ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

ASIGNATURA:	Tecnologia Desarrollo de Software
PROFESOR:	Ing. Juan Carlos Gonzalez MSc.
PERÍODO ACADÉMICO:	2025-B

DOCUMENTACIÓN TÉCNICA DEL SISTEMA

PROYECTO DE PROGRAMACION ORIENTADA A OBJETOS

Nombre del sistema: Sistema de licencias de Conducir V2.0

Autores:

Ismael Narváez

Edwin Chusin

Fecha:

09-01-2026

Resumen ejecutivo

El sistema de Gestión de Licencias de Conducir es una aplicación de escritorio desarrollado en Java, utilizando el entorno de desarrollo Intellij IDEA, en su versión inicial (V1.0), el sistema permitía la administración conductores mediante su registro, la validación de documentos, la evaluación de pruebas psicométricas y todo esto genera un reporte en formato PDF. Para la versión V2.0 el sistema se actualizo incorporando nuevas funcionalidades orientadas a mejorar la gestión y seguridad de la información. Entre las principales mejoras se incluye la generación de un reporte general que indica todos los conductores registrados y su estado de emisión de licencia. Además, se implementó un sistema de gestión de usuarios con control de acceso por roles, donde se define 2 tipos de super usuarios, como lo son el "Administrador" y el "Analista", cada uno con su respectiva funcionalidad y responsabilidad de gestión dentro del sistema.

El sistema registra tanto usuarios que pueden administrar la gestión de datos dentro del sistema como también registran los conductores u operadores que hacen uso de una licencia de conducción. La estructura de las funciones que cumple cada rol se describe como:

Administrador: Registra, actualiza, elimina y da un reporte de los super usuarios, permitiendo un mejor control del uso del sistema garantizando seguridad e integridad de datos.

Analista: Registra conductores postulantes, valida documentos, realiza los resultados de las pruebas psicométricas dando luz verde para validar la emisión de una licencia, genera un reporte de los conductores en el sistema y genera un documento PDF de un conductor.

El sistema está desarrollado bajo la arquitectura MVC (Model, View, Controller), lo que permite una adecuada separación de responsabilidades. Utiliza también una conexión a base de datos PostgreSQl en la nube y control de acceso por roles como lo son "Administrador" y "Analista".

Para llevar a cabo el proyecto, se emplearon la siguiente lista de tecnologías:

Java21 (Intellij IDEA)

Gradle (Dependencies)

Swing + CardLayout

MySql -> PostgreSQL Cloud (Local para pruebas)

iText(Generar PDF)

Arquitecura MVC

Arquitectura del sistema

Arquitectura general

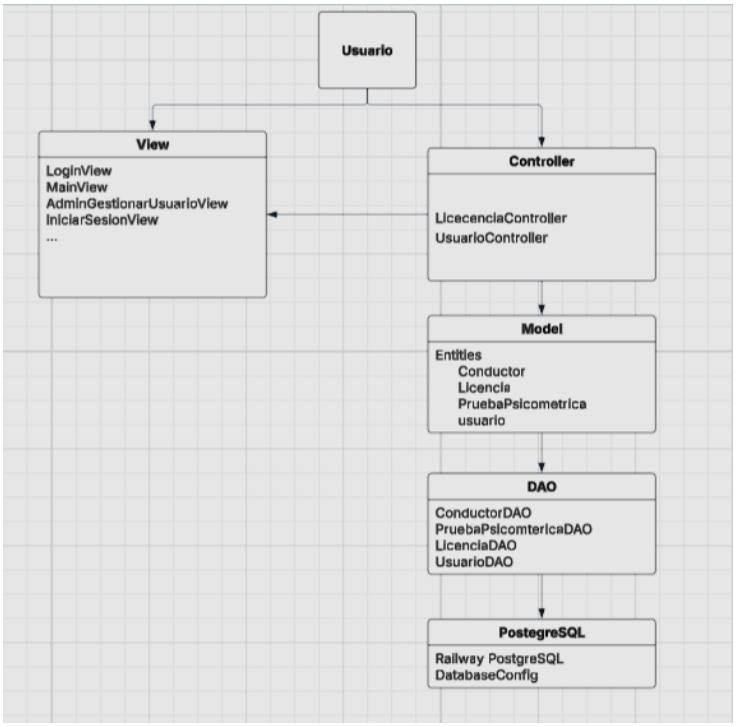
El sistema de Gestión de licencias de Conducir, esta desarrollado bajo el patrón de arquitectura MVC(Model-View-Controller), el cual permite una separación de responsabilidades, lo que facilitara el mantenimiento de este, además de su escalabilidad para futuras actualizaciones y comprensión del sistema que es muy importante. Esta arquitectura organiza las aplicaciones separándolas en 3 componentes que están interconectados para gestionar datos, interfaz de usuario y lógica de control, estos componentes se describen como:

Model (modelo): Gestiona la lógica del negocio y se encarga de los datos, generalmente (pero no obligatoriamente) consultando la base de datos. Actualizaciones, consultas, búsquedas, etc. Gestiona el acceso a la base de datos mediante el patrón DAO.

Controller (Controlador): Actúa como intermediario entre el modelo y la vista, su funcionalidad se basa en recibir las peticiones del usuario desde la vista, seguido de procesar la lógica de aplicación y coordina las operaciones que se enviaran a la base de datos y, por último, solicita al modelo los datos nuevos y lo actualiza en la vista.

View (Vista): Son la representación visual de los datos, donde ni el modelo ni el controlador se preocupan de cómo se verán los datos, esa responsabilidad va dirigida solo a la vista.

Diagrama de arquitectura MVC



El diagrama presenta la arquitectura MVC que esta implementada en el sistema. El Usuario interactúa directamente con la vista, lo cual envía los eventos al controlador, seguido de que este porcesa la lógica del sistema y se comunica con el modelo, el cual encapsula las reglas de negocio. Para el acceso a los datos se utiliza el patron DAO, permititendo una comunicación desacoplada con la base de datos PostgreSQl.

