual permite obtener resultados sencillos a partir del análisis de un bloque de datos complejo.

2 METODOLOGÍA

En el presente Capítulo se mostrarán las particularidades de la construcción de la aplicación de escritorio orientada a la automatización de tareas de mantenimiento de equipos de red: las decisiones de diseño tomadas y los diferentes detalles de desarrollo realizadas.

A lo largo del **Apartado 2.1**, correspondiente a **Gobernanza**, se definirán las principales pautas para el desarrollo de la aplicación de escritorio, por otra parte, en el **Apartado 2.2**, correspondiente a **Desarrollo**, se explorarán las decisiones de desarrollo tomadas y, finalmente; en el **Apartado 2.3,** correspondiente a **Operaciones**, se mostrarán los resultados obtenidos después de finalizar la etapa de implementación del prototipo.

De acuerdo a la metodología *Application Lifecycle Management* (ALM), la primera etapa dentro del desarrollo de aplicaciones de software es la etapa de Gobernanza; en la misma se abarca todos los pasos de toma de decisiones y gestión de proyectos que se extienden a lo largo del ciclo de vida de una aplicación.

2.1 GOBERNANZA

Durante esta sección se mostrarán las pautas tomadas para desarrollar la aplicación de escritorio; así como, los Requerimientos Funcionales y No Funcionales y los diferentes diagramas utilizados en el prototipo.

2.1.1 PLANTEAMIENTO DEL TABLERO ALM

Se utilizó la herramienta *Jira* [19] para realizar el Tablero ALM y actualizar los resultados conforme se vayan consiguiendo los objetivos planteados. Según se puede observar en la Figura 2.1, durante la Fase de Gobernanza se deben realizar once tareas en total, de las cuales se han realizado cuatro y siete están en progreso.

Figura 2.1. Planteamiento del Tablero ALM en Etapa de Gobernanza.

2.1.2 REQUERIMIENTOS FUNCIONALES (RF)

Los Requerimientos Funcionales definen lo que debe hacer el prototipo, de tal manera que se cumpla con los objetivos planteados inicialmente [20]. Estos requerimientos se detallan en las siguientes Historias de Usuario.

Tabla 2.1. Requerimiento Funcional 01.

RF 01	**Configurar Parámetros de Conexión al CLI**
Usuario:	Técnico LIKATELEC
Prioridad:	Alta
Descripción:	
Se requiere configurar los parámetros de conexión al cli de los equipos.	

Criterios de Aceptación:
• El Prototipo requerirá de 3 parámetros para establecer la conexión al CLI de los equipos de Red (Dirección IP, Usuario y Contraseña). • El parámetro Dirección IP consiste en cuatro octetos (secuencia de números binarios de 8 bits) expresados en formato decimal y separados por puntos. Cada número entre los puntos debe estar en el rango de 0 a 255 [21]. • El parámetro Usuario consiste en una combinación de letras y/o números sin ninguna restricción. • El parámetro Contraseña consiste en una combinación de letras y/o números sin ninguna restricción. • El Prototipo validará que el parámetro Dirección IP cumpla con las condiciones antes descritas. • Estos parámetros de Conexión al CLI serán configurados en un archivo .xlsx externo que será leído por el Prototipo al momento de ejecutar el mismo. • El Prototipo validará la existencia del archivo .xlsx externo e informará cuando el mismo no exista.

Tabla 2.2. Requerimiento Funcional 02.

RF 02	**Configurar Comandos para Respaldo de Configuración**
Usuario:	Técnico LIKATELEC
Prioridad:	Alta
Descripción	
Se requiere configurar los comandos para respaldo de configuración de los equipos.	
Criterios de Aceptación	
• El Prototipo requerirá de una secuencia de Comandos que serán ejecutados en el CLI de cada equipo de acuerdo al requerimiento del cliente. • Un Comando consiste en una combinación de palabras en inglés que permite obtener información de los equipos. • Cada Comando se debe escribir de forma completa, es decir, no se debe utilizar abreviaciones para evitar ambigüedades al momento de ejecutar el mismo. • Estos Comandos serán configurados en un archivo .xlsx externo que será leído por el Prototipo al momento de ejecutar el mismo. • El Prototipo validará la existencia del archivo .xlsx externo e informará cuando el mismo no exista.	

Tabla 2.3. Requerimiento Funcional 03.

RF 03	**Crear Archivos de Respaldo de Configuración**
Usuario:	Técnico LIKATELEC
Prioridad:	Alta
Descripción	
Se requiere obtener archivos de respaldo de configuración de cada equipo de red.	
Criterios de Aceptación	
• El Prototipo creará archivos de respaldo de configuración en formato .txt. Para esto, primero se seleccionará los equipos y segundo los comandos a ejecutar. • El Prototipo se conectará al CLI de los equipos seleccionados en modo Acceso Privilegiado (Tabla 1.1), ejecutará los comandos seleccionados y creará un archivo de respaldo de configuración por cada comando para cada equipo. • El Prototipo validará la conexión al CLI de los equipos.	

• El Prototipo validará que, al menos un equipo y un comando estén seleccionados, para crear el archivo de respaldo de configuración. • El Prototipo creará un directorio para guardar los archivos de respaldo de configuración.

Tabla 2.4. Requerimiento Funcional 04.

RF 04	**Ver Respaldos de Configuración**
Usuario:	Técnico LIKATELEC
Prioridad:	Media
Descripción	
Se requiere visualizar los archivos de respaldo de configuración de los equipos y el contenido de los mismos.	
Criterios de Aceptación	
• El Prototipo permitirá visualizar los archivos de respaldo de configuración. • El Prototipo permitirá visualizar el contenido de cada archivo. • El Prototipo no permitirá modificar el contenido de los archivos.	

Tabla 2.5. Requerimiento Funcional 05.

RF 05	**Crear Informes Técnicos**
Usuario:	Técnico LIKATELEC
Prioridad:	Alta
Descripción	
Se requiere crear documentos de informe técnico para cada equipo en formato .docx y .pdf en base a documento base definido por la empresa LIKATELEC.	
Criterios de Aceptación	
• El Prototipo creará un documento de informe técnico por cada equipo. Para esto, se seleccionará las salidas de comandos que el Prototipo utilizará para crear el documento de informe técnico. • El Prototipo utilizará el contenido de los archivos de respaldo de configuración correspondientes a los comandos seleccionados, y generará un archivo de informe técnico, utilizando el documento base. • El Prototipo validará que, exista el archivo del documento base, para generar el documento de informe técnico. • Los documentos de informe técnico tendrán formato .docx y .pdf y serán almacenados en dos directorios respectivamente.	

Tabla 2.6. Requerimiento Funcional 06.

RF 06	**Ver Informes Técnicos**
Usuario:	Técnico LIKATELEC
Prioridad:	Media
Descripción	
Se requiere visualizar los documentos de informe técnico y el contenido de los mismos.	
Criterios de Aceptación	
• El Prototipo permitirá visualizar los documentos de informe técnico en formato .pdf. • El Prototipo permitirá visualizar el contenido de cada documento .pdf. • El Prototipo no permitirá modificar el contenido de los documentos de informe técnico.	

2.1.3 REQUERIMIENTOS NO FUNCIONALES (RNF)

Los Requerimientos No Funcionales representan características generales y restricciones del prototipo [22]. La evaluación de estos requerimientos se realizará a través de una encuesta de 5 niveles utilizando la escala de *Likert* [23], la cual permite cuestionar a una persona sobre su nivel de acuerdo o desacuerdo con una declaración.

Los Requerimientos No Funcionales se detallan en las siguientes Historias de Usuario.

Tabla 2.7. Requerimiento No Funcional 01.

