



Universidad Nacional de la Patagonia San Juan Bosco
Facultad de Ingeniería
Licenciatura en Informática
Programación Orientada a Objetos

Instancia Supervisada de Formación Práctica Profesional

Alumnos:

Mariezcurrena Ivo, Crespo Lautaro, Vidal Oropeza Fabian

Cátedra:

Solá Leiva Alejandro, Pollicelli Debora, Samec Gustavo, Mazzanti Renato

Fecha:

Lunes 11/11/2024

INDICE

SECCIÓN 1

ALCANCE.....	2
1.1. Introducción y objetivos.....	2
1.2. Requerimientos del sistema.....	3
1.3. Herramientas utilizadas.....	4
1.4. Entregables.....	4

SECCIÓN 2

MANUAL DE DESARROLLO.....	4
2.1. Estructura del proyecto.....	5
2.2. Capa Modelo.....	6
2.3. Capa Negocio.....	7
2.4. Capa Controlador.....	8
2.5. Capa Servicio.....	8
2.6. Capa DAO.....	11
2.7. Capa Conexion.....	13
2.8. Capa GUI.....	14
2.10. Funcionamiento.....	15
2.11. Javadoc.....	16
2.12. Posibles mejoras.....	16
2.13. Conclusiones.....	16

SECCIÓN 3

MANUAL DE USUARIO.....	18
3.1. Ejecutar.....	18
3.2. Inicio de sesión.....	18
3.3. Uso.....	20
3.4. Parametrización.....	20
3.5. Configuraciones.....	21

ALCANCE

1.1. Introducción y objetivos

El presente proyecto está enfocado en el desarrollo de software para la gestión remota de una red de equipos dentro de una institución. Está dirigido al personal de mantenimiento y administración de redes como una herramienta intuitiva y amigable con el usuario.

Para estructurar el proyecto, lograr su mantenibilidad y eficiencia, se propone:

- Uso del paradigma de Programación Orientada a Objetos
- Patrones de diseño de software: tales como MVC, DAO, Factory, Facade y Singleton
- Uso de bases de datos y archivos de texto como fuentes de datos
- Uso de estructuras de datos: grafos, mapas, listas, hastables
- Modos simulación y real
- Uso de hilos en procesos de consultas remotas (modo real)
- Desarrollo en capas: el proyecto está segmentado en capas, donde cada capa resuelve una funcionalidad del software

1.2. Requerimientos del sistema

En detalle, los requerimientos funcionales del sistema son los siguientes:

- Interfaz fácil de usar, con un diseño amigable
- Visualización de la red
- Obtención de información de los componentes con sus características y estados a través de consultas
- Permitir la modificación de la red, eliminando, actualizando o añadiendo equipos
- Posibilidad de alternar entre fuentes de datos (bases de datos o archivos de texto)
- Modos simulación y real
- Autenticación de usuarios mediante cuentas e inicio de sesión
- Soporte de idiomas inglés y español

Requerimientos no funcionales

- Rendimiento: se esperan las respuestas en periodos de tiempo determinados
- Escalabilidad: lograr la flexibilidad ante futuras ampliaciones
- Mantenibilidad: documentación y uso de prácticas como los patrones de diseño
- Seguridad: autenticación por correo electrónico
- Compatibilidad: Java es multiplataforma, con una instalación de OpenJDK-17 o superior en el sistema operativo, el equipo puede hacer uso del software

1.3. Herramientas utilizadas

Se utilizan las siguiente herramientas y bibliotecas, ya que determinamos que son las más adecuadas para la aplicación.

- Java [OpenJDK-17](#) (java versión 17)
- Gestor de dependencias [Maven 3.6.3](#) o superior
- [JGraphT 1.5.3](#) para disponer de grafos y sus algoritmos
- [mxGraph 4.2.2](#) para visualizar grafos
- [Javax mail 1.0](#) para el servicio de correo electrónico
- [Postgresql 42.2.17](#) driver para Java
- [JUnit 5](#) para tests unitarios
- [Git 2.34.1](#) o superior para el control de versiones

1.4. Entregables

En la presente entrega, se incluye el [código fuente](#) del proyecto, con los archivos de datos, configuraciones, tests y bibliotecas; [documentación del código](#) generada con Javadoc; y [manuales](#) de desarrollo y usuario en las siguientes secciones de este documento, que incluyen [diagramas UML](#) de clases y secuencia.

SECCIÓN 2

MANUAL DE DESARROLLO

El siguiente manual incluye la documentación necesaria para entender cómo funciona la aplicación, cómo está construida y cómo funciona una vez puesta en ejecución.

2.1. Estructura del proyecto

HERRAMIENTAS DE DESARROLLO

El código fuente se encuentra en el paquete `src`, allí existe un paquete `test` donde se encuentran tests escritos en `JUnit5` relacionados a la lógica de cálculo, y un paquete `main` en el que está la aplicación principal de la red.

Utilizamos `Git` para agilizar el desarrollo en equipo y control de versiones, y `Maven` para gestionar dependencias (de ahí el uso del archivo `pom.xml` que nombra las dependencias necesarias), y dichas dependencias se almacenan en el paquete `lib`.

En el paquete `db` se encuentran las queries con las que se puede configurar una nueva base de datos para la aplicación. El gestor de base de datos que empleamos es `PostgreSQL` para almacenar todos de los usuarios y la red.

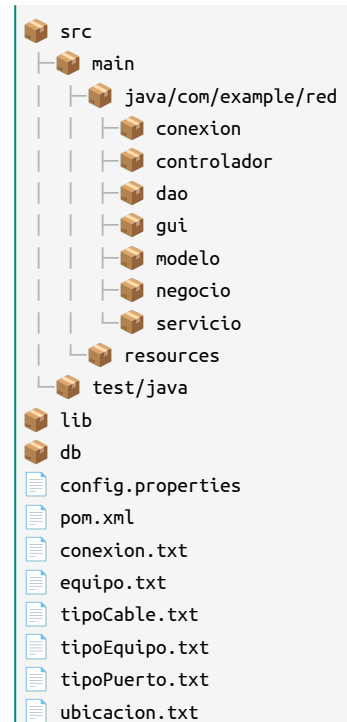
Como se mencionó en los requerimientos, está la posibilidad de cargar datos desde `archivos txt`, que están en la carpeta raíz del proyecto.

Finalmente, el archivo de configuraciones `config.properties` almacena las configuraciones principales del usuario como el idioma y modo de la aplicación.

