Proyecto: Gestión de Licencias de Docentes

Instituto Tecnológico de Informática (ITI)

Asignaturas: Programación, Base de Datos, Redes Informáticas, Análisis de Producción de Texto

Docentes a cargo: Marcela Mederos, Alejandra Domínguez, Ana Silva, Eduardo Mazzetti

Grupo: *Los Balatros*

Integrantes: Joaquín Giménez, Sebastien Alizo, Francisco González

Título del proyecto: *Asistrack*

Fecha de entrega: 12/09/2025

Software destinado: Los administrativos de la UTU y alumnos de la misma.

Índice:

Síntesis de la Tercera Entrega: Proyecto Asistrack

Definición del equipo y asignación de roles

Los roles fueron asignados en base a las fortalezas individuales de cada integrante. A través de una conversación grupal, discutimos nuestras habilidades, puntos fuertes y nuestros intereses, llegando a un acuerdo sobre la mejor distribución de tareas. Para ello aplicamos brevemente el análisis FODA (Fortalezas, Oportunidades, Debilidades y Amenazas).

- **Sebastien Alizo**: Líder del proyecto y programador secundario.

- **Francisco González**: Programador backend y desarrollador de la interfaz.

- **Joaquín Giménez**: Gestor de la base de datos y Documentador.

Reglamento interno del grupo

Como equipo, establecimos un conjunto de normas que nos servirán para organizarnos.principales normas:

- **Responsabilidad de los miembros**: Cada integrante se compromete a entregar sus tareas en tiempo y forma, asistir a las reuniones virtuales, participar activamente y comunicar cualquier dificultad o problema.

- **Mejora continua**: Se revisan los avances periódicamente para implementar correcciones, mejoras y retroalimentación entre compañeros.

- **Tiempos de entrega y revisión**: Se establecen fechas límite internas para tener márgenes de revisión antes de la entrega oficial.

- **Medios de comunicación**: Utilizamos **WhatsApp** como canal principal de comunicación grupal.
- **Herramientas digitales**: Para el almacenamiento y desarrollo colaborativo, usamos **Google Drive**.

- **Gestión de decisiones**: Todas las decisiones se toman mediante discusión grupal, priorizando el consenso y la escucha activa de cada opinión.

- **Simulador de red**: Para la parte de redes, se utilizó **Cisco Packet Tracer** como herramienta principal de simulación.

Nombre de la aplicación: Asistrack

Es un juego de palabras conformado por Asistencias y Track (Pista/Rastro en ingles) “Asis” hace referencia a las asistencias docentes, que son el núcleo de la información que gestiona el sistema.

“Track” alude a la capacidad de seguimiento, control y registro de esas asistencias de forma organizada y centralizada.

Definición de requisitos del sistema

● **Datos registrados**: Nombre, apellido, cédula, materia y grupo asignado de cada profesor, además de sus inasistencias.

● **Acceso a la información**:

○ **Administración**: lectura, escritura y ejecución.

○ **Profesores**: sólo lectura.

○ **Alumnos**: sólo lectura.

● **Información pública**: Nombre, apellido, grupos afectados, turnos (matutino, vespertino o nocturno), y fechas de inasistencia (desde/hasta).