Tecnologías

Java21+Gradle

Java 21 LTS (Long Term Support), es el lengauje principal usado para el desarrollo de la aplicación, LTS significa que es una versión estable y confiable, con soporte extendido por años lo cual es bueno para entornos de trabajo empresariales generalmente, es de multiplataforma que soporta tanto Windows, Linux y macOS.

Gradle

Una herramienta de automatización de compilación (build tool) de código abierto. Usado para sistema de gestión de dependencias, su configuración es mediante build.gradle y genera el JAR ejecutable.

MySQL ->PostgreSQL

El sistema se migro de MySQl a PostgreSQL en su versión V2.0:

Version V1.0: MySQL (Local)

Es una base de datos relacional local, por ende su configuración de servidor es en local y tiene sus limitaciones de acceso remoto.

Version V2.0: PostgreSQL (Cloud)

La base de datos ahora se encuentra en la nube, el acceso remoto es desde cualquier ubicación, este ofrece mejor rendimiento para aquellas consultas complejas y ofrece un soporte para tipos de datos avanzados.

Su conexión es mediante el archivo config/DatabaseConfig.java donde se obtiene las credenciales y URL de la base de datos cloud para posterior conectarlo mediante JDBC PostgreSQL Dirver.

Swing + Cardloyut + iText

Java Swing es aquel framework para interfaz grafica de usuario (GUI), se base en JFrame y JPanel, el Look and Feel es personalizable y su fácil gestión de eventos para buttons, radioButtons, comboBox, entre otros, es mediante listeners.

Cardlayout es un gestor de diseño que organiza los componentes con una pila de cartas, que muestra solo un componente a la vez en el mismo espacio, permitiendo alternar entre diferentes paneles mediante botones o menus, este no extiende de un JFrame o un JPanel como un componente normal de Java Swing.

iTextPDF

Es una librería para la generación de documentos PDF, permite modificar archivos PDF existentes, rellena formularios o extrae información incluso.

Diagrama de capas

Estructura de paquetes

config/

Configuración de conexión a base de datos

Parámetros de configuración del sistema

controller/

Controladores que manejan la lógica de la aplicación

Intermediarios entre Vista y Modelo

dao/

Data Access Objects

Operaciones CRUD sobre la base de datos

Mapeo de entidades a tablas

dto/

Data Transfer Objects

Objetos para transferir datos entre capas

model/entities/

Clases Java que representan las entidades del dominio con getters/setters

model/interfaces/

Constantes del sistema

Interfaces para contratos de implementación

service/

Lógica de negocio compleja

Orquestación de múltiples DAOs

util/

Generadores de PDF

view/

Todas las interfaces gráficas Swing

JFrames y JDialogs del sistema

Cardlayout para el View de usuario.

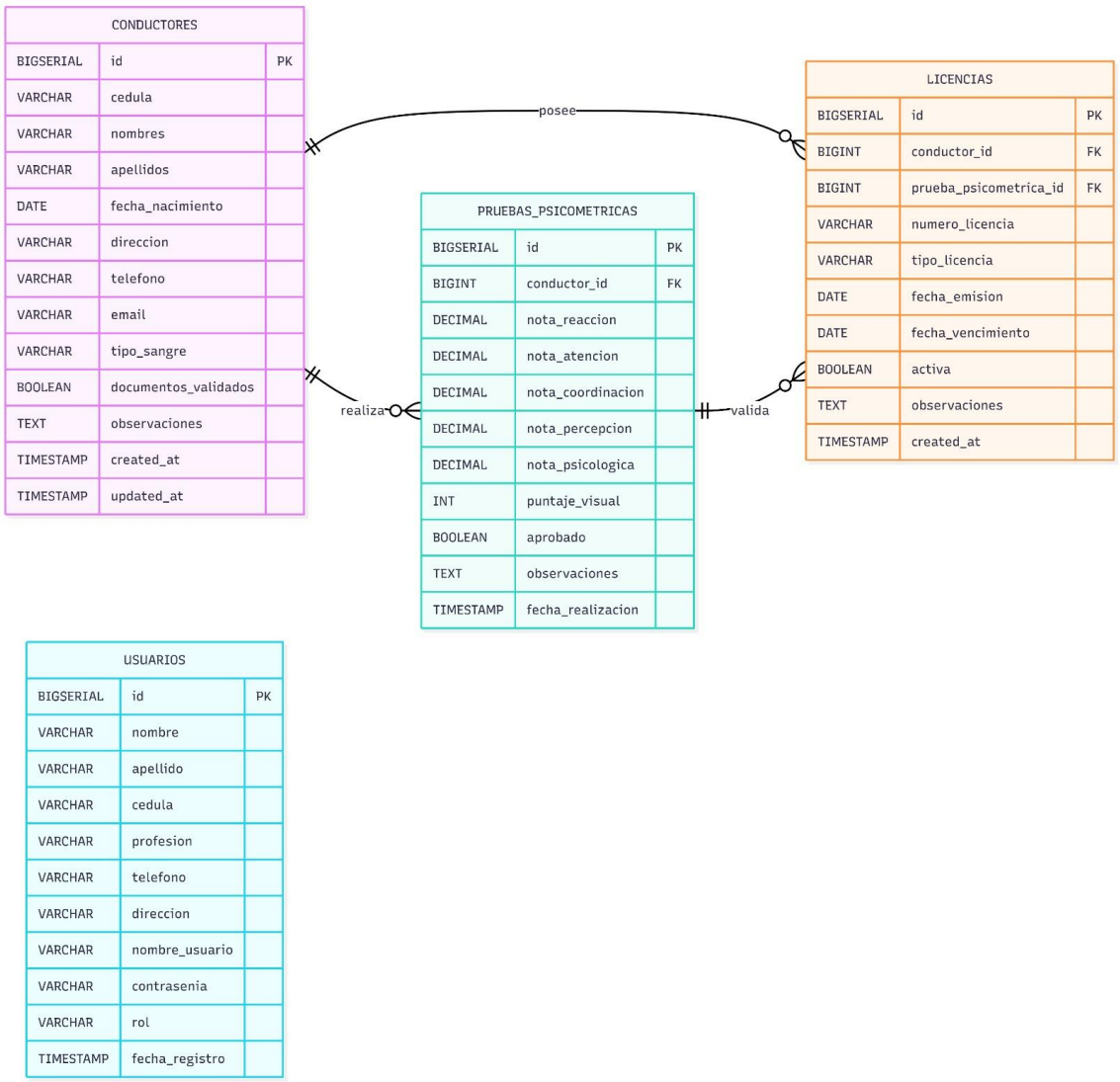
resources/

Recursos estáticos (iconos, imágenes)