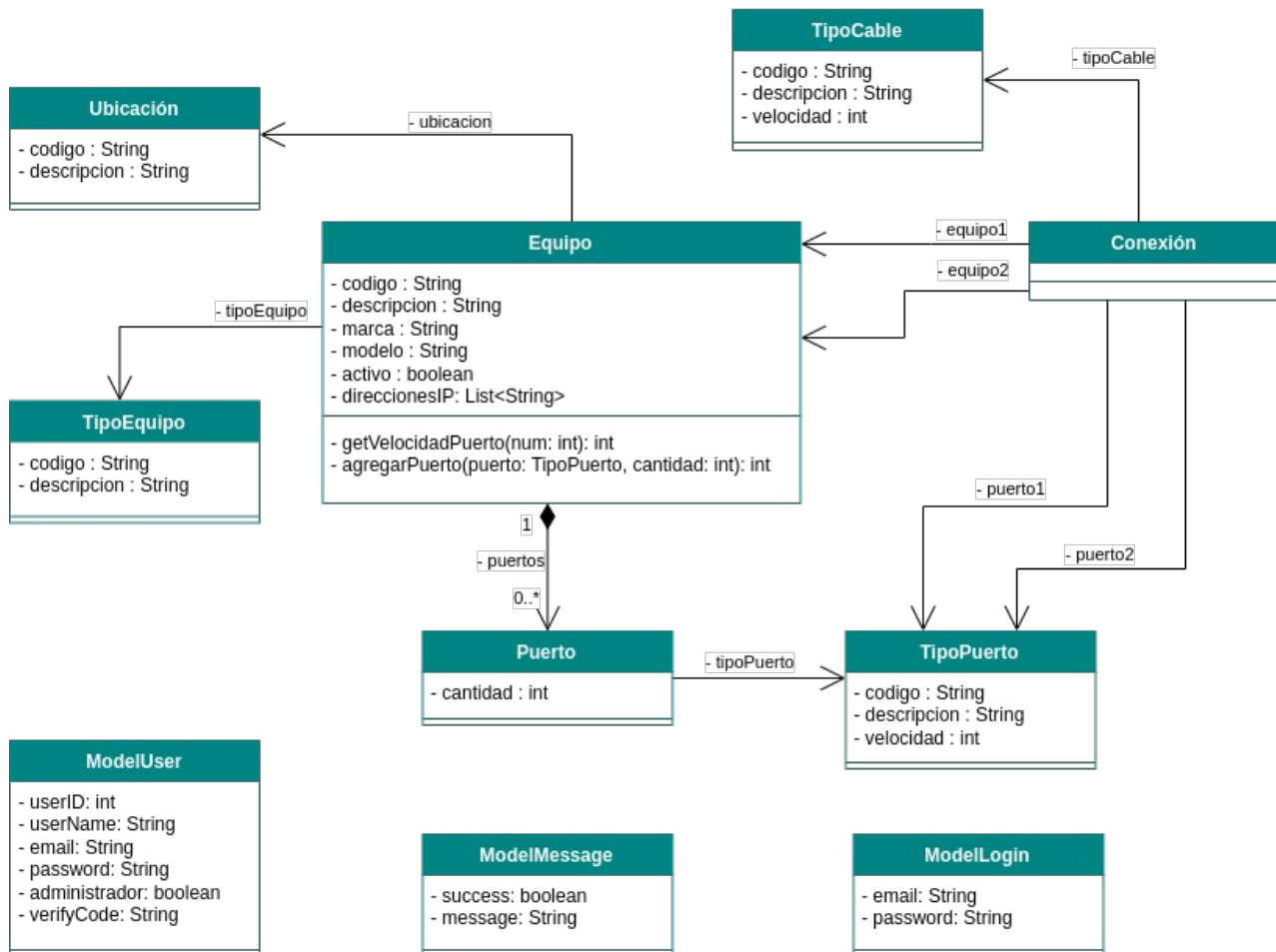
PATRÓN DE DISEÑO MVC

Se decide desarrollar la aplicación en capas, es decir, separar en conjuntos adyacentes de componentes para separar las responsabilidades en 3 componentes principales: modelo (`modelo`, `negocio`, `conexión`, `servicio` y `dao`), vista (`gui`) y controlador (`controlador`).

A continuación se detallan los conceptos que cada capa encapsula.



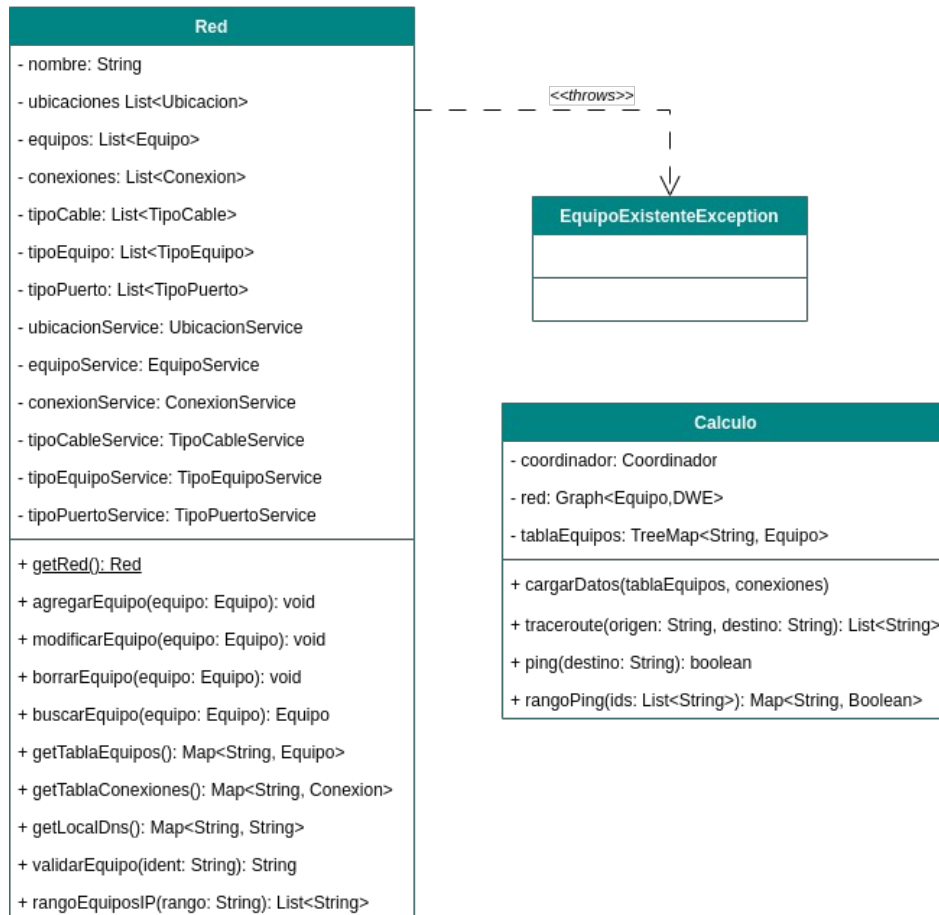
2.2. Capa Modelo



Esta capa contiene las clases que modelan los objetos de la vida real que se involucran en la red, tales son los equipos, tipos de equipos, tipos de puertos de los equipos, ubicaciones en las que se encuentran, conexiones y tipos de cables de las conexiones.

También contiene los modelos para inicio de sesión del usuario, para mostrar mensajes y para verificación.

2.3. Capa Negocio



PATRÓN DE DISEÑO SINGLETON

La clase **Red** además de ser el modelo de la red con todos sus componentes, mantiene comunicación con la fuente de datos (archivos de texto o bases de datos) para cargarlos o realizar actualizaciones a través de las clases Service de cada componente. Debido a que entonces es necesario que su instancia deba ser única y de acceso global, se utiliza el patrón **Singleton**, con un método estático getRed() para instanciar una única vez.

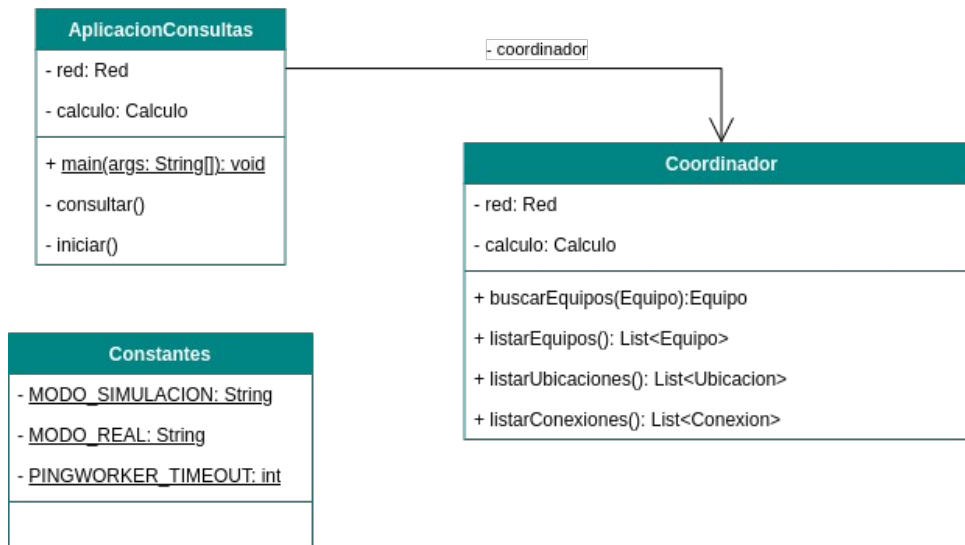
Cuando la red detecte la inserción de un equipo repetido, se lanza la excepción **EquipoExistenteException**.