RNF 01	**Usabilidad del Prototipo**
Usuario:	Técnico LIKATELEC
Prioridad:	Alta
Descripción	
Se requiere tener una interfaz gráfica amigable, intuitiva y fácil de manejar para que el técnico pueda utilizarla sin mayor dificultad.	
Criterios de Aceptación	
• El Prototipo tendrá controles de selección mínimos y necesarios para un funcionamiento óptimo. • La interfaz gráfica tendrá un tamaño de letra adecuado.	

Tabla 2.8. Requerimiento No Funcional 02.

RNF 02	**Ejecución Lineal del Prototipo**
Usuario:	Técnico LIKATELEC
Prioridad:	Alta
Descripción	
Se requiere un flujo de ejecución ordenado y lógico para que el técnico no cometa errores en la ejecución de las tareas de automatización, es decir tendrá una ejecución lineal el prototipo.	
Criterios de Aceptación	
• El Prototipo alertará de posibles acciones incorrectas ejecutadas por el técnico.	

Tabla 2.9. Requerimiento No Funcional 03.

RNF 03	**Compatibilidad del Prototipo**
Usuario:	Técnico LIKATELEC
Prioridad:	Alta
Descripción	
Se requiere que el Prototipo sea compatible con el sistema Operativo Windows debido a la preferencia del mismo por los técnicos que conforman la empresa LIKATELEC.	
Criterios de Aceptación	
• El Prototipo será una Aplicación de Escritorio.	

Tabla 2.10. Requerimiento No Funcional 04.

RNF 04	**Ingreso de datos al Prototipo**
Usuario:	Técnico LIKATELEC
Prioridad:	Alta
Descripción	
Se requiere ingresar datos al Prototipo a través de un archivo externo para el funcionamiento del mismo.	
Criterios de Aceptación	
• El Prototipo requerirá de datos para su funcionamiento. Estos datos son: parámetros de Conexión al CLI de los equipos y comandos a ejecutar en los mismos. • Los parámetros de Conexión al CLI y los comandos serán configurados en un archivo .xlsx externo que será leído por el Prototipo al momento de ejecutar el mismo.	

2.1.4 DISEÑO DE LA CAPA DE NEGOCIO

La Capa de Negocio define las diferentes reglas que se deben cumplir para un correcto funcionamiento del Prototipo.

2.1.4.1 Diseño del Diagrama de Clases

Se utilizó la herramienta *Lucidchart* [24] para realizar el Diagrama de Clases. El Diagrama de Clases se utiliza para representar los elementos que componen el prototipo [25].

El prototipo está compuesto por una clase estática **ScriptBase** y una clase **Equipo.** La clase estática **ScriptBase,** tiene 11 métodos estáticos de tipo público, para implementar las funcionalidades que debe cumplir el prototipo. La clase **Equipo**, tiene 5 atributos de tipo privado: la dirección IP de gestión del equipo, el nombre de usuario, la contraseña y el fabricante del tipo *string,* y el último atributo estado de conexión de tipo *bool*.

En la Figura 2.2 se muestra el Diagrama de Clases del Prototipo:

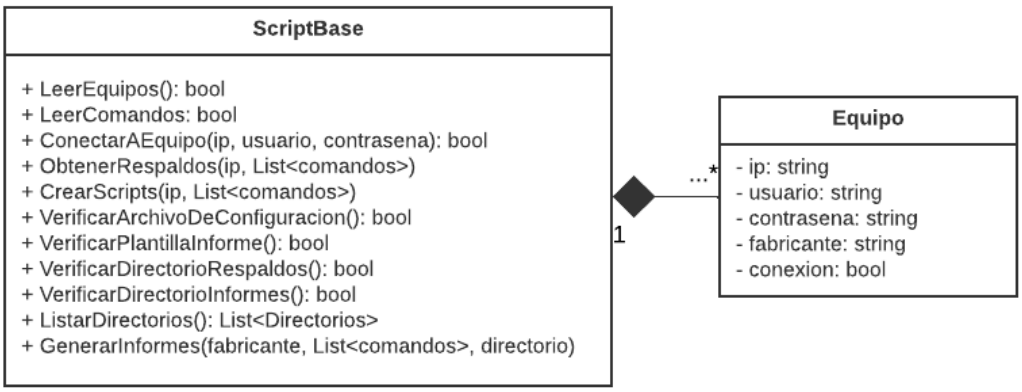

Figura 2.2. Diagrama de Clases del Prototipo.

La clase estática **ScriptBase**, tiene 11 métodos estáticos de tipo público:

- ***LeerEquipos().*** Este método es usado por el prototipo para leer los datos del archivo .xlsx externo correspondientes a la dirección IP del equipo, el usuario y la contraseña e instanciar un objeto de la clase **Equipo.**
- ***LeerComandos().*** Este método es usado por el prototipo para leer los datos del archivo .xlsx externo correspondientes a los comandos a ejecutar en el CLI de los equipos y obtener información de los mismos.
- ***ConectarAEquipos(str ip, str usuario, str contraseña).*** Este método es usado por el prototipo para establecer conexión al CLI de los equipos teniendo como parámetros de entrada la dirección IP del equipo, el usuario y la contraseña. Este método devuelve un valor ***True*** o ***False*** dependiendo si el prototipo se pudo o no conectarse al CLI de los equipos respectivamente.
- ***ObtenerRespaldos(str ip, List<str comandos>).*** Este método es usado por el prototipo para ejecutar los comandos seleccionados por el técnico en el CLI de los equipos y con la salida de los mismos, crea los archivos de respaldo de configuración en el directorio respectivo.
- ***CrearScripts(str ip, List<str comandos>).*** Este método es usado por el prototipo para generar los *scripts* de comandos ejecutados en el CLI de los equipos.
- ***VerificarArchivoDeConfiguracion().*** Este método es usado por el prototipo para validar la existencia del archivo .xlsx externo en el directorio respectivo. Este método devuelve un valor ***True*** o ***False*** dependiendo si el prototipo ha encontrado o no el archivo .xlsx externo respectivamente.
- ***VerificarPlantillaInforme().*** Este método es usado por el prototipo para validar la existencia del archivo de documento base en el directorio respectivo. Este método devuelve un valor ***True*** o ***False*** dependiendo si el prototipo ha encontrado o no el archivo de documento base respectivamente.
- ***VerificarDirectorioRespaldos().*** Este método es usado por el prototipo para validar la existencia del directorio respectivo donde se almacena los archivos de respaldo de configuración creados por el prototipo. Este método devuelve un valor ***True*** o ***False*** dependiendo si el prototipo ha encontrado o no el directorio respectivamente.

- ***VerificarDirectorioInformes().*** Este método es usado por el prototipo para validar la existencia del directorio respectivo donde se almacena los archivos de informe técnico creados por el prototipo. Este método devuelve un valor ***True*** o ***False*** dependiendo si el prototipo ha encontrado o no el directorio respectivamente.
- ***ListarDirectorios().*** Este método es usado por el prototipo para enumerar los directorios donde ha almacenado los archivos de respaldo de configuración. Este método devuelve una lista de directorios.
- ***GenerarInformes(str fabricante, List<str comandos>, str directorio).*** Este método es usado por el prototipo para crear los documentos de informe técnico, a partir del contenido de los archivos de respaldo de configuración, que corresponden a las salidas de comandos seleccionados por el técnico.

2.1.4.2 Diseño del Diagrama de Actividades

Se utilizó la herramienta *Lucidchart* para realizar el Diagrama de Actividades. A través de este Diagrama, se podrá establecer la secuencia lógica de funcionamiento del Prototipo, permitiendo aclarar cuál es el orden de ejecución de las tareas y las condiciones previas que se deben cumplir [26].

Se debe tomar en cuenta que previo a la ejecución del prototipo, se debe crear el archivo externo .xlsx y el archivo de documento base para la elaboración del informe técnico de cada equipo. Adicionalmente, se debe ubicar los mismos en el directorio de “Archivos de Configuración”. En la Tabla 2.11 se detalla el propósito de los archivos mencionados.

Tabla 2.11. Archivos de Configuración.