Archivos de configuración

BASES DE DATOS

Diagrama ER simple



Tablas principales

Para este sistema las entidades o tablas principales que se denominan son:

Conductores:	**Pruebas_Psicometricas**	**Licencias**
id	id	id
cedula	conductor_id	conductor_id
nombres	nota_reaccion	prueba_psicometrica_id
apellidos	nota_atencion	numero_licencia
fecha_nacimiento	nota_coordinacion	tipo_licencia
direccion	nota_percepcion	fecha_emision
telefono	nota_psicologica	fecha_vencimiento
email	puntaje_visual	activa
tipo_sangre	aprobado	observaciones
documentos_validados	observaciones	created_at
observaciones	fecha_realizacion	
created_at		
updated_at		

Usuarios
id
nombre
apellido
cedula
profesion
telefono
direccion
nombre_usuario
contrasenia
rol
fecha_registro

Relaciones básicas:

1. Un conductor puede realizar una o varias pruebas psicométricas.
2. Un conductor puede poseer una licencia de conducir.
3. Una licencia está asociada a una prueba psicométrica aprobada.
4. Los usuarios del sistema gestionan la información de conductores y licencias según su rol asignado.

FUNCIONALIDAES

El sistema de Gestión de Licencias de Conducir implementa un conjunto de funcionalidades organizadas por versiones, las cuales interactúan bajo la arquitectura MVC y utilizan patrones de diseño como DAO y Singleton para garantizar una correcta separación de responsabilidades y un acceso eficiente a los datos.

VERSIÓN 1.0

Gestión de conductores

La gestión de conductores permite registrar, actualizar, consultar y almacenar la información personal de los conductores. Esta funcionalidad sigue el flujo MVC, donde la vista captura los datos mediante formularios en Swing, el controlador valida la información ingresada y el modelo representa la entidad Conductor. El acceso a la base de datos se realiza mediante un DAO, evitando que el controlador interactúe directamente con SQL.

Ejemplo de interacción en el controlador:

```
public void registrarConductor(Conductor c) {
    if(c.getCedula().isEmpty()) {
        throw new IllegalArgumentException("Cédula obligatoria");
    }
    conductorDAO.insertar(c);
}
```

Ejemplo de DAO:

```
public void insertar(Conductor c) {
    String sql = "INSERT INTO conductores (cedula, nombres, apellidos)
VALUES (?, ?, ?)";
    PreparedStatement ps = conexion.prepareStatement(sql);
    ps.setString(1, c.getCedula());
    ps.setString(2, c.getNombres());
    ps.setString(3, c.getApellidos());
    ps.executeUpdate();
}
```

Emisión de licencias

La emisión de licencias se realiza una vez que el conductor ha cumplido con los requisitos establecidos. El controlador verifica que el conductor tenga documentos validados y pruebas psicométricas aprobadas antes de permitir la emisión. El modelo Licencia encapsula los datos de la licencia y el DAO se encarga de su persistencia en la base de datos.

```
public void emitirLicencia(Licencia l) {
    if(!l.isActiva()) {
        throw new IllegalStateException("Licencia no válida");
    }
```

```
    licenciaDAO.insertar(l);
}
```

Validación de documentos

Esta funcionalidad permite registrar el estado de validación de los documentos del conductor. La vista presenta opciones de validación, el controlador procesa la acción y actualiza el estado en la base de datos mediante el DAO correspondiente. Esta validación es utilizada como condición para otras funcionalidades del sistema.

```
public void validarDocumentos(Long conductorId) {
    conductorDAO.actualizarDocumentos(conductorId, true);
}
```

Pruebas psicométricas

El sistema permite registrar los resultados de las pruebas psicométricas del conductor. El controlador recibe los valores ingresados, evalúa si el conductor aprueba o no y almacena los resultados en la base de datos. Esta funcionalidad aplica reglas de negocio dentro del controlador y utiliza el DAO para el almacenamiento.

```
public boolean evaluarPrueba(PruebaPsicometrica p) {
    return p.getNotaPsicologica() >= 7 && p.getPuntajeVisual() >= 8;
}
```

Consulta de licencias

La consulta de licencias permite visualizar el estado de una licencia emitida. El controlador solicita la información al DAO y la vista presenta los resultados al usuario. Esta funcionalidad es de solo lectura y no modifica los datos almacenados.

```
public Licencia buscarPorNumero(String numero) {
    return licenciaDAO.buscarPorNumero(numero);
}
```

VERSIÓN 2.0

Login y autenticación

En la versión 2.0 se implementa un sistema de autenticación que valida las credenciales del usuario antes de permitir el acceso al sistema. La vista captura el usuario y contraseña, el controlador valida la información y consulta al DAO de usuarios. En caso de éxito, se inicia una sesión activa.

```
public Usuario login(String user, String pass) {
    return usuarioDAO.validarCredenciales(user, pass);
}
```

Roles Administrador y Analista

El sistema restringe las funcionalidades disponibles según el rol del usuario autenticado. Esta lógica se gestiona desde el controlador y la sesión del usuario, permitiendo mostrar u ocultar opciones en la interfaz gráfica.

```
if(usuario.getRol().equals("ADMINISTRADOR")) {
      habilitarGestionUsuarios();
  }
```

CRUD de usuarios

El Administrador puede registrar, actualizar y eliminar usuarios del sistema. Esta funcionalidad sigue el patrón MVC completo y utiliza el DAO para gestionar los datos de los usuarios en la base de datos.

```
public void eliminarUsuario(Long id) {
    usuarioDAO.eliminar(id);
}
```

Reportes PDF

El sistema genera reportes en formato PDF tanto individuales como generales. Esta funcionalidad se implementa mediante la librería iText, encapsulada en una clase utilitaria. El controlador invoca esta clase para generar los documentos.

```
PDFGenerator.generarReporteConductor(conductor);
```

PostgreSQL en la nube

La versión 2.0 utiliza una base de datos PostgreSQL alojada en la nube. El acceso se gestiona mediante una única conexión centralizada usando el patrón Singleton, garantizando eficiencia y control de recursos.