LIBRERIA JGRAPH T

La clase **Calculo** se encarga de llevar un grafo especial, con todos los equipos y conexiones de la red, para algoritmos de cálculo. Dicho grafo es proveído por la librería **JGraphT** que ofrece clases de todo tipo y sus algoritmos disponibles.

En la aplicación lo utilizamos para calcular el camino más corto entre dos equipos, usando el algoritmo de **Dijkstra**.

2.4. Capa Controlador

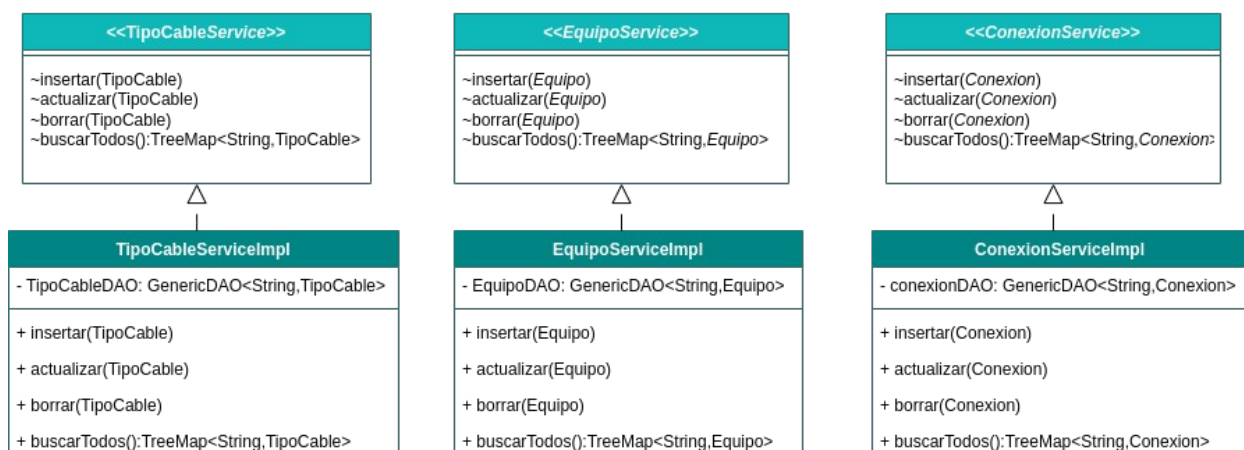


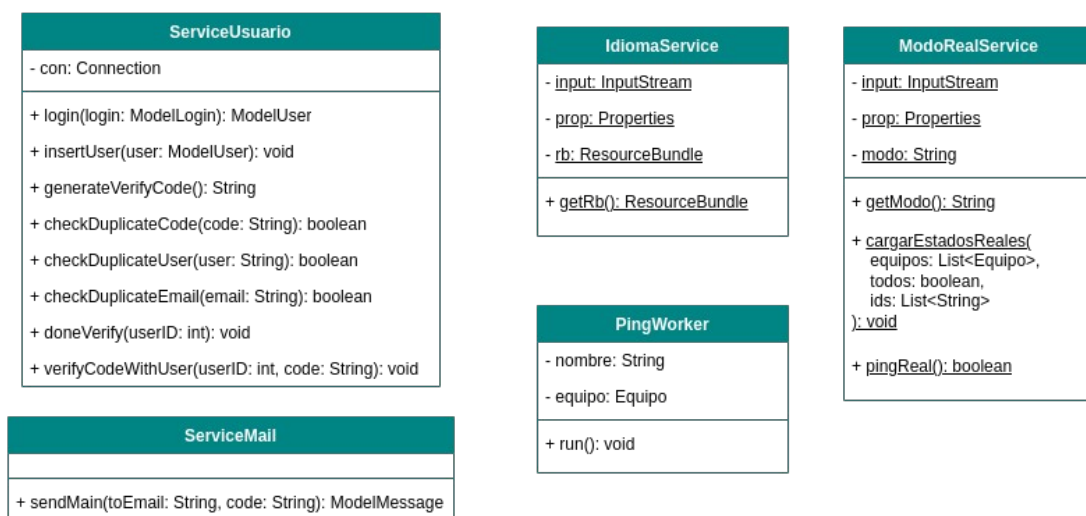
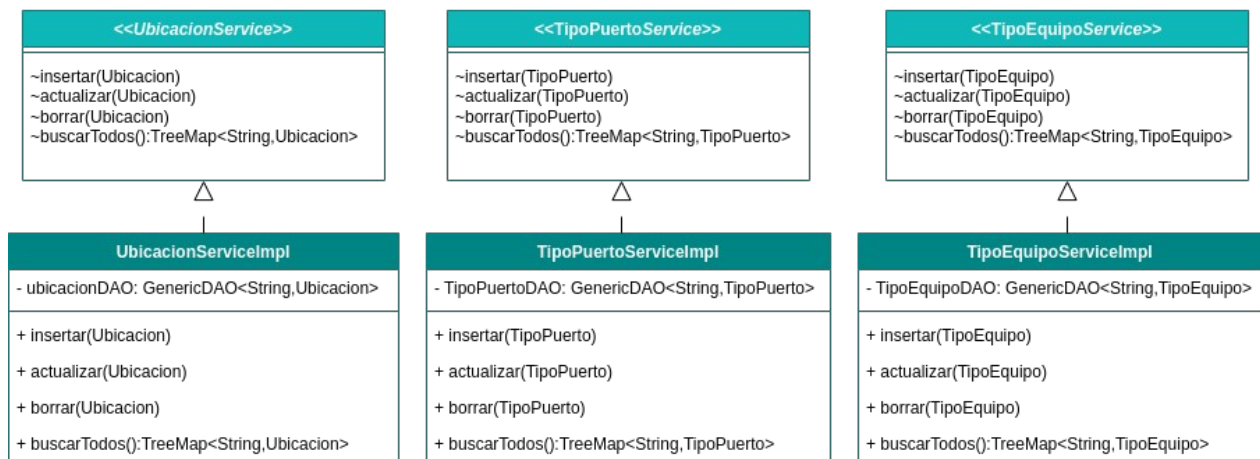
La clase **AplicacionConsultas** es la se debe ejecutar para iniciar la aplicación, contiene el método `main()`.

Coordinador es una clase auxiliar que posee las referencias de la única instancia de Red y la instancia de Cálculo.

Constantes lleva constantes definidas que se usan en varias partes del proyecto: el nombre del modo simulación y el modo real, y el tiempo de espera para consultas remotas en milisegundos.

2.5. Capa Servicio





PATRÓN DE DISEÑO FACADE

Esta capa provee interfaces de servicio simplificadas para la obtención y modificación de datos de componentes, servicios de email, idiomas y consultas remotas

Las interfaces [EquipoService](#), [ConexionService](#), [UbicacionService](#), [TipoCableService](#), [TipoPuertoService](#) y [TipoEquipoService](#) declaran los métodos para insertar(), actualizar(), borrar() y buscarTodos(). Sus implementaciones se encuentran en el mismo paquete. Estos hacen uso de los DAO para llamar a los métodos de acceso de datos, se explicará más adelante

La clase [ServiceUsuario](#) provee métodos para el registro e inicio de sesión de usuarios a la aplicación

[IdiomaService](#) se encarga de leer la configuración de idioma elegido en el archivo de configuraciones, para cargar el idioma correcto en un recurso ResourceBundle y poder usarlo en la interfaz gráfica