Nombre del archivo	Extensión	Descripción
PARAMETROS_COMANDOS	.xlsx	Permite establecer los parámetros de conexión al CLI de los equipos y los comandos que se podrán ejecutar.
PLANTILLA_INFORME	.docx	Documento base para la elaboración del informe técnico de cada equipo definido por la empresa LIKATELEC.

En la Figura 2.3 se muestra el Diagrama de Actividades del Prototipo:

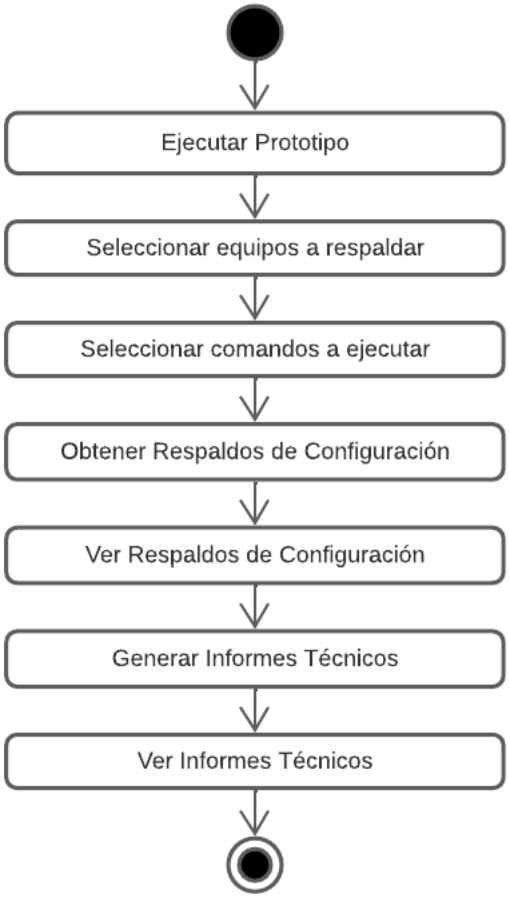

Figura 2.3. Diagrama de Actividades del Prototipo.

2.1.5 DISEÑO DE LA CAPA DE PRESENTACIÓN

La Capa de Presentación es la interfaz gráfica que permitirá al técnico ejecutar las acciones para automatizar las tareas en un mantenimiento de equipos de red. Esta capa se comunica únicamente con la Capa de Negocio.

2.1.5.1 Diseño de la Interfaz Gráfica

La interfaz gráfica del prototipo estará conformada por tres formularios. El formulario principal permitirá realizar todas las acciones relacionadas a la automatización de tareas de un mantenimiento de equipos de red y los formularios secundarios permitirán visualizar la información obtenida en cada tarea. Se diseñó la interfaz gráfica de forma secuencial, de tal manera que el técnico cometa la mínima cantidad de errores; es decir el flujo de actividades que el técnico debe realizar estará pre establecido, guiando al mismo a seguir un orden específico.

En la Figura 2.4 se muestra el diseño del formulario principal del prototipo, el mismo que consta de tres secciones:

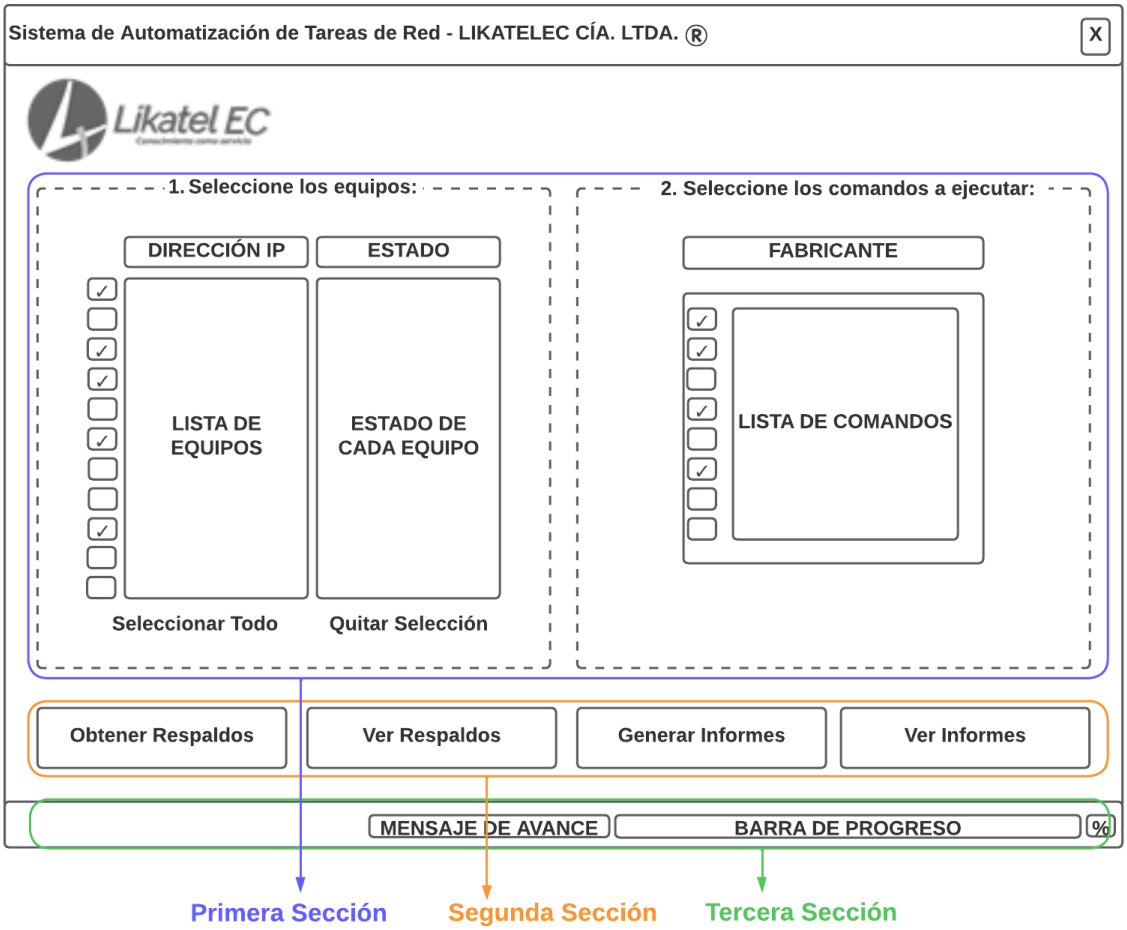

Figura 2.4. Formulario Principal del Prototipo.

En la primera sección del formulario principal del prototipo, el técnico podrá seleccionar los equipos de los cuales requiere obtener el respaldo de configuración y posteriormente, seleccionar los comandos que se ejecutarán en el CLI de los equipos. Por defecto, todos los equipos y todos los comandos estarán seleccionados al ejecutar el prototipo. Por otra parte, en la segunda sección, el técnico podrá realizar diferentes acciones que permitan la automatización de las tareas de un mantenimiento de equipos de red. Las acciones que el técnico podrá realizar son: Obtener Respaldos, Ver Respaldos, Generar Informes y Ver Informes. Finalmente, la tercera sección corresponderá a una barra de estado. La misma estará conformada por una sección para mostrar un mensaje de avance, una barra de progreso y una sección para mostrar el porcentaje de avance cuando el técnico realiza la tarea de Obtener Respaldos y Generar Informes.