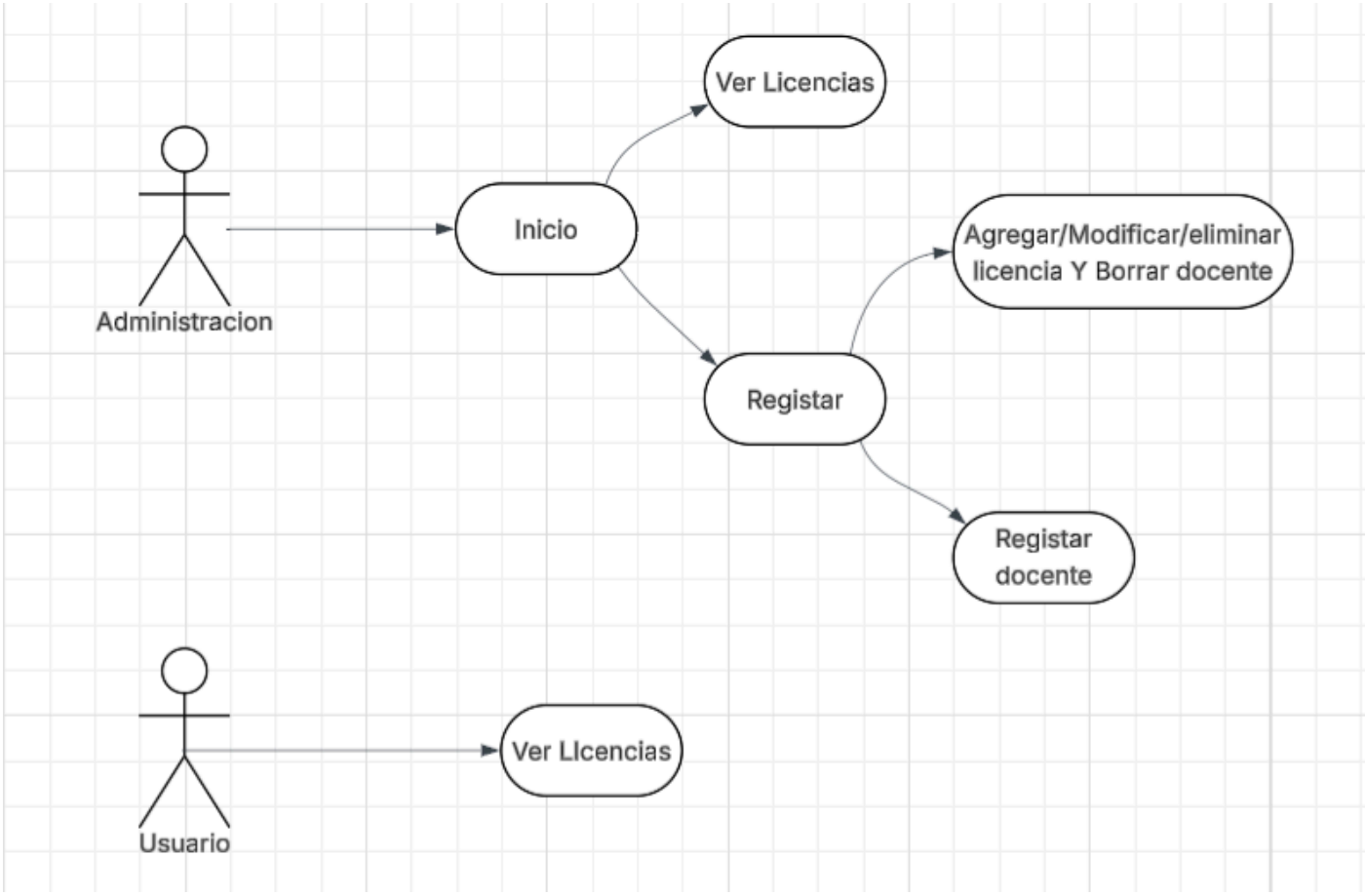
Casos de uso: Accesos según tipo de usuario

La siguiente imagen representa un diagrama de **casos de uso**, en el que se ilustran los usuarios de la aplicación (*Asistrack*) y las acciones que pueden realizar, según sus permisos definidos:

● **Administración:** Puede **leer**, **añadir** y **modificar** información relativa a las inasistencias.

● **Docentes:** Tienen permiso para **leer** los registros de inasistencia.

● **Alumnos:** Solo pueden **leer** las inasistencias registradas de los docentes, lo cual les permite conocer qué clases han sido afectadas.



6. Prototipo de interfaz en Figma

Como parte del desarrollo de *Asistrack*, se diseñó un **prototipo de interfaz gráfica** utilizando la herramienta **Figma**, con el objetivo de representar visualmente cómo funcionará la aplicación.

El prototipo incluye las siguientes secciones:

- **Pantalla principal**.
- **Formulario de registro de inasistencias**.
- **Vista de licencias activas**.

● **Panel de control** para adscripción.

Acceso al prototipo:

Figma

Redes Informáticas

El trabajo tiene como objetivo la creación y configuración de una red local simulada en Cisco Packet Tracer, en la que se integren los servicios de DHCP, DNS y Web, y se permita el acceso de los usuarios a una página web simulada mediante un nombre de dominio. Se utilizaron conocimientos de direccionamiento IP, configuración de servidores y diseño de topologías de red. Esta actividad se realizó en equipo, fomentando el trabajo colaborativo y el uso de herramientas de simulación.

Objetivo

● Configurar un servidor de base de datos en el packet tracer.

Descripción de la topología

La red fue diseñada con **topología en árbol**, en la cual todos los dispositivos finales (servidores) están conectados a un **switch central**, en el caso de las PCs estan conectadas a un **switch** que establece conexión con el **switch central**, el cual a su vez se conecta a un router.

Componentes utilizados:

● 1 Router (Cisco ISR 4331)

● 2 Switches (2960-24TT)

- 6 PCs (alumnos y docentes)
- 1 Servidor DHCP
- 1 Servidor DNS
- 1 Servidor Web

Configuración realizada

- **Router:** se configuró la interfaz FastEthernet con IP 192.168.1.1 como puerta de enlace.
- **Servidor DHCP:** asignado con una IP, configurado para entregar direcciones.
- **Servidor DNS:** se le asignó una IP fija y se configuró para traducir el nombre www.licencias.com a la IP del servidor web.
- **Servidor Web:** configurado con una página HTML simulando una base de datos de licencias docentes.
- **PCs alumnos y docentes:** configuradas en modo DHCP para recibir IP's dinámicamente.