SUBPROCESAMIENTO MÚLTIPLE

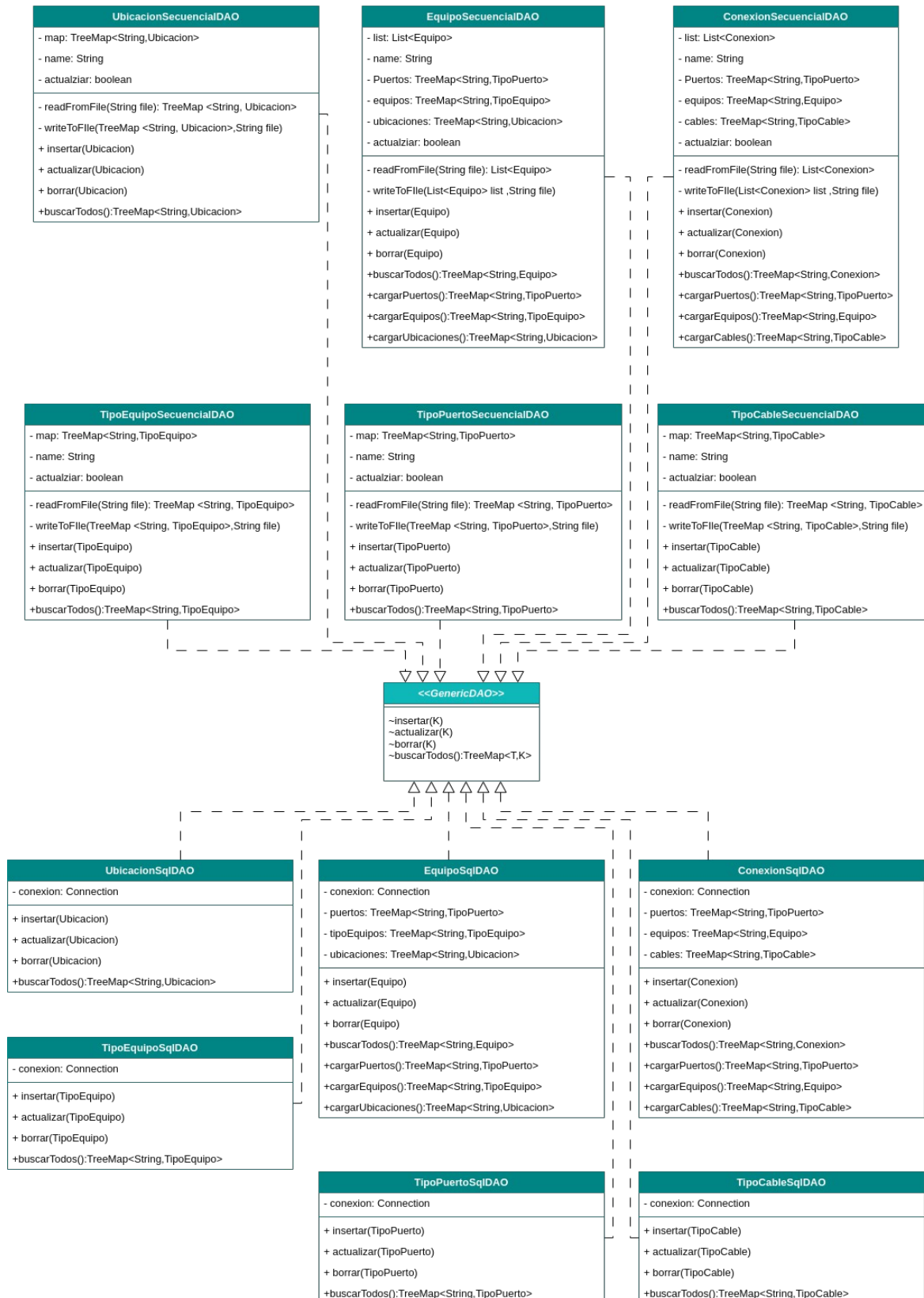
ModoRealService se encarga de la carga de la configuración del modo de uso, y en caso de que el usuario desee utilizar el modo real, también ofrece métodos de consulta remota. Dichas consultas requieren un tiempo de espera de respuesta, y puede que se desee hacer varias consultas a la vez, por lo que se instancian múltiples hilos **PingWorker** en un **ExecutorService** y el hilo principal se queda en espera de los resultados.

PingWorker implementa la interfaz **Runnable**, y se encarga de recibir un equipo para consultar su estado real (realiza un ping desde la terminal del sistema operativo), se queda en espera del resultado

LIBRERIA JAVAX.MAIL

La clase **ServiceMail** provee métodos para enviar emails a direcciones de correo. Esto se usa en el proceso de registro de una nueva cuenta

2.6. Capa DAO



PATRON DE DISEÑO DAO

Esta capa resuelve la lectura y manipulación de datos almacenados a través de objetos que proveen los métodos. En el proyecto se contempla como fuentes de datos posibles, archivos de texto o una base de datos relacional

La interfaz GenericDAO declara los métodos para la lectura y modificación de los datos, que luego se implementan según la fuente de los mismos.

- En caso de archivos de texto: [EquipoSecuencialDAO](#), [ConexionSecuencialDAO](#), [UbicacionSecuencialDAO](#), [TipoCableSecuencialDAO](#), [TipoEquipoSecuencialDAO](#) y [TipoPuertoSecuencialDAO](#)
- En caso de bases de datos relacionales: [EquipoSqlDAO](#), [ConexionSqlDAO](#), [UbicacionSqlDAO](#), [TipoCableSqlDAO](#), [TipoEquipoSqlDAO](#) y [TipoPuertoSqlDAO](#).

BASE DE DATOS RELACIONAL POSTGRESQL

Para la base de datos, optamos por el gestor PostgreSQL, usando un servidor cuyo host se encuentra en la universidad.

Para crear las tablas que modelan los componentes de la red que se desean guardar, se utilizó el siguiente diseño

usuarios	
PK	<u>id_usuario INT</u>
	nombre_usuario VARCHAR(255) NOT NULL
	email VARCHAR(255)
	contraseña VARCHAR(255) NOT NULL
	administrador boolean NOT NULL
	codigo_verificacion VARCHAR(200)
	estado VARCHAR(200)

ubicacion	
PK	<u>codigo VARCHAR(50)</u>
	descripcion VARCHAR(100) NOT NULL

direccionip	
PK	<u>id INT</u>
	equipo_codigo VARCHAR
	direccion_ip VARCHAR

equipo	
PK	<u>codigo VARCHAR</u>
	descripcion VARCHAR
	marca VARCHAR
	modelo VARCHAR
	tipoequipo_codigo VARCHAR
	ubicacion_codigo VARCHAR
	activo VARCHAR

conexion	
PK	<u>UniquelD</u>
	equipo1 VARCHAR(100)
	equipo2 VARCHAR(100)
	tipocable VARCHAR(100)
	tipopuerto1 VARCHAR(50)
	tipopuerto2 VARCHAR(50)

puerto	
PK	<u>id INT</u>
	equipo_codigo VARCHAR
	tipopuerto_id VARCHAR
	cantidad INT

tipoequipo	
PK	<u>codigo VARCHAR(100)</u>
	descripcion VARCHAR(100) NOT NULL

tipopuerto	
PK	<u>codigo VARCHAR(50)</u>
	descripcion VARCHAR(100) NOT NULL
	velocidad INT NOT NULL

tipocable	
PK	<u>codigo VARCHAR(100)</u>
	descripcion VARCHAR(100) NOT NULL
	velocidad INT NOT NULL

ARCHIVOS DE TEXTO

Formato del archivo conexión.txt

Equipo 1	Tipo puerto 1	Equipo 2	Tipo puerto 2	Tipo cable
SWAM	1G	AP09	1G	C5E
SWAM	1G	AP08	1G	C5E
...				

Formato del archivo equipo.txt

Cod.	Descrip.	Marca	Modelo	Tipo equipo	Ubi.	Tipo puerto	Puerto	Direcciones IP	Activo
AP01	APAulas1			AP	A01	1G	1	192.168.16.2	false
AP03	APAulas3			AP	A03	1G	1	192.168.16.4	true
...									

Formato del archivo tipoCable.txt

Código	Descripcion	Velocidad
C5	UTP Cat. 5	100
C5E	UTP Cat. 5e	1000
...		

Formato del archivo tipoPuerto.txt

Código	Descripcion	Velocidad
100M	100 Mbps	100
1G	1 Gbps	1000
...		

Formato del archivo tipoEquipo.txt

Código	Descripcion
AP	Access Point
COM	Computadora
...	

Formato del archivo ubicaciones.txt

Código	Descripcion
A01	Aula 1
A03	Aula 3
...	