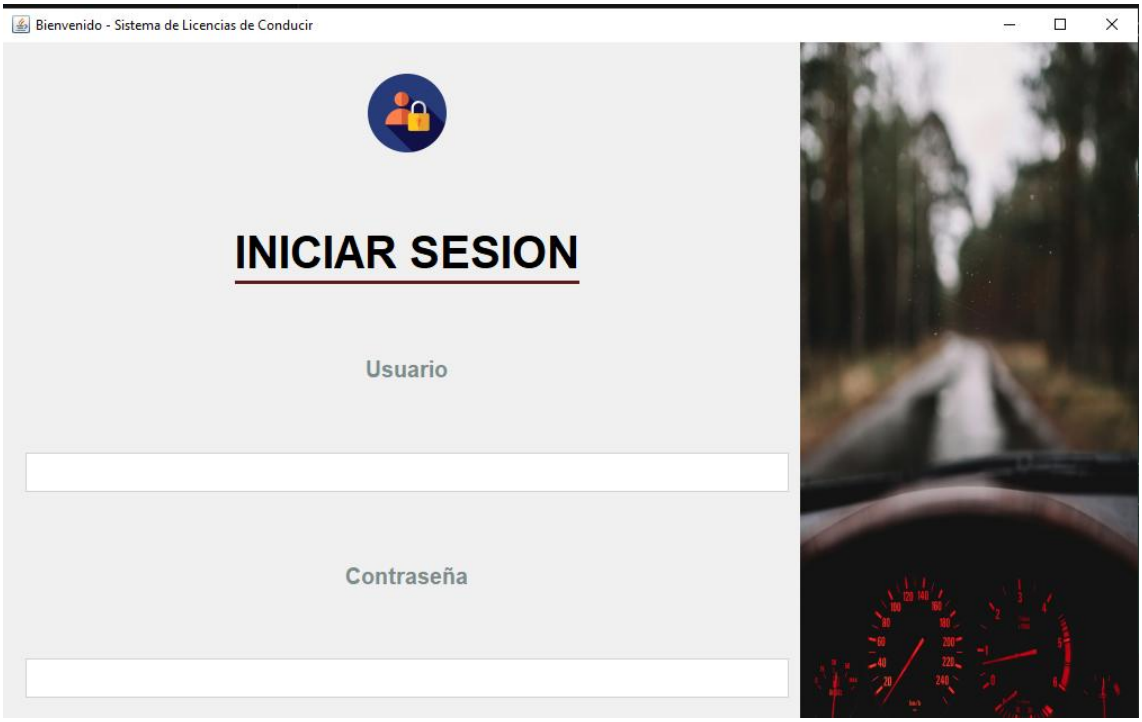
```
public static Connection getConexion() {
    if(conexion == null) {
        conexion = DriverManager.getConnection(url, user, pass);
    }
    return conexion;
}
```

Todas las funcionalidades descritas interactúan bajo la arquitectura MVC, donde las vistas gestionan la interfaz gráfica, los controladores coordinan la lógica del sistema y los modelos representan los datos y reglas de negocio. El uso de los

patrones DAO y Singleton garantiza un sistema desacoplado, mantenible y escalable.