A continuación, en la Figura 2.5 se muestra el diseño del segundo formulario del prototipo, el mismo que consta de dos secciones:

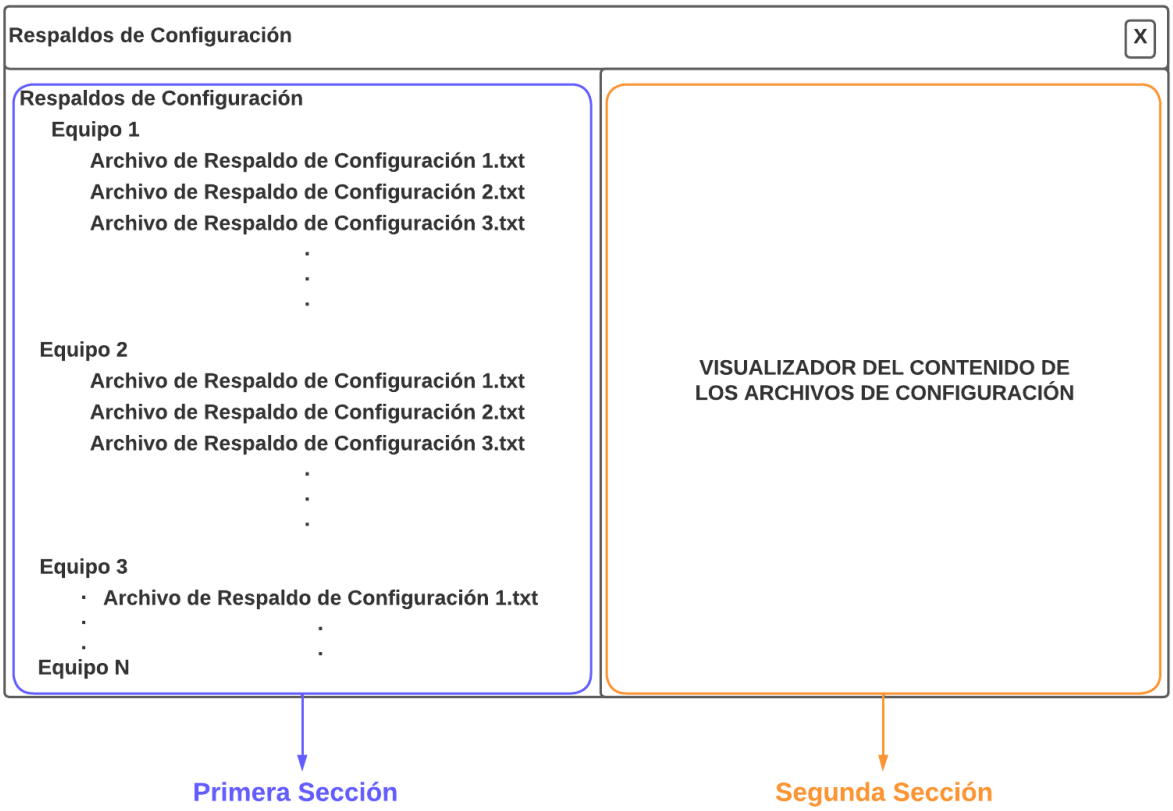

Figura 2.5. Segundo Formulario del Prototipo.

En la primera sección del segundo formulario del prototipo, el técnico podrá visualizar los archivos de respaldo de configuración de los equipos que fueron seleccionados en el formulario principal (Figura 2.4). El formato de estos archivos será ***.txt.***

Por otra parte, en la segunda sección, en el mismo formulario el técnico podrá visualizar el contenido de los archivos de respaldo de configuración.

Finalmente, en la Figura 2.6 se muestra el diseño del tercer formulario del prototipo, el mismo que consta de dos secciones:

Figura 2.6. Tercer Formulario del Prototipo.

En la primera sección del tercer formulario del prototipo, el técnico podrá visualizar los documentos de Informe Técnico generados a partir de las salidas de comandos que fueron seleccionados en el formulario principal (Figura 2.4). El formato de estos archivos será ***.pdf***. Por otra parte, en la segunda sección, el técnico podrá visualizar el contenido de los documentos de Informe Técnico.

2.2 DESARROLLO

Durante esta sección se presentará el proceso de desarrollo y codificación del prototipo. En la Capa de Negocio se tendrán todas las funciones que permitirá al prototipo cumplir con los objetivos planteados descritas en el **Apartado 2.1.4**. La Capa de Presentación mostrará la interfaz gráfica del prototipo que se ha diseñado en el **Apartado 2.1.5**.

2.2.1 ACTUALIZACIÓN DEL TABLERO ALM

En la Figura 2.7 se observan las tareas que ya han sido completadas satisfactoriamente y las tareas que están en proceso.

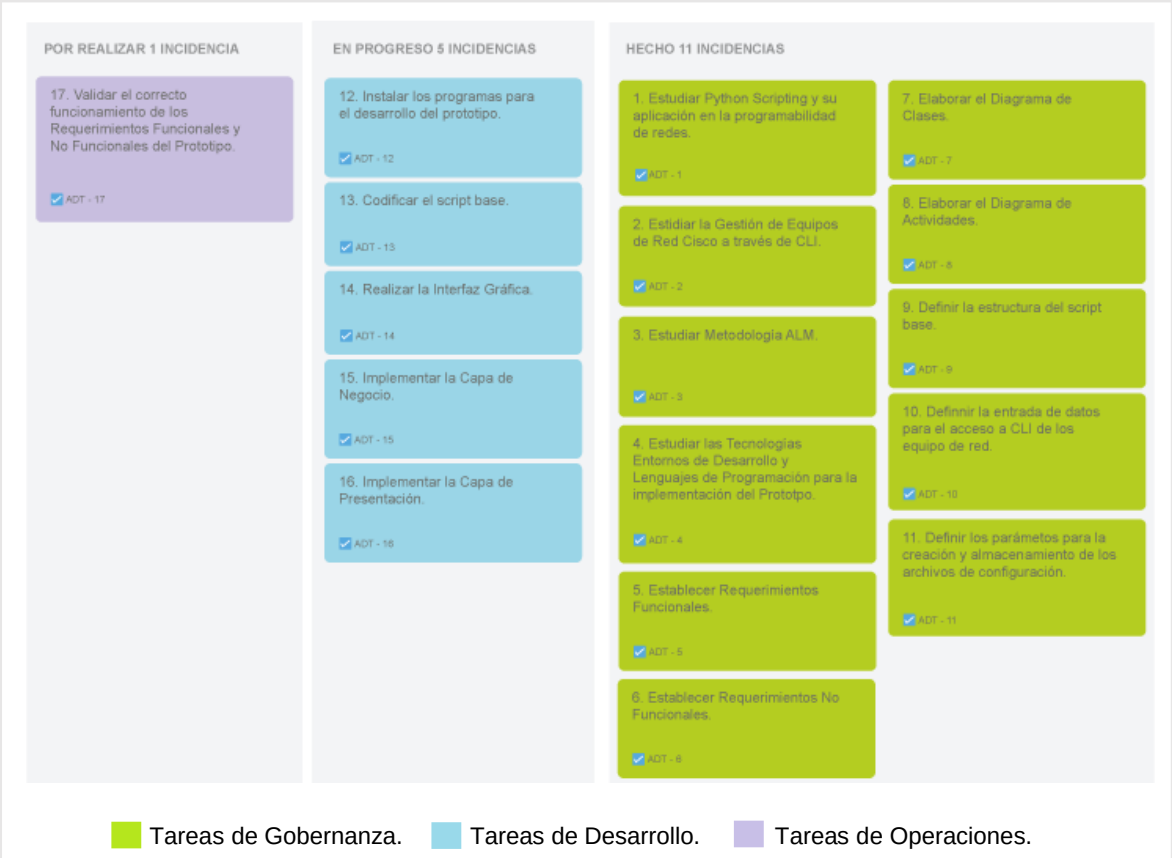

Figura 2.7. Actualización del Tablero ALM en la Etapa de Desarrollo.

2.2.2 IMPLEMENTACIÓN DE LA CAPA DE NEGOCIO

La Capa de Negocio consiste en la lógica que realizará las funciones principales del prototipo.