2.7. Capa Conexion

ConexionDB
- <u>input: InputStream</u>
- <u>prop: Properties</u>
- <u>instance: ConexionDB</u>
- connection: Connection
+ connectToDatabase(): void

Factory
- <u>instancias: HashTable<String, Object></u>
+ getInstancia(objName: String): I

Esta capa se encarga de la conexion a la base de datos, en caso de necesitarlo

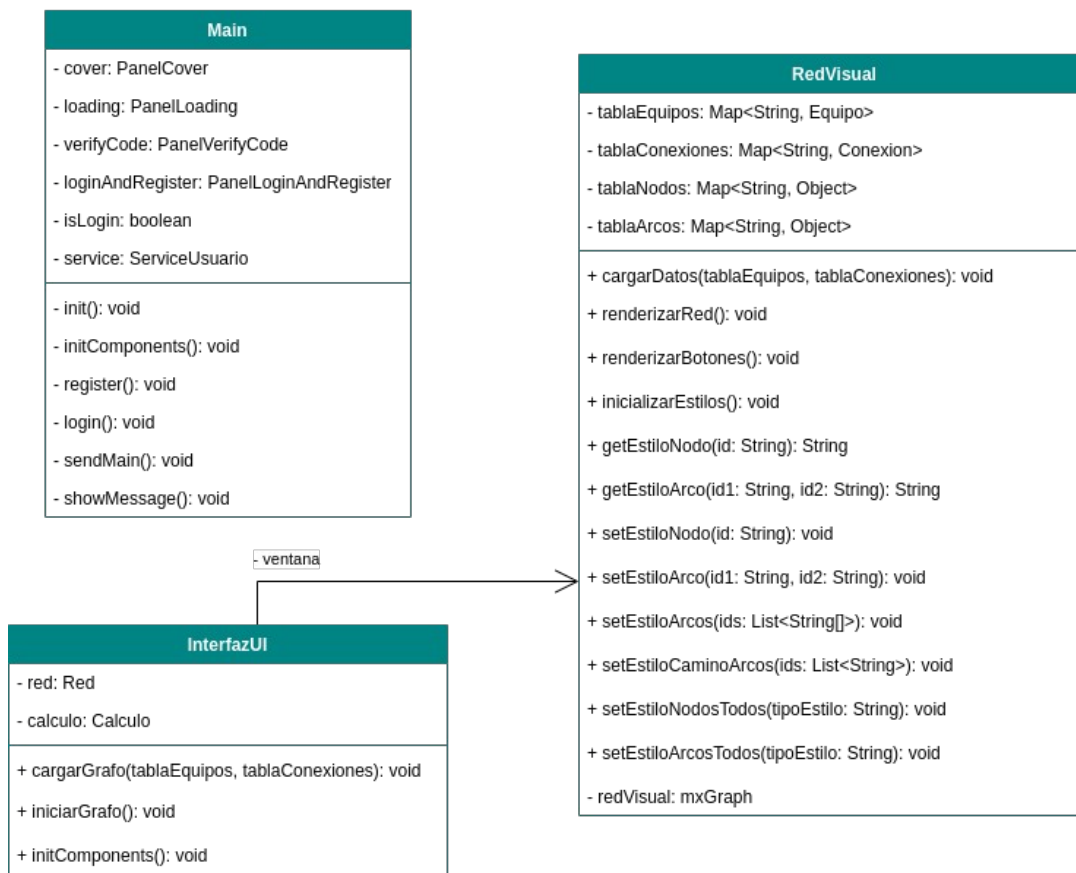
PATRÓN DE DISEÑO SINGLETON

La clase `ConexionDB` carga los parámetros de conexión desde el archivo de configuraciones y realiza la conexión a la base de datos. Solo puede existir una instancia de ella, y se obtiene mediante el método `getInstance()`

PATRÓN DE DISEÑO FACTORY

La clase `Factory` gestiona una tabla de los DAO. Si se le pide un objeto que no existe, intenta agregarlo usando un constructor sin parámetros

2.8. Capa GUI



Contiene las clases necesarias para mostrar una interfaz gráfica de usuario. Principalmente se utiliza Swing, junto con otras utilidades como `MigLayout` que facilita la distribución de los elementos visuales.

`RedVisual` muestra una ventana con la red de equipos y conexiones.

`InterfazUI` se encarga de la interfaz de la aplicación principal, el sistema para gestionar la red. Al instanciarlo hace la llamada a los métodos para obtener todos los datos de la red.

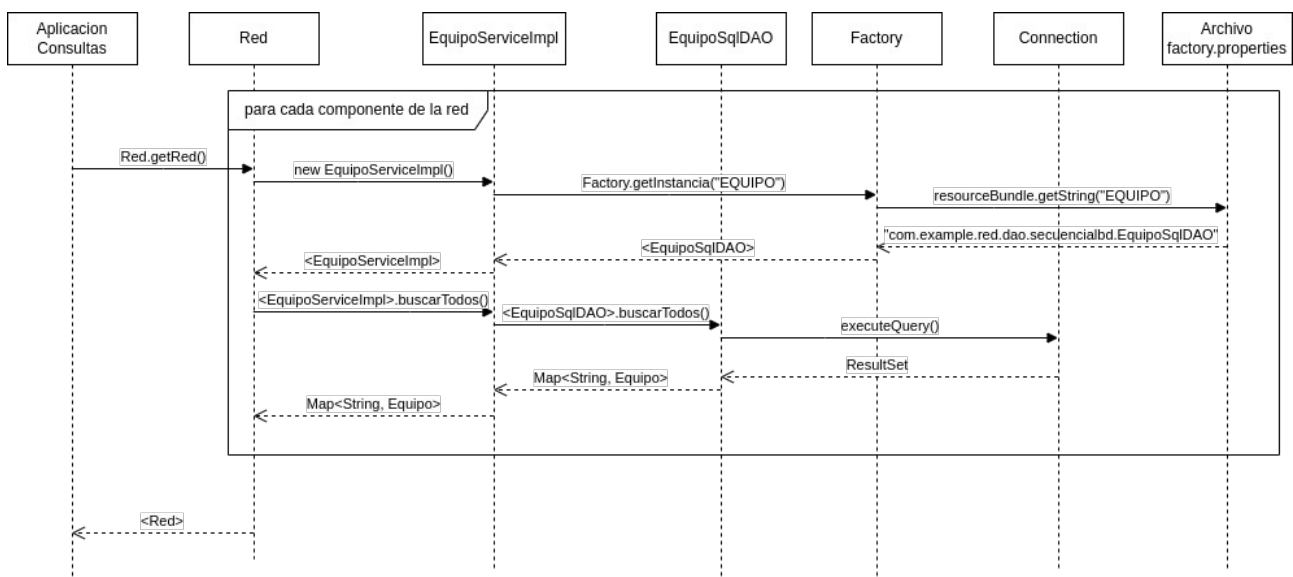
El resto de clases como [ConexionAdd](#), [ConexionList](#), [ConexionMod](#), etc. que se pueden encontrar en los subpaquetes, son formularios que se mostrarán cuando se desee listar, añadir o modificar algún elemento de la red

La clase [Main](#) se encarga de mostrar la ventana de inicio de sesión y registro de usuario. Obtiene los datos que introdujo el usuario y llama a los métodos que realizan las validaciones necesarias para registrarlo en la base de datos (usuario ya existente, email o nombre ya registrados, errores de conexión).