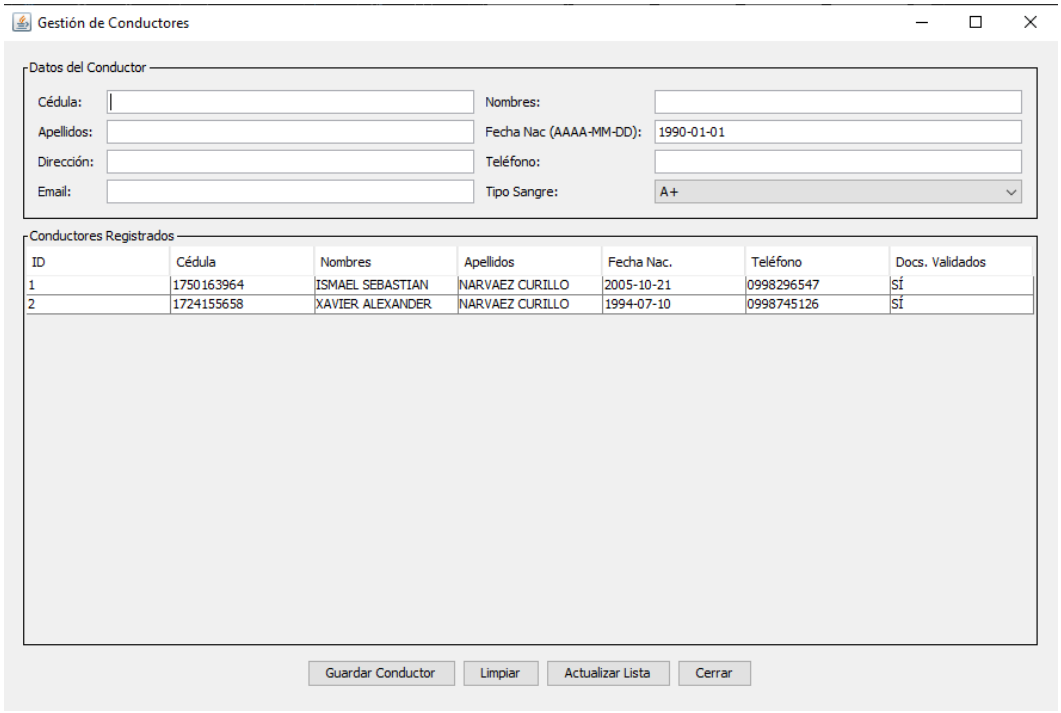
VISTAS DE PANTALLA DEL SISTEMA

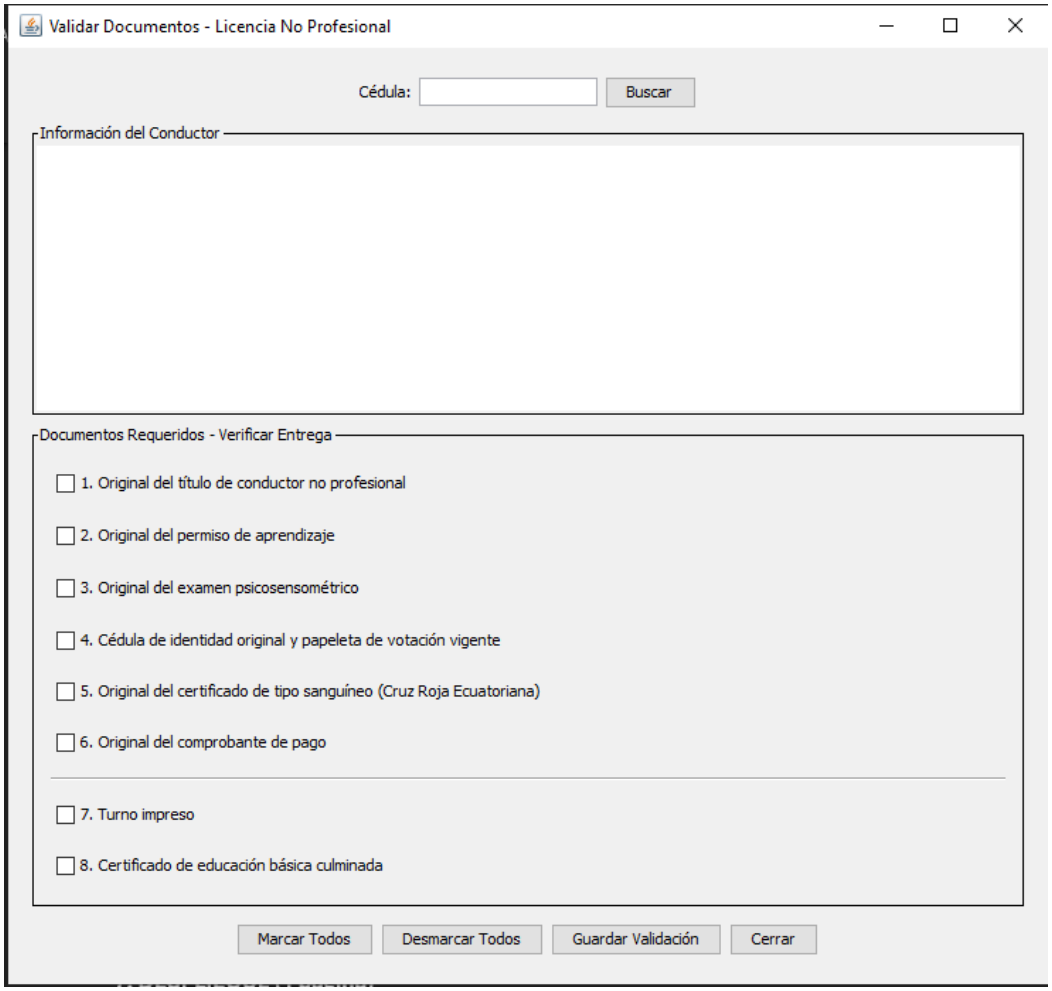
- LoginView (nueva)