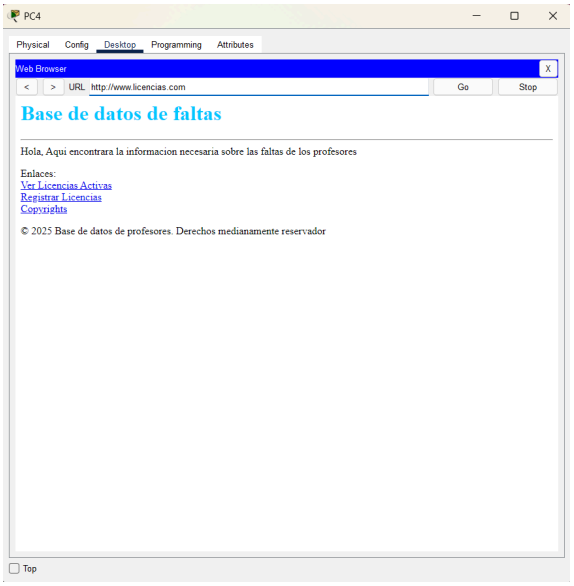
Resultados

La simulación fue exitosa:

- Los dispositivos recibieron direcciones IP automáticamente desde el servidor DHCP.
- Se accedió a la página web ingresando el nombre www.licencias.com, lo cual demuestra

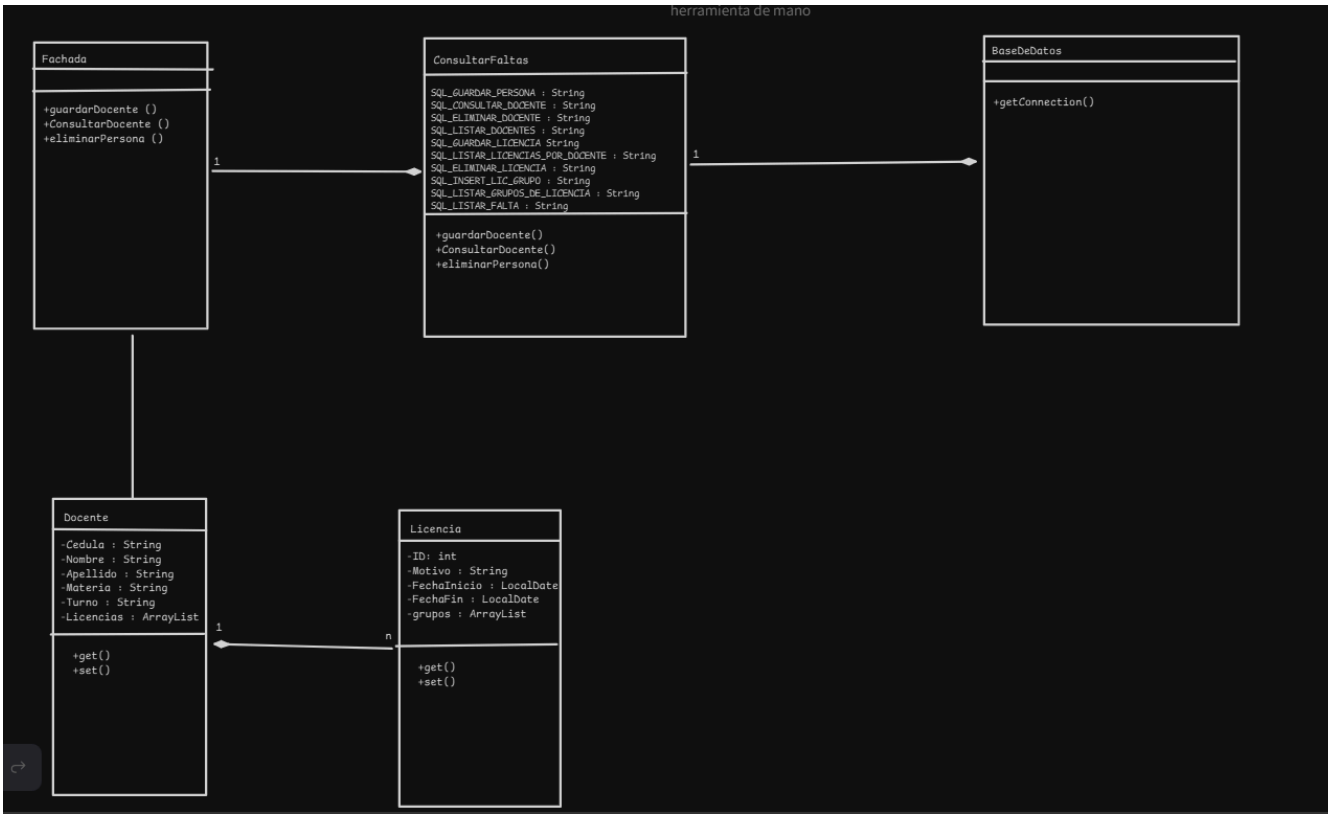
que el DNS funciona correctamente.

- El sitio web fue visible desde cualquier PC de la red que estuviera correctamente configurada.



Modelo de clases UML: estructura de datos

A continuación, se presenta el diagrama UML del sistema, el cual define la estructura de clases principales utilizadas en el desarrollo de la lógica del sistema.