2.10. Funcionamiento

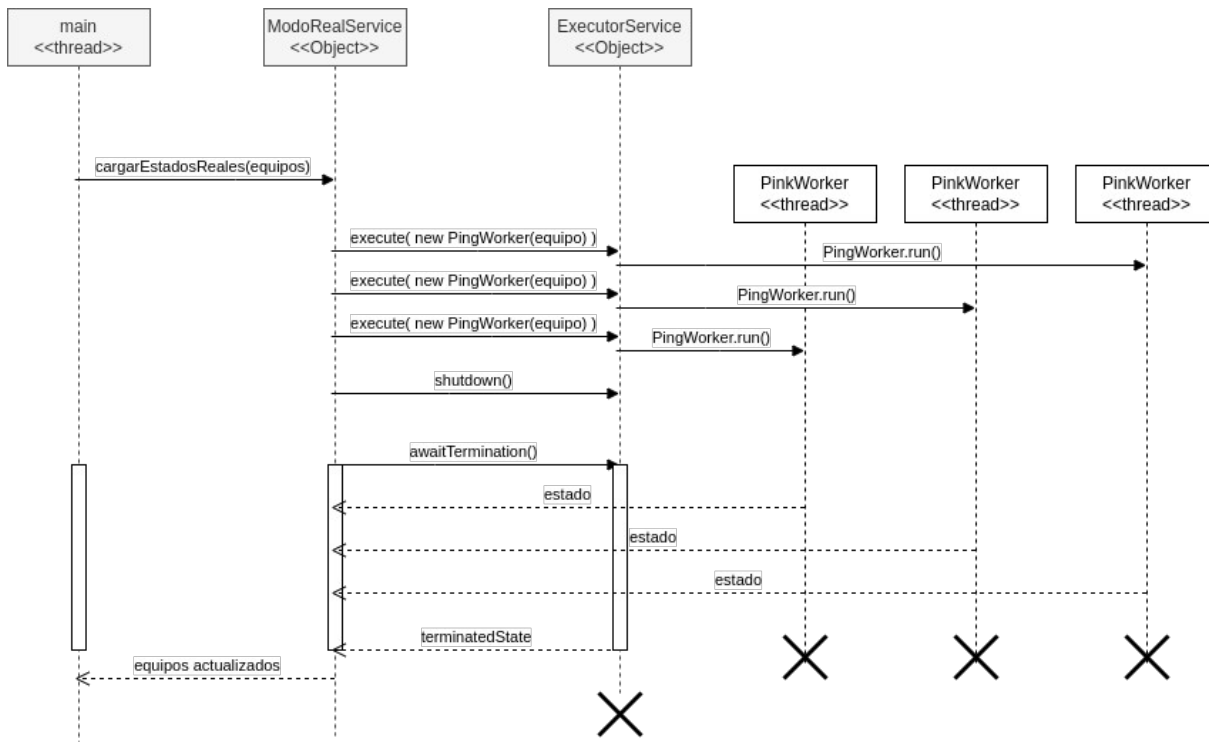
CARGA DE DATOS

Este diagrama de secuencia representa el orden de llamada de los métodos para cargar los datos de Equipos. El proceso encuadrado se repite para cada componente (conexiones, tipos de equipos, tipos de puertos, etc.)



CONSULTAS REMOTAS

Para las consultas remotas (en el modo real) utilizamos hilos en clases PingWorker, que son gestionados con ExecutorService. Cada hilo PingWorker se encarga de ejecutar el ping en la terminal del Sistema Operativo del usuario y esperar el resultado. Esto fue necesario para que se pudieran hacer múltiples ping a la vez y que la aplicación no se quede esperando las respuestas por demasiado tiempo.



2.11. Javadoc

Para generar la documentación utilizamos Javadoc, se puede abrir desde el archivo llamado “Index.html” en la carpeta “Javadoc” de esta entrega.

OVERVIEW
PACKAGE
CLASS
USE
TREE
INDEX
HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD
SEARCH:

Package com.example.red.servicio

Interface ConexionService

All Known Implementing Classes:
ConexionServiceImpl

```
public interface ConexionService
```

Interfaz que define las operaciones para gestionar conexiones Provee los métodos para insertar, leer, modificar y eliminar

Method Summary

All Methods
Instance Methods
Abstract Methods

Modifier and Type	Method	Description
void	actualizar(Conexion conexion)	Actualizar una conexión de la red
void	borrar(Conexion conexion)	Borrar una conexión de la red
TreeMap<String,Conexion>	buscarTodos()	Obtener todas las conexiones de la red
void	insertar(Conexion conexion)	Insertar una nueva conexión en la red

Method Details

insertar

2.12. Posibles mejoras

ROLES

Por cuestiones de tiempo y complejidad omitimos la implementación de roles de usuario con distintos permisos para la aplicación. De todas maneras la idea para incluirlo habría sido que exista el rol administrador con el permiso para editar y eliminar equipos, y un rol usuario o demo que permita revisarlos únicamente

2.13. Conclusiones

Este proyecto nos permitió no solo adquirir experiencia práctica en el desarrollo de aplicaciones complejas, sino también enfrentarnos a retos nuevos derivados de la escala y los requisitos específicos del sistema. Al resolver estos desafíos, aprendimos a aplicar conceptos avanzados, como patrones de diseño, que mejoraron la estructura y mantenibilidad del código. Además, integramos múltiples librerías y adoptamos herramientas nuevas, como gestores de dependencias y sistemas de control de versiones, que facilitaron la gestión del proyecto y el trabajo en equipo.

También destacamos la importancia de la documentación adecuada para facilitar la comprensión del código y la colaboración entre los miembros del equipo, lo que resultó esencial para el éxito del desarrollo. En general, este proyecto nos ayudó a expandir nuestras habilidades técnicas y a reforzar buenas prácticas en programación, dejándonos mejor preparados para abordar futuros desarrollos con una perspectiva más profesional y metodológica.

SECCIÓN 3

MANUAL DE USUARIO

En esta sección se explica cómo utilizar la aplicación, desde cómo ejecutarlo a cómo configurarlo.

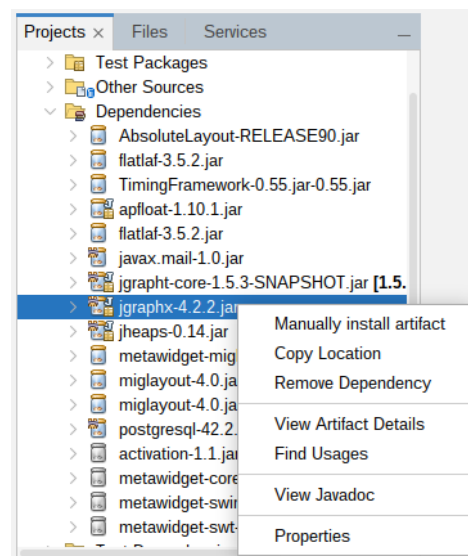
3.1. Ejecutar

Para ejecutar el proyecto será necesario hacerlo desde el [IDE Apache NetBeans](#)

Una vez abierto el proyecto, solo se debe abrir la carpeta "Dependencias" y para cada paquete "Instalar artifact manualmente"

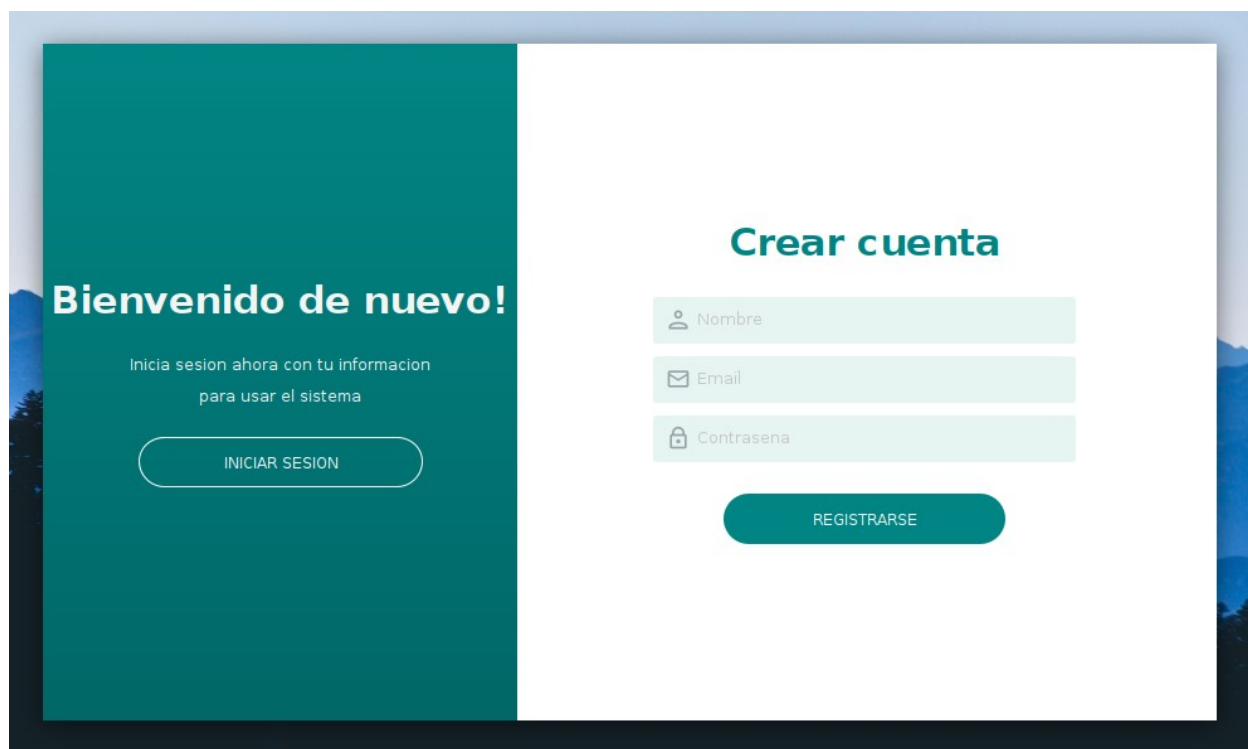
Con esto se instalarán algunas dependencias que el proyecto brinda, que no se encuentran para descargar desde repositorios confiables