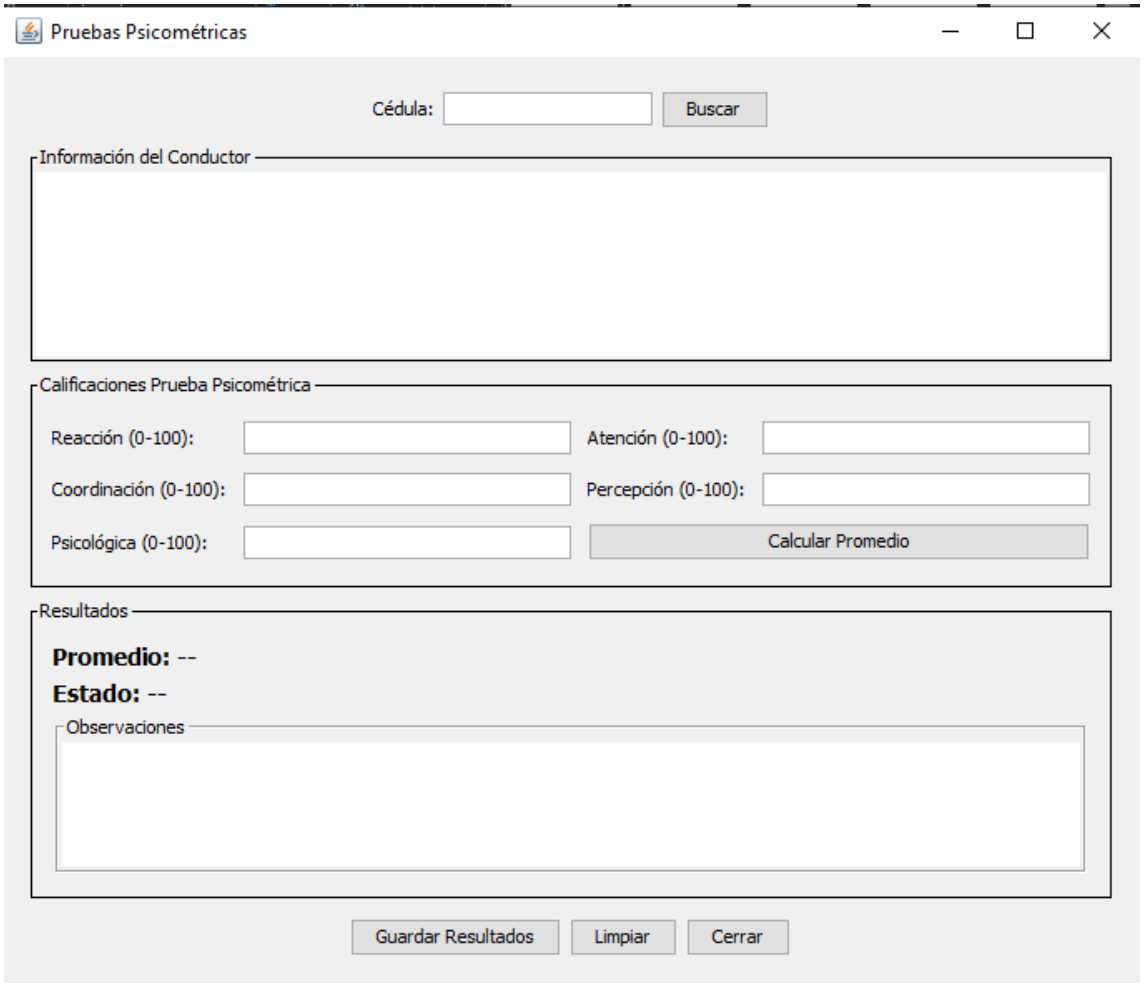


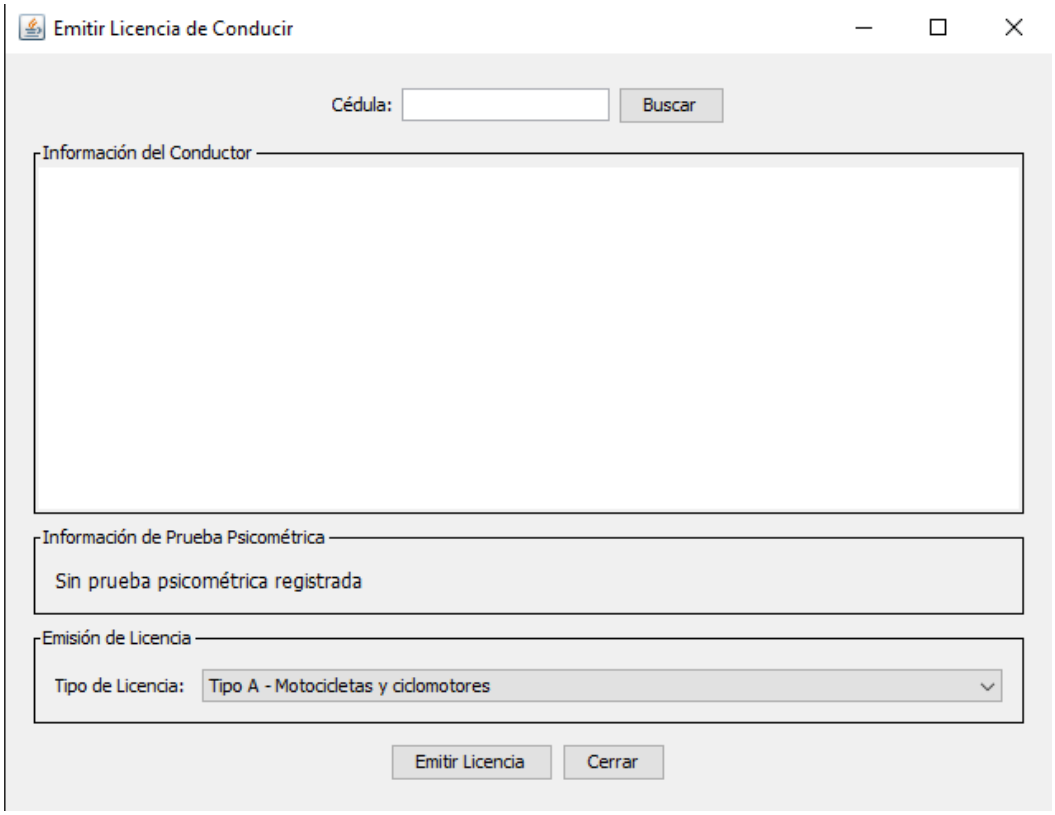
- MainView

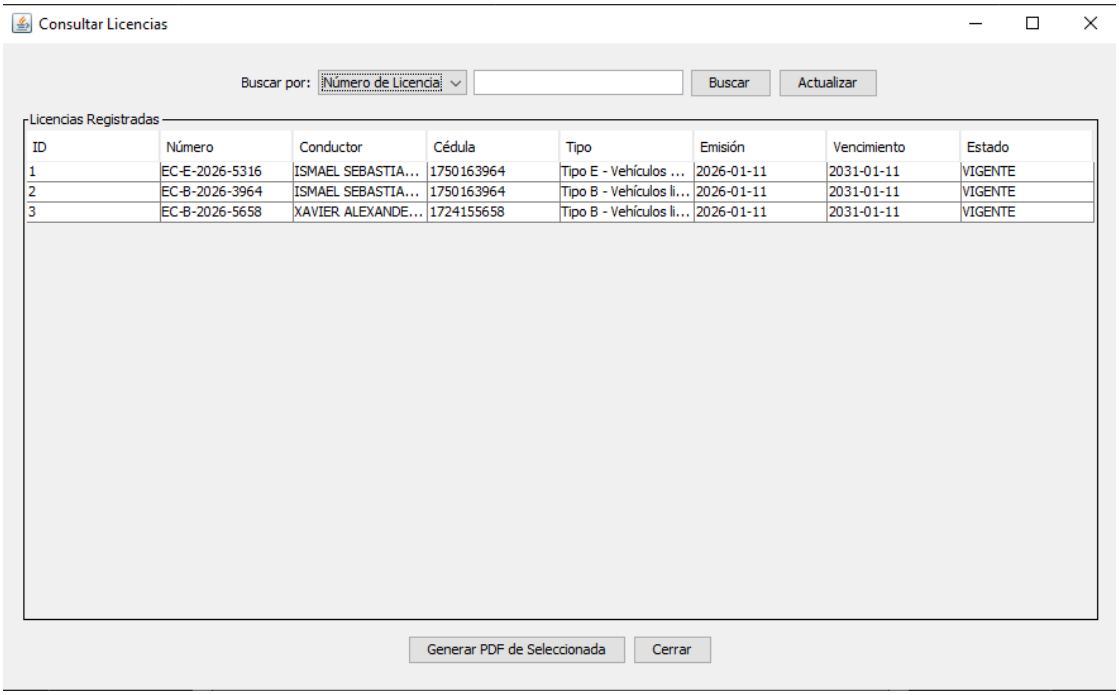
- EmitirLicenciaView





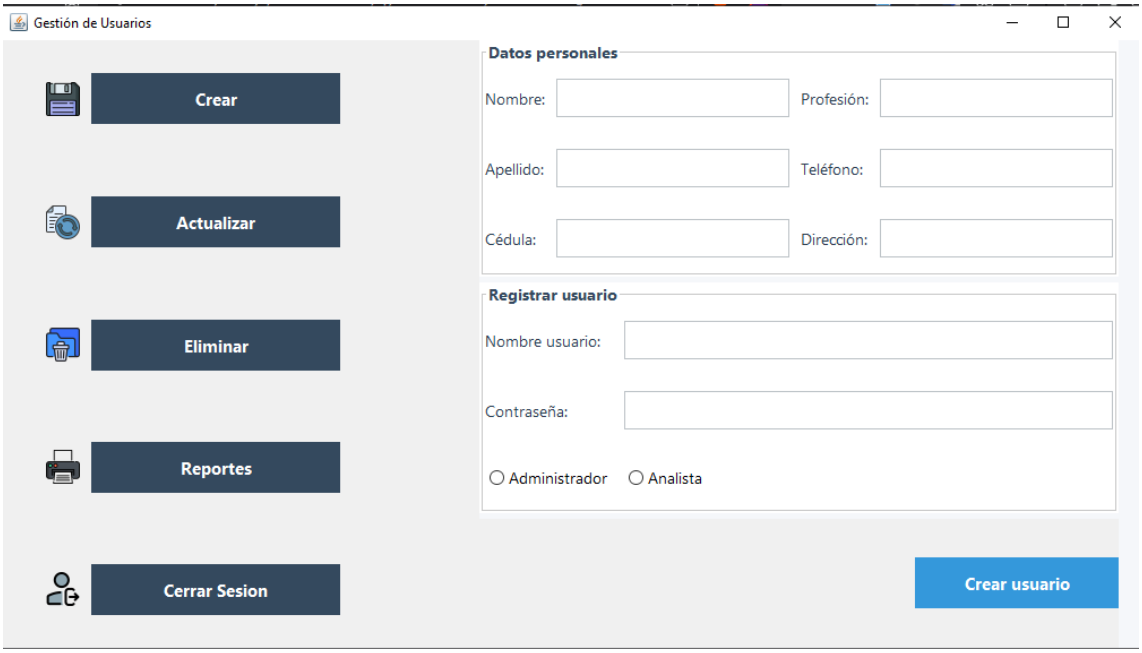

Pruebas Psicométricas
Cédula:
Buscar
Información del Conductor
Calificaciones Prueba Psicométrica
Reacción (0-100):
Atención (0-100):
Coordinación (0-100):
Percepción (0-100):
Psicológica (0-100):