Capa Lógica del sistema

Para desarrollar la lógica del sistema *Asistrack*, se implementaron cinco clases principales en Java

Clase Docente

Define cada docente registrado en el sistema. Esta clase contiene información personal y académica como:

1. Cédula
2. Nombre y apellido
3. Materia
4. Turno
5. Lista de licencias asociadas

```
5  package CapaLogica;
6
7  import java.util.ArrayList;
8  import java.util.List;
9
10 /**
11  *
12  * @author sebas
13  */
```

Clase Licencia

Modela las inasistencias justificadas de los docentes. Sus atributos principales incluyen:

1. Id (Autoincrement)
2. Motivo de la licencia
3. Fecha de inicio
4. Fecha de finalización
5. Lista de grupos

Esta clase permite registrar todas las inasistencias de un docente, especificando qué grupos se ven impactados por cada una.

```java
public class Licencia {
    private int id;                     // es autoincrement
    private String motivo;
    private LocalDate fechaInicio;
    private LocalDate fechaFin;
    private java.util.List<String> grupos = new java.util.ArrayList<>();

    public Licencia() {} //por si acaso uno vacio

    public Licencia(String motivo, LocalDate fechaInicio, LocalDate fechaFin, List<String> grupos) {
        this.motivo = motivo;
        this.fechaInicio = fechaInicio;
        this.fechaFin = fechaFin;
        setGrupos(grupos);
        validarFechas(); //es un metodo que esta creado por alla abajo
    }

    // Constructor completo (para leer desde la BD)
    public Licencia(int id, String motivo, LocalDate fechaInicio, LocalDate fechaFin, List<String> grupos) {
        this.id = id;
        this.motivo = motivo;
        this.fechaInicio = fechaInicio;
        this.fechaFin = fechaFin;
        setGrupos(grupos);
        validarFechas(); //es un metodo que esta creado por alla abajo
    }
    public Licencia(String motivo, LocalDate fechaInicio, LocalDate fechaFin, String gruposConcatenados) {
        this.motivo = motivo;
        this.fechaInicio = fechaInicio;
        this.fechaFin = fechaFin;
        setGruposDesdeCadena(gruposConcatenados);
        validarFechas();
    }

    // --- Getters/Setters ---
```

Clase Fachada

Permite conectar al Docente con la base de datos, en vez de que la interfaz gráfica interactúe con la base de datos. Principalmente tiene métodos para poder guardar, consultar y eliminar docente

```java
package CapaLogica;
import CapaPersistencia.ConsultasFaltas;
import CapaExcepcion.BDexcepcion;
import CapaExcepcion.FaltasExcepcion;
/**
 *
 * @author sebas
 */
public class fachada {
    public void guardarDocente (Docente docente) throws Exception{
        ConsultasFaltas per = new ConsultasFaltas();
        per.GuardarDocente(docente);
    }
    public Docente ConsultarDocente(String ci) throws FaltasExcepcion,BDexcepcion, Exception {
        Docente per = new Docente();
        ConsultasFaltas docente = new ConsultasFaltas();
        per=docente.ConsultarDocente(ci);
    return per;
    }
    public void eliminarPersona (String ci) throws FaltasExcepcion, Exception{
        ConsultasFaltas docente = new ConsultasFaltas();
        docente.eliminarDocente(ci);
    }
}
```

Parte Gráfica:

Contenidos:

1- Inicio

2- Licencias Por Docente

3- Registrar Docente

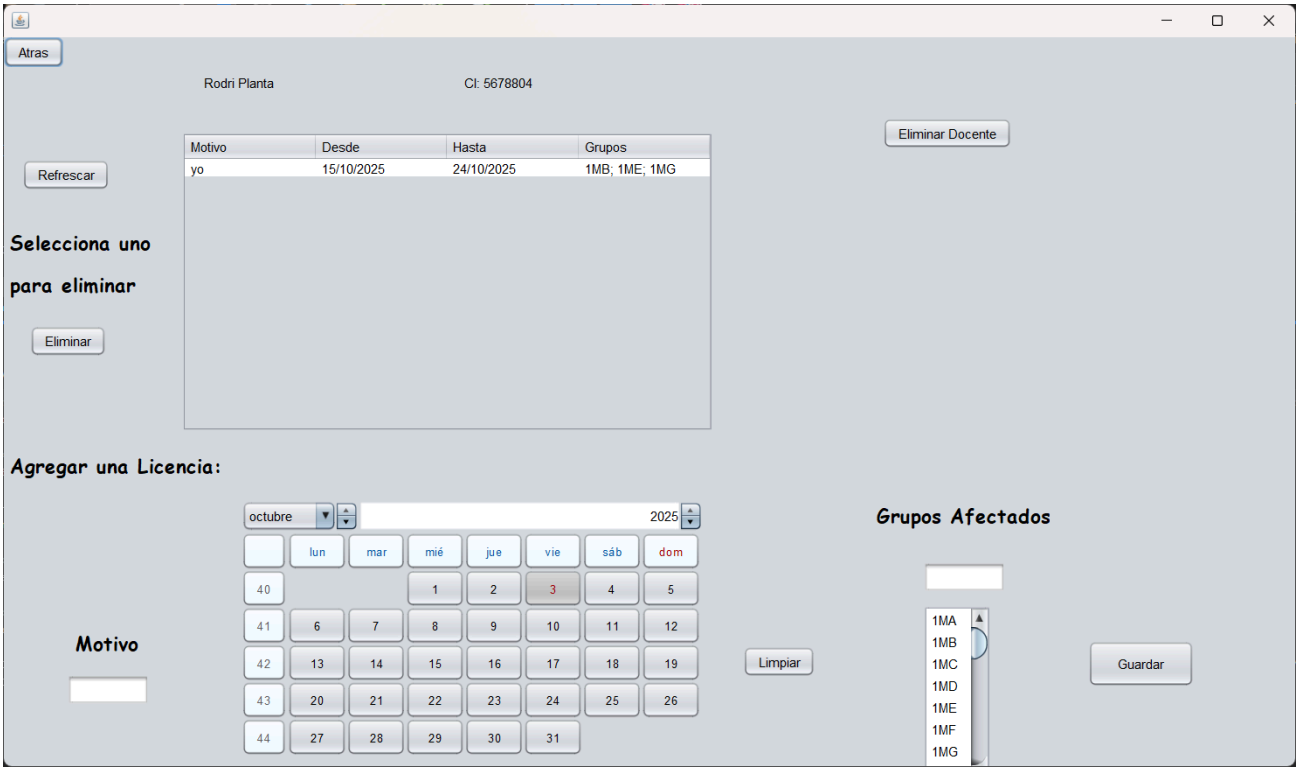
4- Selección

5- Tabla

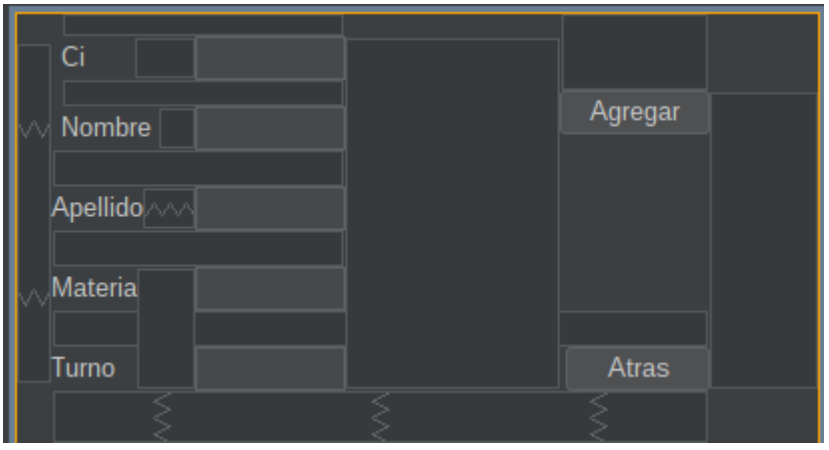
1-


Bienvenido al
Gestor de Faltas
Tabla de Faltas
Continuar
Registrar
Continuar

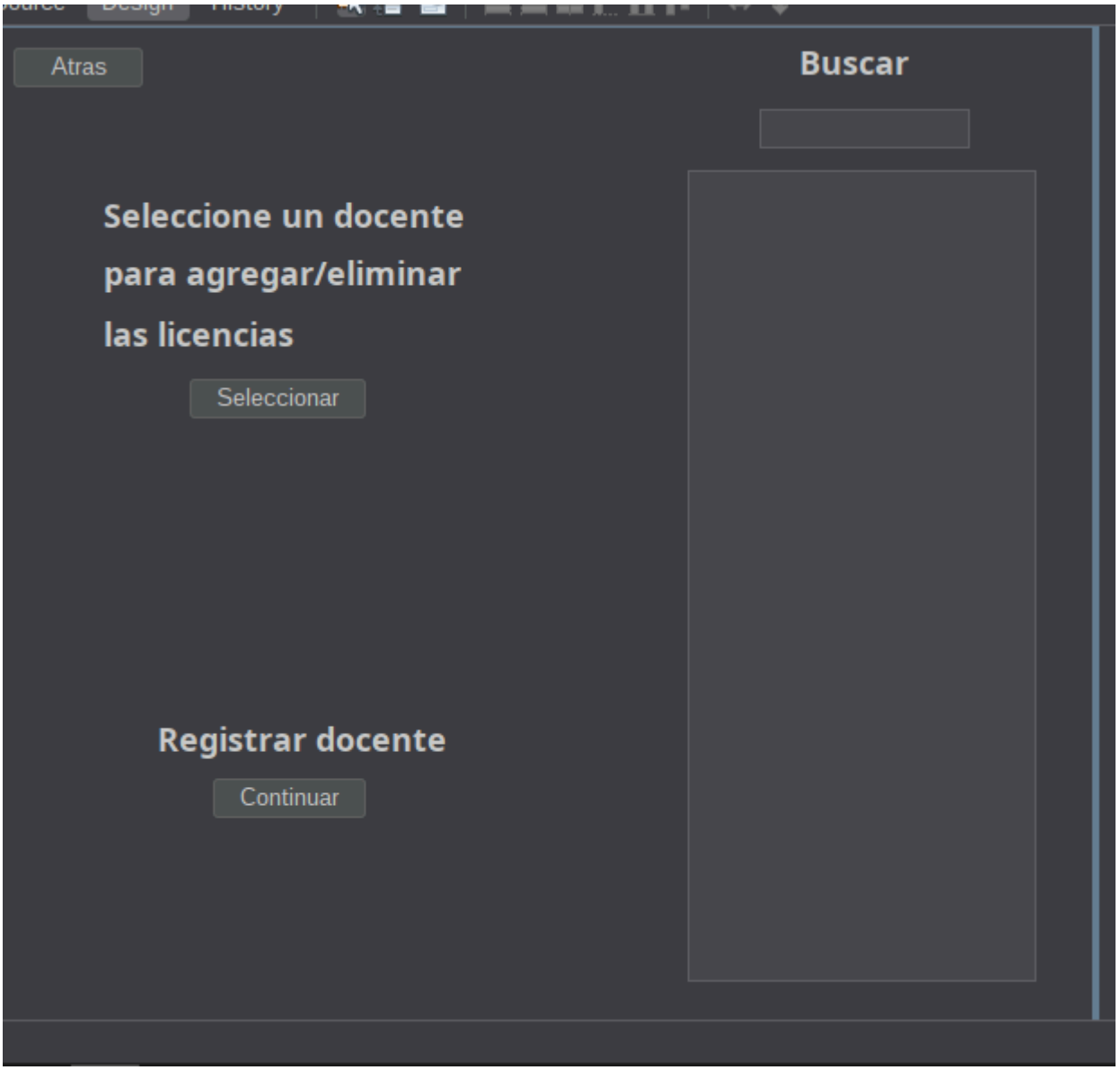
2-


Atras
Rodri Planta
CI: 5678804
Eliminar Docente
Motivo
Desde
Hasta
Grupos
yo
15/10/2025
24/10/2025
1MB; 1ME; 1MG
Refrescar
Selecciona uno
para eliminar
Eliminar
Agregar una Licencia:
octubre
2025
lun
mar
mié
jue
vie
sáb
dom
Grupos Afectados
Motivo
Limpiar
1MA
1MB
1MC
1MD
1ME
1MF
1MG
Guardar