Finalmente, la clase que se debe ejecutar es la clase AplicacionConsultas.java

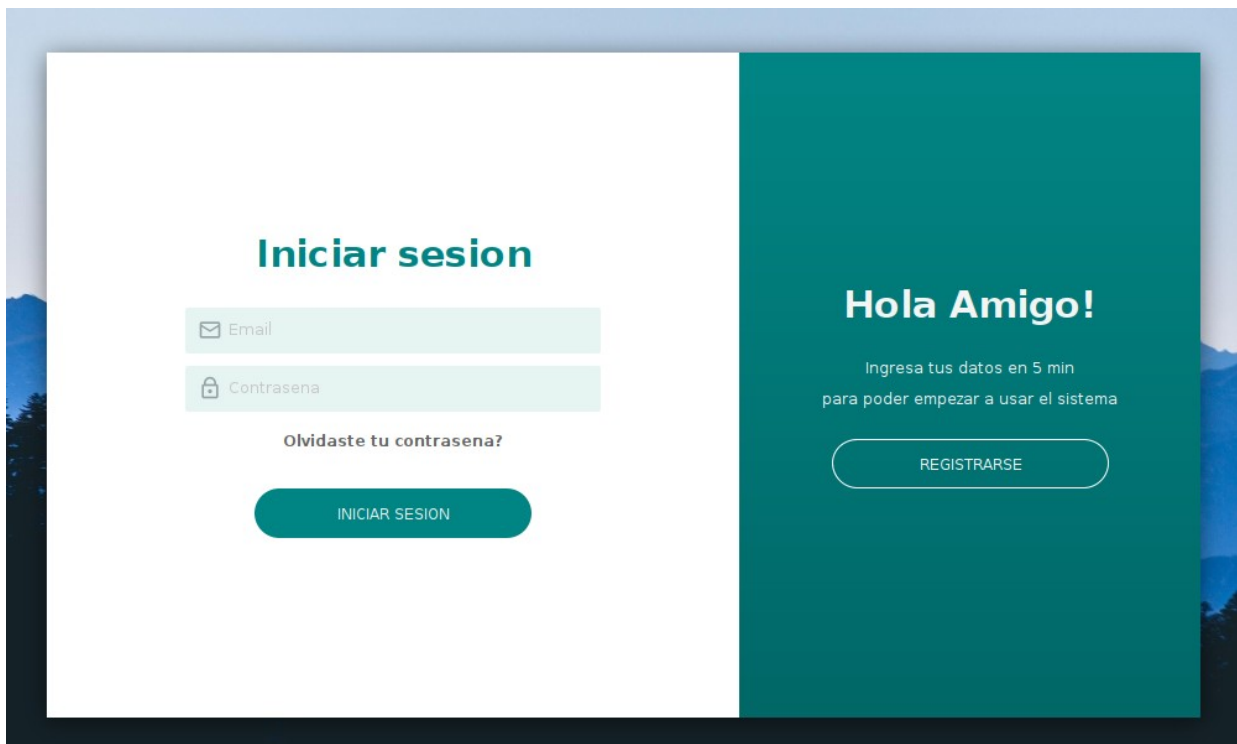


3.2. Inicio de sesión

Se abrirá una ventana para iniciar sesión o registrarse. Si es su primera vez será necesario ingresar sus datos y registrarse.

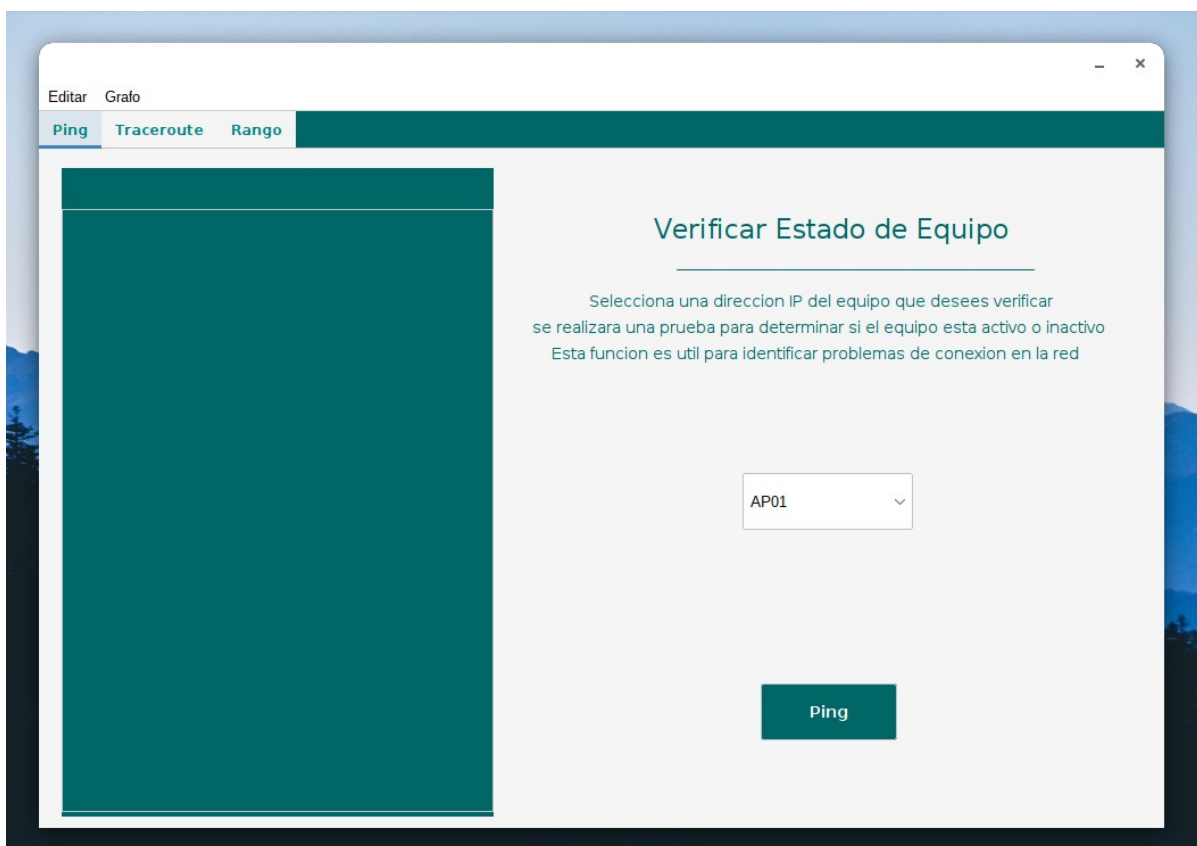


Si ya tiene una cuenta, presione el boton "Inicia sesion" y se mostrará un formulario para iniciar sesión



The image shows a login and registration interface. On the left, under the heading "Iniciar sesion", there are two input fields: "Email" with an envelope icon and "Contraseña" with a lock icon. Below these is a link "Olvidaste tu contraseña?". At the bottom of this section is a teal button labeled "INICIAR SESION". On the right, a teal sidebar contains the text "Hola Amigo!", followed by "Ingresa tus datos en 5 min para poder empezar a usar el sistema", and a white button with a teal border labeled "REGISTRARSE".