Calcular Promedio
Resultados
Promedio: --
Estado: --
Observaciones
Guardar Resultados
Limpiar
Cerrar

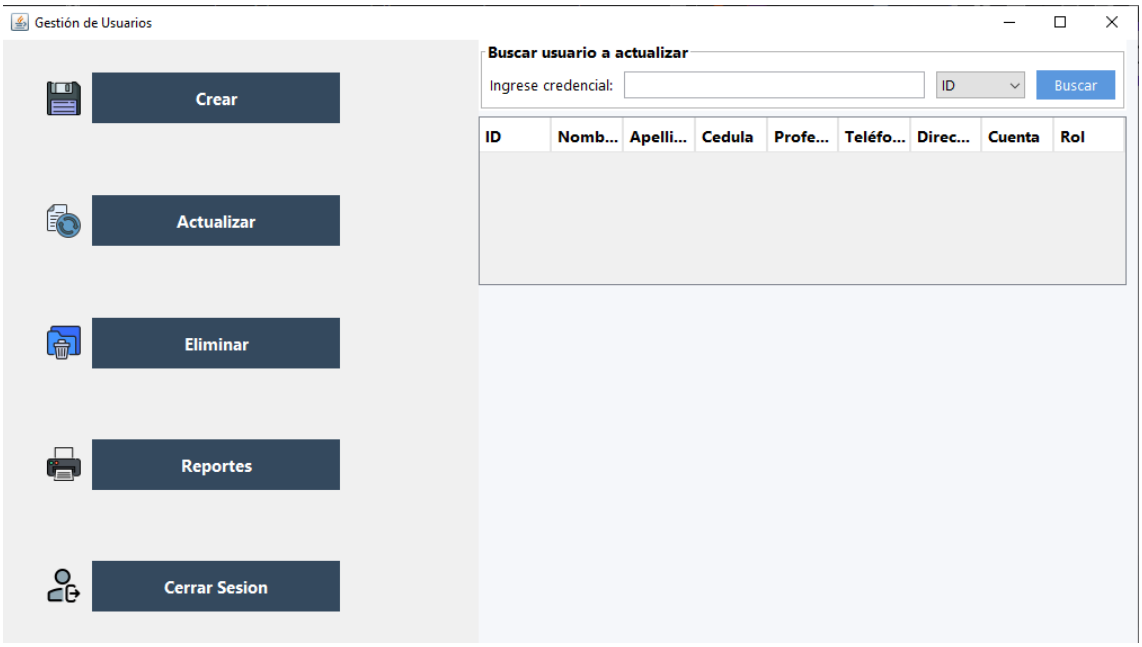

Emitir Licencia de Conducir
Cédula:
Buscar
Información del Conductor
Información de Prueba Psicométrica
Sin prueba psicométrica registrada
Emisión de Licencia
Tipo de Licencia:
Tipo A - Motocicletas y ciclomotores
Emitir Licencia
Cerrar