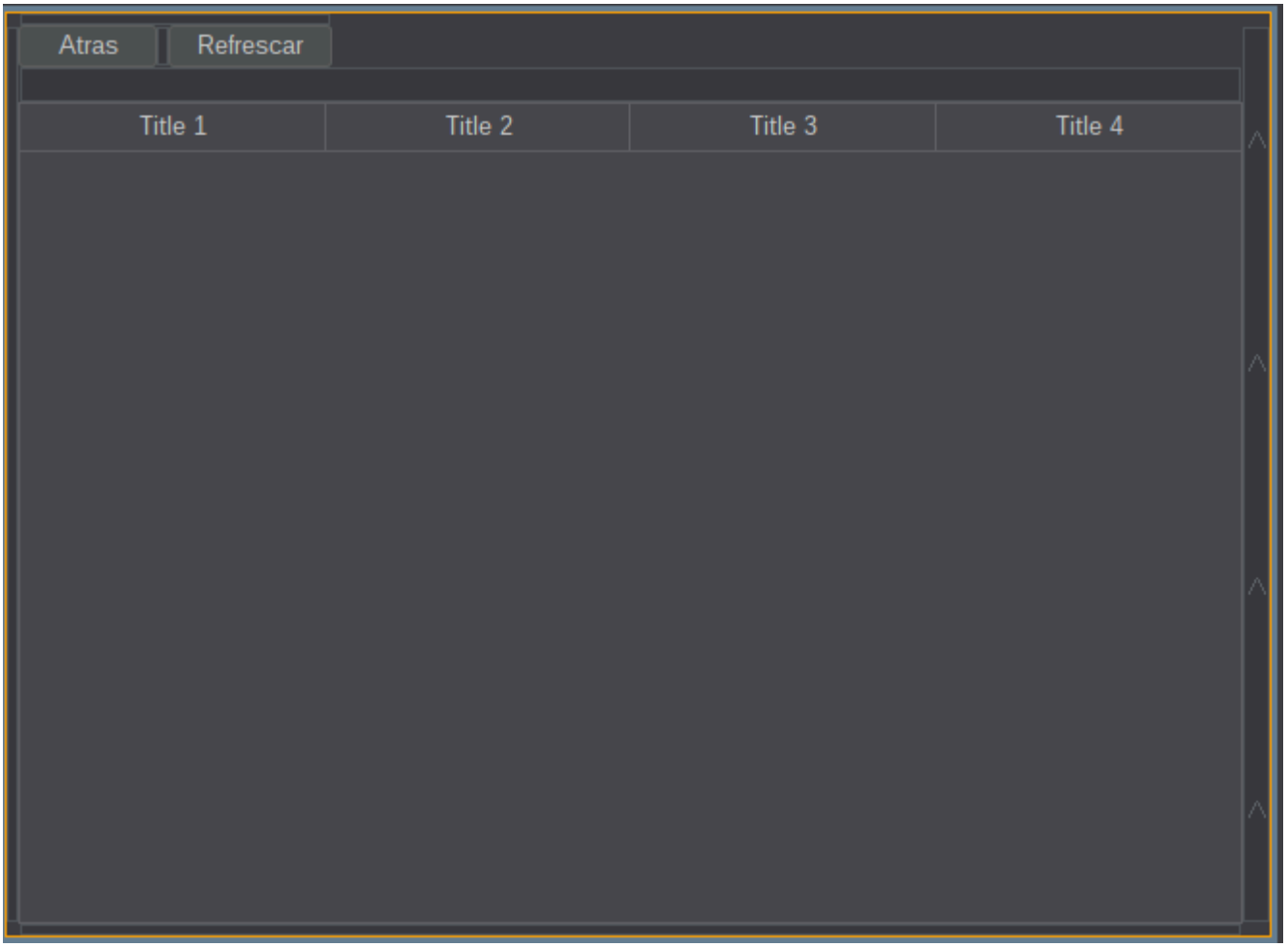
3-


Ci
Nombre
Apellido
Materia
Turno
Agregar
Atras

4-


Atras
Buscar
Seleccione un docente
para agregar/eliminar
las licencias
Seleccionar
Registrar docente
Continuar

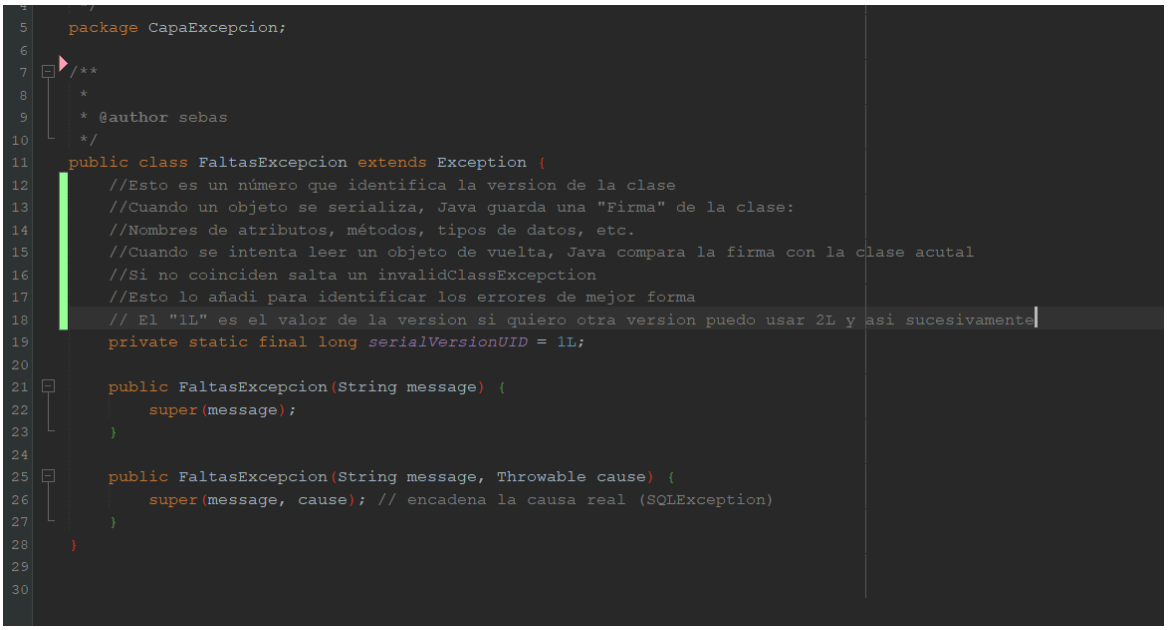
5-



Capa Excepción

Clase FaltasExcepción

Esta clase permite verificar los errores que puedan aparecer, se utilizó para comprobar errores que ocurrían.



```java
package CapaExcepcion;

/**
 *
 * @author sebas
 */
public class FaltasExcepcion extends Exception {
    //Esto es un número que identifica la version de la clase
    //Cuando un objeto se serializa, Java guarda una "Firma" de la clase:
    //Nombres de atributos, métodos, tipos de datos, etc.
    //Cuando se intenta leer un objeto de vuelta, Java compara la firma con la clase acutal
    //Si no coinciden salta un invalidClassExcepction
    //Esto lo añadi para identificar los errores de mejor forma
    // El "1L" es el valor de la version si quiero otra version puedo usar 2L y asi sucesivamente
    private static final long serialVersionUID = 1L;

    public FaltasExcepcion(String message) {
        super(message);
    }

    public FaltasExcepcion(String message, Throwable cause) {
        super(message, cause); // encadena la causa real (SQLException)
    }
}
```

Clases BDexception

Sirve para comprobar errores todo lo relacionado a las consultas de Base de datos.

```java
package CapaExcepcion;
import java.sql.SQLException;

/**
 *
 * @author sebas
 */
public class BDexcepcion extends Exception {
     public BDexcepcion(String string){
     super(string);
 }
}
```