La clase **Equipo**, tiene 5 atributos de tipo privado:

- **ip** de tipo ***string*****:** Este atributo almacena el valor de la dirección IP de gestión del equipo.
- **usuario** de tipo ***string*****:** Este atributo almacena el valor del nombre de usuario utilizado para realizar la conexión al CLI del equipo.

- **contrasena** de tipo ***string*:** Este atributo almacena el valor de la contraseña del usuario utilizado para realizar la conexión al CLI del equipo.
- **fabricante** de tipo ***string*:** Este atributo almacena el valor del fabricante del equipo.
- **conexion** de tipo ***bool*:** Este atributo almacena el valor del estado de conexión del equipo.

En el Código 2.1 se presenta una parte de la clase **Equipo.** En este fragmento de código se observa el constructor de la clase y parte de los atributos que tiene la misma. Para observar todo el código de la clase observe el **Anexo II.**

```
class Equipo:

    def __init__(self, direccionIP, nombreUsario, contrasena,
                 fabricante, estadoConexion=False):
        self.__ip = direccionIP
        self.__usuario = nombreUsario
        self.__contrasena = contrasena
        self.__fabricante = fabricante
        self.__conexion = estadoConexion

    @property
    def ip(self):
        return self.__ip

    @ip.setter
    def ip(self, direccionIP):
        self.__ip = direccionIP
```

Código 2.1. Clase Equipo del Prototipo.

En el Código 2.2 se presenta una parte de la clase estática **ScriptBase.** En este fragmento de código se observan las variables globales que usa la clase para realizar las diferentes tareas de automatización y uno de los métodos estáticos implementados para leer el archivo de configuración externo. Para observar todo el código de la clase observe el **Anexo II.**

```
class ScriptBase:
    lstEquipos = []
    lstComandos = []
    lstFabricantes = []
    clienteSSH = paramiko.SSHClient()
    rutaCompletaRespaldos = os.path.abspath(os.getcwd())+'\ENTREGABLES\RESPALDOS_DE_CONFIGURACION\\'
    rutaCompletaInformes = os.path.abspath(os.getcwd())+'\ENTREGABLES\INFORMES_TECNICOS_PDF\\'
    rutaRelativa = os.path.abspath(os.getcwd())+'\ENTREGABLES\\'

    @staticmethod
    def LeerEquipos():
        dfEquipos = pd.read_excel(os.getcwd() + '\ARCHIVOS_DE_CONFIGURACION\PARAMETROS_COMANDOS.xlsx'
        dfEquipos = dfEquipos.fillna('')

        contadorEquipos = 0
        patronDireccionIP = re.compile('^((25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])\.){3}(25[0-5]
```

Código 2.2. Clase ScriptBase del Prototipo.

2.2.3 IMPLEMENTACIÓN DE LA CAPA DE PRESENTACIÓN

Se ha utilizado el diseño de la interfaz gráfica descrito en el **Apartado 2.1.5.1** para implementar los tres formularios que conforman la Capa de Presentación del prototipo. Se ha seguido los parámetros definidos en los Requerimientos Funcionales y Requerimientos No Funcionales del **Apartado 2.1.2** y **Apartado 2.1.3**. Para observar toda la Implementación de la Capa de Presentación observe el **Anexo II.**

En la Figura 2.8. se muestra el formulario principal final implementado:

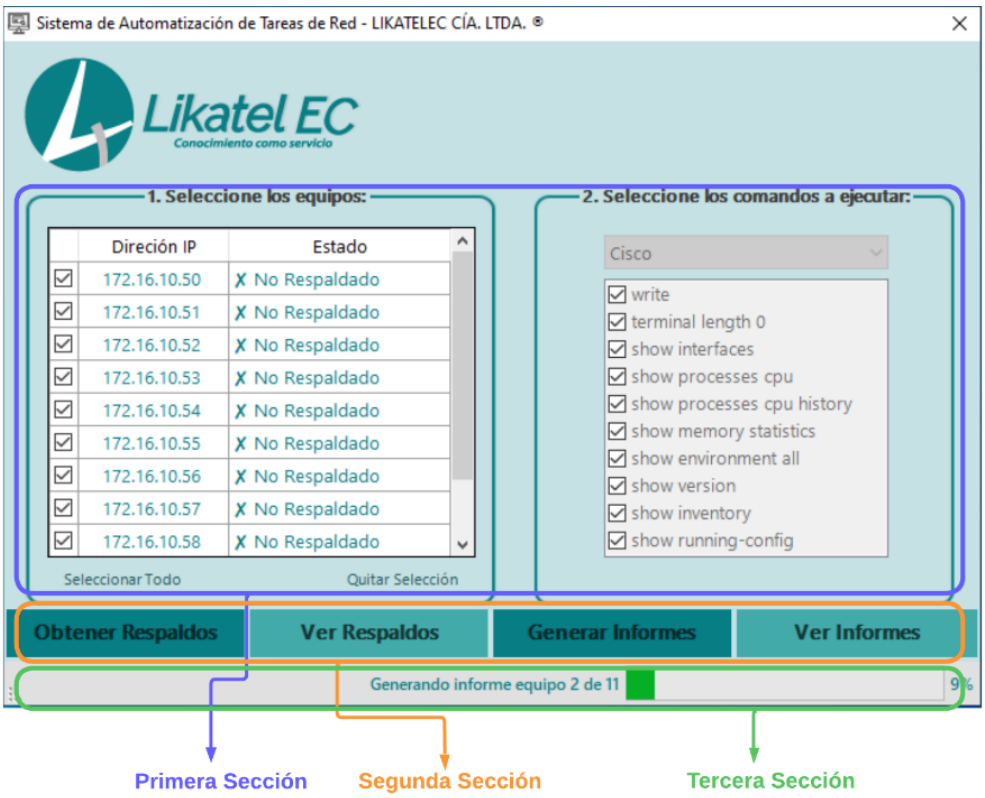

Figura 2.8. Formulario Principal del Prototipo.

Para ejecutar las tareas de automatización involucradas en un mantenimiento preventivo de equipos de red, el técnico de la empresa LIKATELEC hará *click* en los botones de la segunda sección del formulario principal del prototipo (Figura 2.8), permitiéndole realizar 4 acciones específicas que son: Obtener Respaldos, Ver Respaldos, Generar Informes y Ver Informes.

En el Código 2.3 se presenta una parte del método **ObtenerRespaldos.** Este método se ejecutará cuando el técnico haga *click* en el botón “Obtener Respaldos” y permitirá que el prototipo se conecte al CLI de los equipos de red y ejecute los comandos seleccionados en la primera sección del formulario principal (Figura 2.8). En este fragmento de código se

observan las variables usadas para implementar el método, la ejecución de un nuevo hilo para obtener los archivos de respaldo de configuración de los equipos de red y el control de errores del método. Para observar todo el código del método observe el **Anexo II.**

```
def ObtenerRespaldos(self, sender, e):
                equiposSeleccionados, lstEquiposSeleccionados = self.BuscarEquiposSeleccionados(sender, e)
                comandosSeleccionados, lstComandosSeleccionados = self.BuscarComandoSeleccionados(sender, e)

                if(equiposSeleccionados == True and comandosSeleccionados == True):
                        for indiceEquipo in lstEquiposSeleccionados:
                                self.dataGridViewEquipos.Rows[indiceEquipo].Cells['txtColEstado'].Value = "✗ No Respaldado"

                        self.dataGridViewEquipos.CurrentCell = None
                        self.dataGridViewEquipos.ReadOnly = True
                        self.dataGridViewEquipos.Rows[lstEquiposSeleccionados[0]].Cells['txtColEstado'].Value = "⌛ Respaldando..."
                        self.cmbBxFabricantes.Enabled = False
                        self.chkLstBxComandos.Enabled = False
                        self.DeshabilitarBotonesFormulario()
                        self.HabilitarControlesToolStrip()
                        self.tlStripStLblMensajes.Text = 'Respaldando equipo 1 de ' + str(len(lstEquiposSeleccionados))

                        if (self.bckgrWkrEstado.IsBusy != True):
                                self.bckgrWkrEstado.RunWorkerAsync([lstEquiposSeleccionados, lstComandosSeleccionados])

                elif(equiposSeleccionados == False and comandosSeleccionados == True):
                        MessageBox.Show('No se ha seleccionado ningún equipo.','Seleccionar Equipos', MessageBoxButtons.OK, Messagel

                elif(equiposSeleccionados == True and comandosSeleccionados == False):
                        MessageBox.Show('No se ha seleccionado ningún comando.','Seleccionar Comandos', MessageBoxButtons.OK, Messa
                else:
                        MessageBox.Show('No se ha seleccionado ningún equipo y/o comando.','Seleccionar Equipos/Comandos', MessageB
```

Código 2.3. Método ObtenerRespaldos del Prototipo.