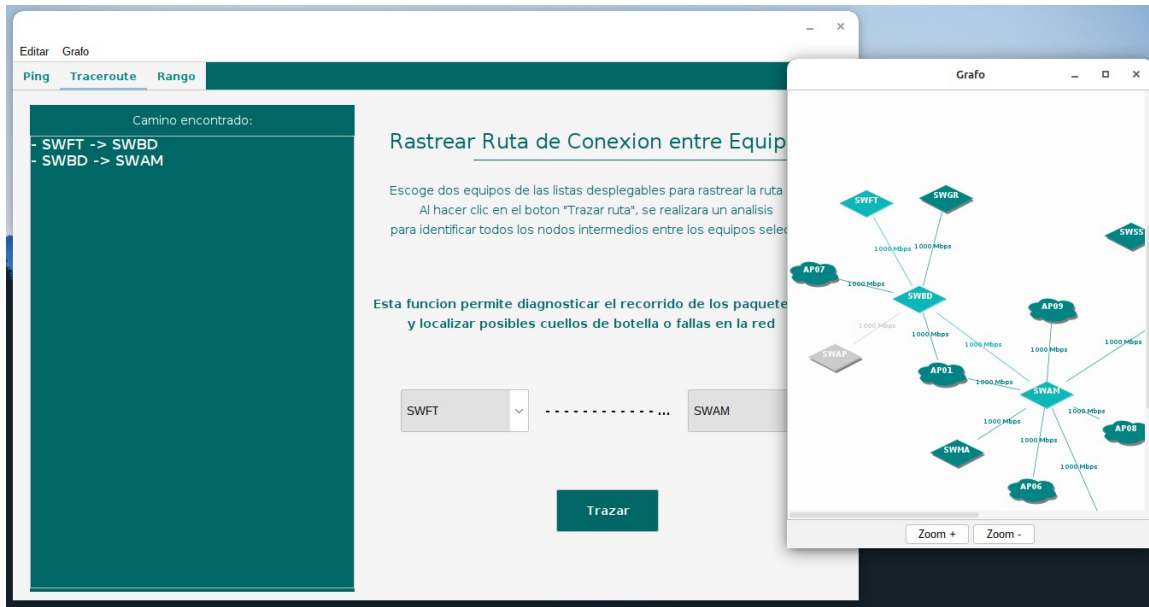
Una vez iniciado sesión podrá ver la interfaz principal para empezar a operar



The image is a screenshot of a web application window. The top bar has "Editar" and "Grafo" buttons. Below it is a navigation menu with "Ping", "Traceroute", and "Rango" tabs. The "Ping" tab is active. The main content area is titled "Verificar Estado de Equipo". It contains the text: "Selecciona una direccion IP del equipo que desees verificar se realizara una prueba para determinar si el equipo esta activo o inactivo Esta funcion es util para identificar problemas de conexion en la red". Below this text is a dropdown menu showing "AP01" with a downward arrow. At the bottom of the content area is a teal button labeled "Ping".

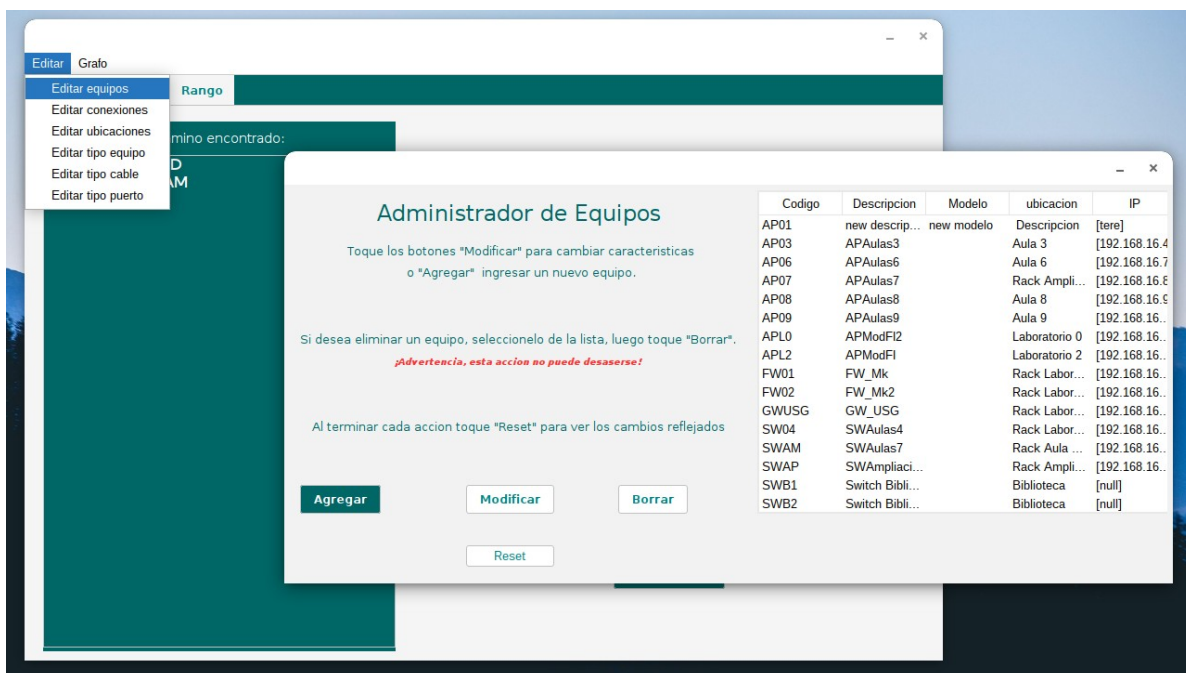
3.3. Uso

Con el botón “Grafo” de la barra de menú puede mostrar la opción para visualizar la red en otra ventana. En las pestañas “Ping”, “Traceroute” y “Rango” puede empezar a realizar operaciones, y los equipos involucrados se verán resaltados en la ventana de visualización de grafo



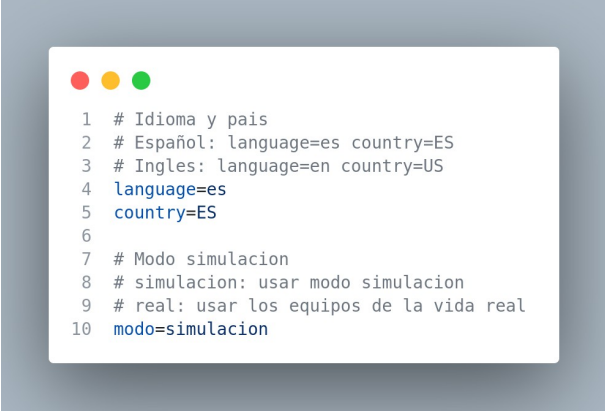
3.4. Parametrización

Con el botón “Editar” de la barra de menú puede desplegar las opciones para observar los detalles que cada elemento de la red y modificarlos, agregarlos o borrarlos. Por ejemplo, los equipos.



3.5. Configuraciones

Puede acceder a configuraciones como el idioma o el modo de uso (simulacion o real) desde el archivo `config.properties`. El mismo se encuentra en la carpeta raíz del proyecto.



```
1 # Idioma y país
2 # Español: language=es country=ES
3 # Inglés: language=en country=US
4 language=es
5 country=ES
6
7 # Modo simulacion
8 # simulacion: usar modo simulacion
9 # real: usar los equipos de la vida real
10 modo=simulacion
```