Capa Persistencia

Clase BaseDeDatos

Sirve para conectar la base de datos, también previniendo errores de conexión de base de datos

```java
package CapaPersistencia;

import CapaExcepcion.BDexcepcion;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
/**
 *
 * @author sebas
 */
public class BaseDeDatos {
    public static Connection getConnection() throws BDexcepcion{
    Connection con=null;
    try{
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/proyectofaltas?zeroDateTimeBehavior=CONVERT_TO_NULL","root","");
    }catch (SQLException sqle){
        throw new BDexcepcion("Error de conexion de base de datos");
    }
    return con;
}
}
```

Clases LicenciasFilas

Permite armar una fila de licencia perfecta para preparar el armado para luego enviarla a una tabla ubicada en la parte gráfica.

```java
package CapaPersistencia;

/**...4 lines */
//Esta clase se encargara de armar la fila para mostrarla en la tabla
//Por eso LicenciaRow (Row = fila pero en ingles :v)
public class LicenciaFilas {
    private int id = 0;
    private final String cedula;
    private final String docente;
    private final String materia;
    private final String turno;
    private final String motivo;
    private final java.time.LocalDate desde;
    private final java.time.LocalDate hasta;
    private final String grupos;

    public LicenciaFilas(int id, String cedula, String docente, String materia, String turno,
                         String motivo, java.time.LocalDate desde, java.time.LocalDate hasta,
                         String grupos) {
        this.id = id;
        this.cedula = cedula;
        this.docente = docente;
        this.materia = materia;
        this.turno = turno;
        this.motivo = motivo;
        this.desde = desde;
        this.hasta = hasta;
        this.grupos = grupos;
    }
```

Clase ConsultasFaltas

```java
public class ConsultasFaltas {
    //help me no queria poner tantas consultas :(
    // DOCENTES
private static final String SQL_GUARDAR_DOCENTE =
    "INSERT INTO proyectofaltas.docentes(Cedula, Nombre, Apellido, Materia, Turno) VALUES (?,?,?,?,?)";

private static final String SQL_CONSULTAR_DOCENTE =
    "SELECT * FROM proyectofaltas.docentes WHERE Cedula = ?";

private static final String SQL_ELIMINAR_DOCENTE =
    "DELETE FROM proyectofaltas.docentes WHERE Cedula = ?";

private static final String SQL_LISTAR_DOCENTES =
    "SELECT * FROM proyectofaltas.docentes ORDER BY Apellido, Nombre";

// LICENCIAS sin Grupos_afectados
private static final String SQL_GUARDAR_LICENCIA =
    "INSERT INTO proyectofaltas.licencias (docente_ci, Motivo, Fecha_inicio, Fecha_fin) VALUES (?, ?, ?, ?)";

private static final String SQL_LISTAR_LICENCIAS_POR_DOCENTE =
    "SELECT l.id, l.Motivo, l.Fecha_inicio, l.Fecha_fin, " +
    "COALESCE(GROUP_CONCAT(lg.grupo_codigo ORDER BY lg.grupo_codigo SEPARATOR '; '), '') AS Grupos " +
    "FROM proyectofaltas.licencias l " +
    "LEFT JOIN proyectofaltas.licencias_grupos lg ON lg.licencia_id = l.id " +
    "WHERE l.docente_ci = ? " +
    "GROUP BY l.id " +
    "ORDER BY l.Fecha_inicio DESC";

private static final String SQL_ELIMINAR_LICENCIA =
    "DELETE FROM proyectofaltas.licencias WHERE id = ? AND docente_ci = ?";

// Tabla licencia ↔ grupos
private static final String SQL_INSERT_LIC_GRUPO =
    "INSERT INTO proyectofaltas.licencias_grupos (licencia_id, grupo_codigo) VALUES (?, ?)";

```

Modificaciones respecto a la anterior entrega:

Se ha eliminado a **Rafael Giacoia** del proyecto y se le dio su rol a **Joaquin Gimenez**, además arreglamos el link figma,arreglamos el casos de uso además de aplicar el formato APA a el texto.Respecto a la capa lógica se agrego los grupos como una lista dentro de licencias.Respecto a la capa gráfica se tuvo que cambiar todo ya que el equipo se dio cuenta que le faltaba coherencia y conectividad con la capa lógica. En la parte de base de datos agregamos una tabla llamada Licencia_Grupos. Y al cambiar la capa gráfica tuvimos que cambiar el casos de uso y el figma.

Conclusión

Este trabajo nos permitió conocer sobre la configuración de redes y servidores en entornos simulados, aplicando conceptos fundamentales de direccionamiento IP, servicios de

red y diseño de topologías. El uso de Cisco Packet Tracer nos brindó una experiencia práctica en la administración de redes. Además este proyecto nos ha enseñado a trabajar mejor en equipo y cosas que necesitaremos a futuro como por ejemplo cómo conectar una base de datos y un código en apache.Además de enseñarnos a agregar un calendario en la parte grafica.

Biografía

OpenAI. (2025). *Chat GPT* [GPT-5]. OpenAI. https://chat.openai.com/.

Enseñanzas en clases de programación instruidas por la profesora Marcela Maderos.