En el Código 2.4 se presenta una parte del método **GenerarInformes.** Este método se ejecutará cuando el técnico haga *click* en el botón “Generar Informes” y permitirá que el prototipo generé documentos de informe técnico para cada equipo a partir del documento base. En este fragmento de código se observan las variables usadas para implementar el método, la ejecución de un nuevo hilo para generar los documentos de informe técnico y el control de errores del método. Para observar todo el código del método observe el **Anexo II.**

```
def GenerarInformes(self, sender, e):
                lstDirectorios = []
                lstDirectorios = ScriptBase.ListarDirectoriosRespaldos()
                if((len(lstDirectorios) != 0) and ScriptBase.VerificarPlantillaInforme()):
                        self.dataGridViewEquipos.ReadOnly = True
                        self.cmbBxFabricantes.Enabled = False
                        self.chkLstBxComandos.Enabled = False
                        self.DeshabilitarBotonesFormulario()
                        self.HabilitarControlesToolStrip()
                        self.tlStripStLblMensajes.Text = 'Generando informe técnico equipo 1 de ' + str(len(lstDirectorios))

                        if (self.bckgrWkrGenerarInformes.IsBusy != True):
                                self.bckgrWkrGenerarInformes.RunWorkerAsync(lstDirectorios)

                elif(ScriptBase.VerificarPlantillaInforme() == False):
                        MessageBox.Show('No se encuentra el archivo "PLANTILLA_INFORME.docx" en el directorio '+'"'+os.getcw
                else:
                        MessageBox.Show('No se encuentran los directorios necesarios para generar los informes. Revise y/o a
```

Código 2.4. Método GenerarInformes del Prototipo.

En el Código 2.5 se presenta una parte del método **VerRespaldos.** Este método se ejecutará cuando el técnico haga *click* en el botón “Ver Respaldos” y permitirá que el prototipo despliegue el segundo formulario del prototipo (Figura 2.9) donde se visualizará los archivos de respaldo de configuración de los equipos y el contenido de los mismos. En este fragmento de código se observa la codificación para desplegar el segundo formulario del prototipo. Para observar todo el código del método observe el **Anexo II.**

```
1  def VerRespaldos(self, sender, e):
2                  self.formularioRespaldosConfiguracion.ShowDialog()
```

Código 2.5 Método VerRespaldos del Prototipo.

En el Código 2.6 se presenta una parte del método **VerInformes.** Este método se ejecutará cuando el técnico haga *click* en el botón “Ver Informes” y permitirá que el prototipo despliegue el tercer formulario del prototipo (Figura 2.10) donde se visualizará los documentos de informe técnico generado para cada equipo y el contenido de los mismos. En este fragmento de código se observa la codificación para desplegar el tercer formulario del prototipo. Para observar todo el código del método observe el **Anexo II.**

```
1  def VerInformes(self, sender, e):
2                  self.formularioInformesTecnicos.ShowDialog()
```

Código 2.6. Método VerInformes del Prototipo.

En la Figura 2.9. se muestra el segundo formulario final implementado:

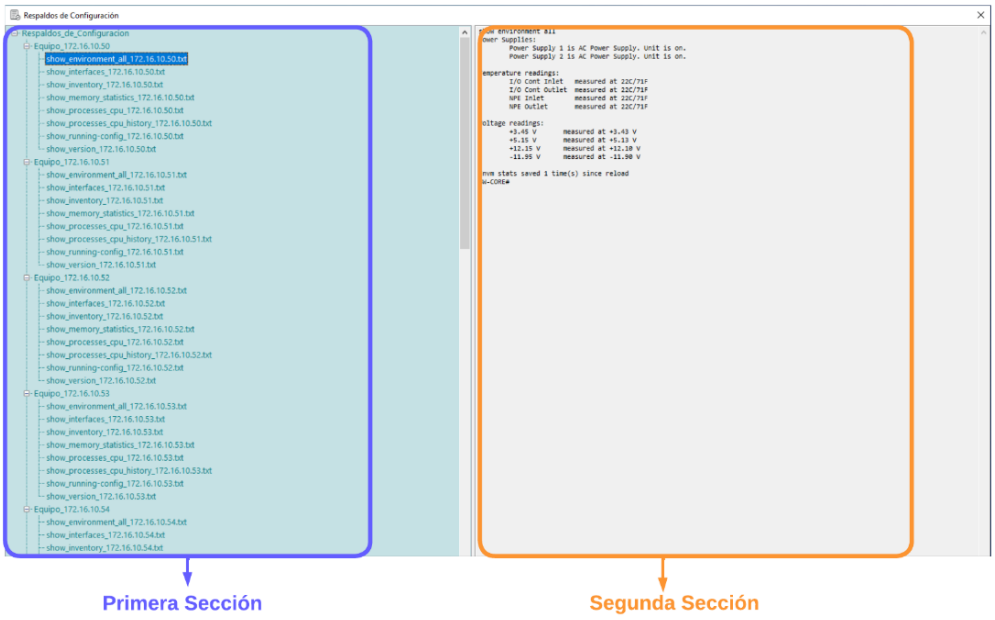

Figura 2.9. Segundo Formulario del Prototipo.

En la Figura 2.10. se muestra el tercer formulario final implementado:

Figura 2.10. Tercer Formulario del Prototipo.

2.3 OPERACIONES

Durante esta sección se mostrarán los detalles de las pruebas realizadas en el prototipo y los resultados obtenidos tras finalizar la etapa de Implementación, verificando los requerimientos planteados en la Etapa de Gobernanza en el **Apartado 2.1.2** y **Apartado 2.1.3.**

En el **Apartado 2.3.1** se actualizará el tablero ALM, en el **Apartado 2.3.2** se definirá el entorno de pruebas para evaluar el funcionamiento del prototipo; en el **Apartado 2.3.3** y en el **Apartado 2.3.4,** se validarán los Requerimientos Funcionales y los Requerimientos No Funcionales respectivamente. Finalmente, en el **Apartado 2.3.5** una vez finalizadas las pruebas de funcionamiento, se cerrará el tablero ALM.

2.3.1 ACTUALIZACIÓN DEL TABLERO ALM

Se ha actualizado el tablero ALM, mostrando las tareas que ya han sido completadas satisfactoriamente y las tareas que están en proceso. En la Figura 2.11 se observa la actualización del tablero ALM en la Etapa de Operaciones.

Figura 2.11. Actualización del Tablero ALM en la Etapa de Operaciones.

2.3.2 DEFINICIÓN DEL ENTORNO DE PRUEBAS

Para comprobar el funcionamiento del prototipo, se ha creado un entorno de pruebas controlado; por tema de confidencialidad, usando datos de un cliente de la empresa LIKATELEC. Este entorno de pruebas se ha realizado a través de una simulación de red como se muestra en la Figura 2.12.

La simulación de red está conformada por equipos CISCO de la siguiente manera: 1 equipo de núcleo, 1 equipo de distribución y 10 equipos de acceso. Los equipos han sido configurados para permitir la conectividad entre los equipos y el prototipo.