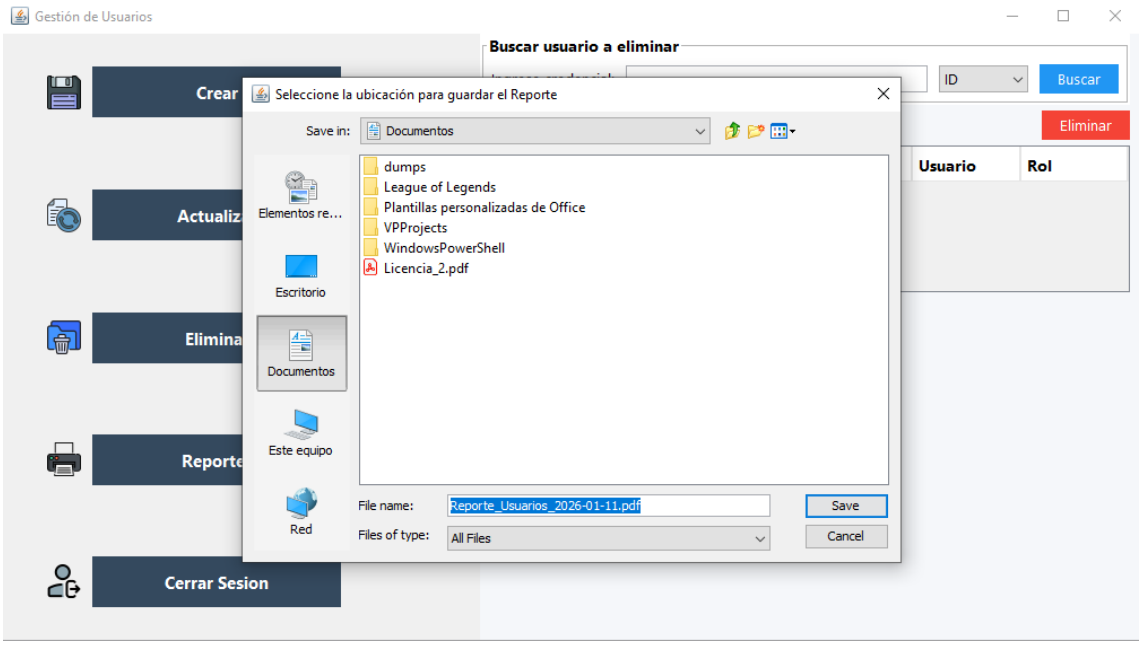
- GestionUsuariosView (nueva)


Gestión de Usuarios
Crear
Actualizar
Eliminar
Reportes
Cerrar Sesion
Bienvenido al sistema


Gestión de Usuarios
Crear
Actualizar
Eliminar
Reportes
Cerrar Sesion
Datos personales
Nombre:
Profesión:
Apellido:
Teléfono:
Cédula:
Dirección:
Registrar usuario
Nombre usuario:
Contraseña:
Administrador
Analista
Crear usuario


Gestión de Usuarios
Crear
Actualizar
Eliminar
Reportes
Cerrar Sesion
Buscar usuario a actualizar
Ingrese credencial:
ID
Buscar
ID
Nomb...
Apelli...
Cedula
Profe...
Teléfo...
Direc...
Cuenta
Rol


Gestión de Usuarios
Crear
Actualizar
Eliminar
Reportes
Cerrar Sesion
Buscar usuario a eliminar
Ingrese credencial:
ID
Buscar
Eliminar
ID
Nombres
Apellidos
Cedula
Usuario
Rol



7. DESPLIEGUE (1 página)

- Requisitos del sistema
- Requisitos de Hadware
- Procesador: Intel o AMD de 64 bits (mínimo 2 núcleos)
  - Memoria RAM:
  - Mínimo: 4 GB
  - Recomendado: 8 GB
- Espacio en disco:
  - Mínimo: 500 MB libres
  - Recomendado: 1 GB libre
  - Versión de Java 21.0.8 o posterior

Requisitos de Software

- **Sistema Operativo:**
  - Windows 10 (64 bits)
  - Windows 11 (64 bits)
- **Java Runtime Environment (JRE):**
  - **Java 21 (LTS)** o superior
  - Requerido para la ejecución del archivo .exe generado a partir del proyecto Java
- **Base de Datos:**

- PostgreSQL 14 o superior
- Puede ejecutarse localmente o en la nube (Railway)

- **Conectividad a Internet:**
  - Obligatoria si la base de datos se encuentra alojada en Railway
  - Opcional si se utiliza una base de datos local

Instalación

Requisitos Previos

- **Sistema Operativo:**
  - Windows 8 en adelante (64 bits)
- **Java Runtime Environment (JRE):**
  - Java **21 LTS** correctamente instalado y configurado
- **Permisos de Administrador:**
  - Necesarios para instalar y ejecutar el archivo .exe
- **Conexión a Internet:**
  - Requerida durante la instalación si se utiliza base de datos en la nube (Rail

Para la instalación del archivo ejecutable es necesario contar con JRE 21 o superior previamente instalado en el sistema. Una vez descargado el archivo, se ejecuta directamente, lo que inicia el proceso de instalación en el sistema. Durante este proceso la aplicación carga su configuración interna y establece la conexión con la base de datos alojada en la nube. Finalizada la validación, el sistema queda listo para su uso sin requerir configuraciones adicionales.

- Configuración BD cloud

Para esto es necesario agregar dependencias al gestor de dependencias de grendle utilizado en este proyecto, una vez configurada las dependencias, se procede con la impelmentacion del servidor y la creación en PgAdmin4 para luego vincular al servicio de Railway mediante las credenciales dadas por la base de datos en la nube, luego se crean las tablas y los registros. Una vez hecho este proceso la base de datos queda alzada con los parámetros modificados en PgAdmin4.