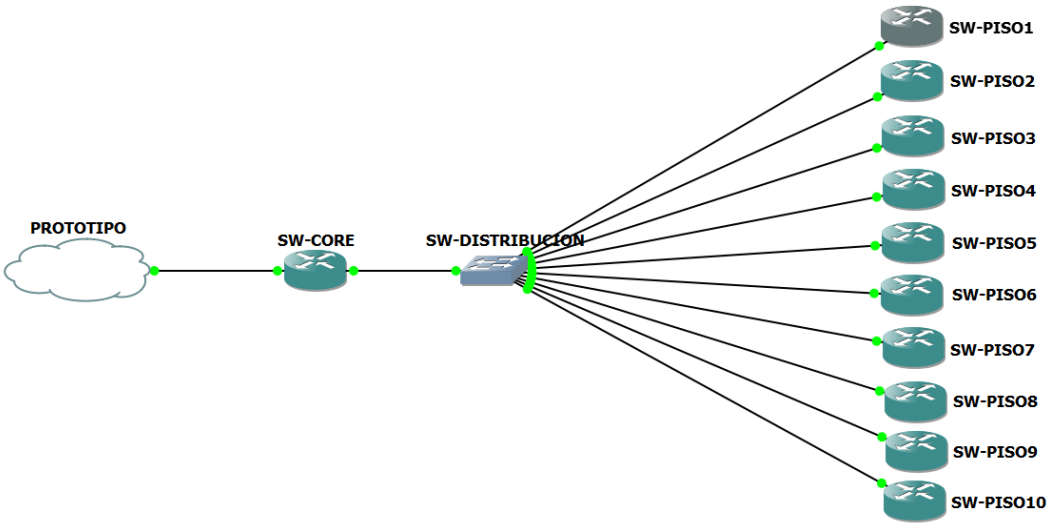

Figura 2.12. Simulación de red cliente LIKATELEC.

La empresa LIKATELEC está conformada por cuatro técnicos especialistas en varias áreas tecnológicas, como: Datacenter, Colaboración, Wireless, Seguridad, etc.; dada la predisposición de le empresa, las pruebas se realizaron con la totalidad de los participantes. Previo a la utilización del prototipo, cada técnico ha recibido una capacitación de una hora acerca del uso del mismo. Posteriormente, cada técnico ha utilizado el prototipo por 4 ocasiones permitiendo evaluar tanto los Requerimientos Funcionales como los Requerimientos No Funcionales. Finalmente, luego de que cada técnico ha usado el prototipo se ha realizado unas preguntas para obtener una retroalimentación que permite corregir posibles errores y/o mejorar alguna funcionalidad del prototipo.

2.3.3 VALIDACIÓN DE REQUERIMIENTOS FUNCIONALES

Las validaciones de los Requerimientos Funcionales se han realizado tomando como referencia la retroalimentación brindada por los técnicos una vez finalizada la ejecución de las pruebas del prototipo.

Tabla 2.11. Validación de Requerimientos Funcionales.

RF	Descripción	Estado		Observación
RF01	Configurar Parámetros de Conexión al CLI	Primera y Segunda Reunión	✓	Los técnicos manifestaron conformidad al momento de configurar los 3 parámetros de conexión al CLI de los equipos dado que permite ingresar los parámetros

				de varios equipos de manera sencilla.
RF02	Configurar Comandos para Respaldo de Configuración	Primera y Segunda Reunión		Los técnicos manifestaron conformidad al momento de configurar los comandos a ejecutar en el CLI de los equipos dado que permite ingresar varios comandos de manera sencilla.
RF03	Crear Archivos de Respaldo de Configuración	Primera y Segunda Reunión		Los técnicos manifestaron conformidad al momento de crear archivos de respaldo de configuración dado que la salida de cada comando para cada equipo se almacena en un archivo independiente lo cual facilita su búsqueda y posterior entrega al cliente.
RF04	Ver Respaldos de Configuración	Primera y Segunda Reunión		Los técnicos manifestaron conformidad al momento de visualizar los archivos y el contenido de los archivos de respaldo de configuración dado que el visualizador despliega la información de una manera legible y fácil de usar.
RF05	Crear Informes Técnicos	Primera y Segunda Reunión		Los técnicos manifestaron conformidad al momento de crear los documentos de informe técnico dado que el prototipo genera los mismos en un período corto de tiempo.
RF06	Ver Informes Técnicos	Primera y Segunda Reunión		Los técnicos manifestaron conformidad al momento de visualizar los documentos y el contenido de los documentos de informe técnico dado que el visualizador despliega la información de una manera legible y fácil de usar.

2.3.4 VALIDACIÓN DE REQUERIMIENTOS NO FUNCIONALES

Las validaciones de los Requerimientos No Funcionales se han realizado tomando como referencia la retroalimentación brindada por los técnicos una vez finalizada la ejecución de las pruebas del prototipo.

Tabla 2.13. Validación de Requerimientos No Funcionales.

RNF	Descripción	Estado		Observación
RNF01	Usabilidad del Prototipo	Primera y Segunda Reunión		Los técnicos manifestaron conformidad con respecto al diseño y usabilidad del prototipo dado que la información desplegada en el mismo es clara.
RNF02	Ejecución Lineal del Prototipo	Primera y Segunda Reunión		Los técnicos manifestaron claridad al momento de utilizar el prototipo dado que los pasos numerados los ayuda a tener una clara idea del orden de ejecución de las tareas.
RNF03	Compatibilidad del Prototipo	Primera y Segunda Reunión		Los técnicos manifestaron satisfacción dado que la totalidad de técnicos de prueba utilizan sistema operativo Windows y este prototipo fue desarrollado para este sistema.
RNF04	Ingreso de datos al Prototipo	Primera y Segunda Reunión		Los técnicos manifestaron conformidad dado que el archivo externo es claro y sencillo para ingresar los datos (parámetros de conexión y comandos) que el prototipo necesita para su funcionamiento.

2.3.5 CIERRE DEL TABLERO ALM

Luego de finalizar las pruebas en el prototipo, se ha procedido a cerrar el tablero ALM como se observa en la Figura 2.12, debido a que han culminado todas las tareas planteadas para las diferentes etapas de la metodología.

Figura 2.12. Finalización del Tablero ALM.

3 CONCLUSIONES Y RECOMENDACIONES

En el presente Capítulo se definirán las conclusiones obtenidas del prototipo y las recomendaciones en base a eventualidades al momento del desarrollo del prototipo.

3.1 CONCLUSIONES

El prototipo fue desarrollado en base a tres etapas: Gobernanza, Desarrollo y Operaciones y a partir de este proceso realizado se desprenden las siguientes conclusiones:

- Se ha logrado implementar un prototipo funcional que facilita la automatización de tareas de mantenimiento de equipos de red utilizando *Python Scripting* en la empresa LIKATELEC.
- El estudio de *Python Scripting* y su aplicación en la programabilidad de redes ha permitido identificar los procesos manuales y repetitivos que pueden ser optimizados en una organización y/o empresa dentro de este ámbito tecnológico (redes de información), trayendo como beneficio una reducción significativa en la inversión de tiempo y esfuerzo que al Administrador de Red emplea para realizar dichos procesos.
- El estudio de la metodología *Application Lifecycle Management* (ALM), a través de las etapas de Gobernanza, Desarrollo y Operaciones, ha permitido establecer los lineamientos adecuados para implementar el prototipo, de tal manera que el mismo cumpla con altos estándares de calidad al mismo tiempo que cumpla con todos los requerimientos establecidos por la empresa LIKATELEC.
- La utilización de los Diagramas de Clases y Diagrama de Actividades ha permitido diseñar el prototipo, orientando a la automatización de tareas de tal manera que, cumpla con diferentes reglas que aseguran el correcto funcionamiento del mismo.
- El diseño utilizado para la Capa de Presentación en la implementación del prototipo, ha permitido establecer en base a la primera toma de requerimientos, el orden de ejecución de las tareas que los técnicos de la empresa LIKATELEC deben realizar al momento de utilizar el prototipo, de tal manera que se cometan la menor cantidad de errores cuando se interactúa con el mismo.
- A través de la utilización del prototipo por parte de los técnicos de la empresa LIKATELEC sobre una simulación de red, se ha evaluado las funcionalidades

planteadas para el mismo, las cuales han permitido optimizar el tiempo que los técnicos dedican para realizar las tareas involucradas en un mantenimiento preventivo de equipos de red, cumpliendo de manera satisfactoria con las expectativas y requerimientos establecidos por la empresa.

3.2 RECOMENDACIONES

En base a la experiencia adquirida durante el desarrollo del prototipo se presentan recomendaciones enfocadas a desarrollos futuros sobre este prototipo o de proyectos similares:

- Para el desarrollo de aplicaciones con *Python* se recomienda el uso de *Visual Studio Code,* ya que es un editor de código fuente gratuito, el cual está disponible para sistemas operativos Windows, macOS y Linux. Este entorno de desarrollo brinda facilidades para escribir, depurar y probar código en tiempo real, así como para instalar paquetes externos que permiten añadir funcionalidades al entorno de desarrollo.
- Para el desarrollo de aplicaciones de escritorio .NET utilizando *Python* se recomienda el uso de Python.NET (pythonnet). Python.NET es un paquete que permite a los desarrolladores crear aplicaciones, utilizando servicios y componentes .NET, de tal manera que se pueda crear interfaces gráficas para aplicaciones Python sin la necesidad de escribir código.
- El presente Trabajo de Integración Curricular contempla el desarrollo una aplicación de escritorio para la automatización de tareas en un mantenimiento preventivo de equipos de red CISCO. En futuros trabajos se puede añadir mejoras como, por ejemplo: el soporte multifabricante, lo cual permitiría que en la aplicación se pueda seleccionar el fabricante de los equipos y acorde a esta selección se procesen los comandos ya que cada fabricante maneja un conjunto de comandos diferente.

4 REFERENCIAS BIBLIOGRÁFICA

[1] J. Edelman, S. S. Lowe, y M. Oswalt, *Network Programmability and Automation: Skills for the Next-Generation Network Engineer*. O'Reilly Media, Inc., 2018.

[2] "What Is Application Lifecycle Management (ALM)?", *Software Quality*. https://www.techtarget.com/searchsoftwarequality/definition/application-lifecycle-management-ALM (consultado el 25 de enero de 2023).

[3] "redes-por-software-SDN.pdf". Consultado: el 25 de enero de 2023. [En línea]. Disponible en: https://informatica.ucm.es/data/cont/media/www/pag-103596/transparencias/redes-por-software-SDN.pdf

[4] A. M. Mazin, R. A. Rahman, M. Kassim, y A. R. Mahmud, "Performance Analysis on Network Automation Interaction with Network Devices Using Python", en *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, abr. 2021, pp. 360–366. doi: 10.1109/ISCAIE51753.2021.9431823.

[5] C. Gil y A. Manuel, "Automatización de redes informáticas con Python".

[6] E. Burns, "What is Cisco?", *Tech Monitor*, el 9 de enero de 2017. https://techmonitor.ai/what-is/what-is-cisco (consultado el 25 de enero de 2023).

[7] "Using the Command-Line Interface", *Cisco*. https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3850/software/release/3se/consolidated_guide/b_consolidated_3850_3se_cg_chapter_01.html (consultado el 25 de enero de 2023).

[8] shmcarth, "Administración del ciclo de vida de la aplicación (ALM) con Microsoft Power Platform - Power Platform", el 9 de diciembre de 2022. https://learn.microsoft.com/es-es/power-platform/alm/overview-alm (consultado el 25 de enero de 2023).

[9] "¿Qué es ALM de gestión del ciclo de vida de aplicaciones? Definición | Las mejores herramientas | Guía Completa - Soluciones Visure", *Visure Solutions*. http://visuresolutions.com/es/blog/alm/ (consultado el 25 de enero de 2023).

[10] K. Admin, "What are ALM and Integrated ALM? - White Paper - Kovair", el 5 de junio de 2015. https://www.kovair.com/white-papers/what-are-alm-and-integrated-alm/ (consultado el 25 de enero de 2023).

[11] D. G. Cardacci, "Arquitectura de software académica para la comprensión del desarrollo de software en capas", Serie Documentos de Trabajo, Working Paper 574, 2015. Consultado: el 25 de enero de 2023. [En línea]. Disponible en: https://www.econstor.eu/handle/10419/130825

[12] S. Satav, S. Satpathy, y K. Satao, "A Comparative Study and Critical Analysis of Various Integrated Development Environments of C, C++, and Java Languages for Optimum Development", vol. 1, pp. 9–15, ene. 2011.

[13] "Stack Overflow Developer Survey 2022", *Stack Overflow*. https://survey.stackoverflow.co/2022/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2022 (consultado el 25 de enero de 2023).

[14] "Documentation for Visual Studio Code". https://code.visualstudio.com/docs (consultado el 25 de enero de 2023).

[15] j-martens, "Documentación de Visual Studio". https://learn.microsoft.com/es-es/visualstudio/windows/ (consultado el 25 de enero de 2023).

[16] TerryGLee, "Información general sobre Visual Studio", el 28 de octubre de 2022. https://learn.microsoft.com/es-es/visualstudio/get-started/visual-studio-ide (consultado el 25 de enero de 2023).

[17] "Python_para_todos.pdf". Consultado: el 25 de enero de 2023. [En línea]. Disponible en: https://repositorio.uci.cu/jspui/bitstream/123456789/10206/1/Python_para_todos.pdf

[18] "181531232001.pdf". Consultado: el 25 de enero de 2023. [En línea]. Disponible en: https://www.redalyc.org/pdf/1815/181531232001.pdf

[19] T. D. P. Manager, “10 Best ALM Tools For 2023”, *The Digital Project Manager*. https://thedigitalprojectmanager.com/tools/alm-tools/ (consultado el 25 de enero de 2023).
[20] “66612870011.pdf”. Consultado: el 25 de enero de 2023. [En línea]. Disponible en: https://www.redalyc.org/pdf/666/66612870011.pdf
[21] “IP Addressing and Subnetting for New Users”, *Cisco*. https://www.cisco.com/c/es_mx/support/docs/ip/routing-information-protocol-rip/13788-3.html (consultado el 25 de enero de 2023).
[22] C. Ávila Garzón, “Requerimientos NO funcionales”, *Dep. Educ. Virtual*, dic. 2019, Consultado: el 25 de enero de 2023. [En línea]. Disponible en: https://repositorio.konradlorenz.edu.co/handle/001/1527
[23] “CATEGORÍAS DE RESPUESTA EN ESCALAS TIPO LIKERT | Psicothema”, Consultado: el 25 de enero de 2023. [En línea]. Disponible en: https://reunido.uniovi.es/index.php/PST/article/view/7489
[24] A. Faulkner, “Lucidchart for Easy Workflow Mapping”, *Ser. Rev.*, vol. 44, núm. 2, pp. 157–162, abr. 2018, doi: 10.1080/00987913.2018.1472468.
[25] J. Cueto y C. Bertolotti, "Diagrama de clases en UML". [En línea]. Disponible en: *http://es. scribd. com/doc/31096724/Diagrama-de-Clases-en-UML# scribd*.
[26] X. F. Grau y M. I. S. Segura, “Desarrollo Orientado a Objetos con UML”, [En línea]. Disponible en: https://www.uv.mx/personal/maymendez/files/2011/05/umlTotal.pdf

5 ANEXOS

Los anexos que han sido adjuntados como parte complementaria al desarrollo del presente Trabajo de Integración Curricular se observan a continuación:

ANEXO I. Encuestas.

ANEXO II. Ejecutable y Código del Prototipo.