Sistema Gestión Licencias

Angel Hernández Alonso

Kristian Águila Luna

Adrián Alonso Méndez

Grupo #22

Repositorio: <https://github.com/LuXuzDev/Sistema-Gestion-Licencias>

Índice

[Explicación y contextualización del sistema 2](#_Toc196075724)

[Roles 2](#_Toc196075725)

[Requisitos funcionales 2](#_Toc196075726)

[Gestión Entidades (Administrador) 2](#_Toc196075727)

[Gestión de Conductores (Administrador, Trabajador-Centro) 2](#_Toc196075728)

[Gestión de Infracciones (Administrador, Trabajador-Centro) 3](#_Toc196075729)

[Gestión de Licencias (Administrador, Trabajador-Centro) 3](#_Toc196075730)

[Gestión de Exámenes (Administrador, Administrador-Autoescuela, Administrador-Medico, Trabajador-Centro, Trabajador-Autoescuela, Medico) 3](#_Toc196075731)

[Gestión del sistema (Administrador) 4](#_Toc196075732)

[Requisitos no funcionales 4](#_Toc196075733)

[Rendimiento 4](#_Toc196075734)

[Seguridad 4](#_Toc196075735)

[Usabilidad 4](#_Toc196075736)

[Compatibilidad 4](#_Toc196075737)

[Mantenibilidad 5](#_Toc196075738)

[Documentación 5](#_Toc196075739)

[Mostrar reportes 5](#_Toc196075740)

[Ficha Centro 5](#_Toc196075741)

[Ficha Conductor 5](#_Toc196075742)

[Ficha Entidad Asociada 6](#_Toc196075743)

[Reporte de Licencias Emitidas en un Período de Tiempo 6](#_Toc196075744)

[Reporte de Exámenes Realizados en un Período de Tiempo 6](#_Toc196075745)

[Reporte de Infracciones Registradas en un Período de Tiempo 6](#_Toc196075746)

[Reporte Consolidado de Infracciones por Tipo en un Año Determinado 7](#_Toc196075747)

[Reporte de Conductores con Licencias Vencidas en un Período de Tiempo 7](#_Toc196075748)

[Arquitectura 8](#_Toc196075749)

[Organización de carpetas 8](#_Toc196075750)

[Consultas 8](#_Toc196075751)

[Ejemplo 8](#_Toc196075752)

[Validaciones 9](#_Toc196075753)

[Ejemplo como usar esta arquitectura 9](#_Toc196075754)

[Interfaz Usuario 10](#_Toc196075755)

[Extras 12](#_Toc196075756)

[Notaciones 12](#_Toc196075757)

# Explicación y contextualización del sistema

* Las restricciones nacen del examen médico, por esta razón la licencia se crea a partir del último examen médico del usuario para actualizarla con las restricciones pertinentes.
* Se separa conductor de aspirante porque el aspirante no tiene licencia.

# Roles

|  |  |
| --- | --- |
| **ROL** | **ACCESO AL SISTEMA** |
| Administrador | SI |
| Administrador-Autoescuela | SI |
| Administrador-Medico | SI |
| Trabajador-Autoescuela | SI |
| Trabajador-Centro | SI |
| Medico | SI |
| Conductor | NO |
| Aprendiz | NO |

# Requisitos funcionales

## Gestión Entidades (Administrador)

* Crear
* Modificar
* Eliminar
* Agregar trabajador
* Filtrar por
  + Tipo (Clínica, Autoescuela)

## Gestión de Conductores (Administrador, Trabajador-Centro)

* Crear
* Modificar
* Eliminar
* Ver todos
* Buscar por:
  + Nombre
  + Numero Licencia
  + CI

## Gestión de Infracciones (Administrador, Trabajador-Centro)

* Registrar
* Ver Todas
* Eliminar
* Modificar
  + Pagada
* Filtrar
  + Tipo
* Buscar por
  + CI conductor
  + Numero Licencia

## Gestión de Licencias (Administrador, Trabajador-Centro)

* Crear
* Eliminar
* Modificar
  + Categoría
  + Restricciones
  + Renovación
  + Estado (*vigente, vencida, suspendida, revocada*),
    - cuando se revoca una licencia se elimina del sistema
* Buscar por
  + Numero Licencia
* Filtrar por
  + Categoría
  + Tipo
  + Restricciones
* Ver todas

## Gestión de Exámenes (Administrador, Administrador-Autoescuela, Administrador-Medico, Trabajador-Centro, Trabajador-Autoescuela, Medico)

* Registrar
* Ver todos
* Eliminar
* Buscar por
  + CI Aspirante
  + CI Evaluador
* Filtrar por
  + Tipo (Medico, Teórico, Practico)
  + Resultado (Aprobado, Reprobado)
  + Entidad (Clínica, Autoescuela)

## Gestión del sistema (Administrador)

* Añadir tipo de licencia
* Añadir categoría de licencia
* Añadir tipo de infracción

# Requisitos no funcionales

### Rendimiento

* **Tiempos de respuesta:**
  + Operaciones CRUD (crear, modificar, eliminar registros): **< 1 segundo** en condiciones normales.
  + Consultas complejas con JOINs (ej: reportes): **< 3 segundos** con hasta 50,000 registros.
* **Carga inicial:** El sistema debe cargar la interfaz principal en **< 2 segundos** al iniciar.

### Seguridad

* **Autenticación:**
  + Contraseñas encriptadas.
  + Bloqueo tras **3 intentos fallidos** (protección contra fuerza bruta).
* **Integridad de datos:**
  + Validación de tipos y restricciones a nivel de **PostgreSQL** (CHECK, NOT NULL, FK).

### Usabilidad

* **Interfaz (JavaFX/FXML):**
  + Mensajes de error contextuales (ej: "La licencia ya existe" en lugar de "Error SQL 23505").
  + Interfaces intuitivas y fáciles de usar.

### Compatibilidad

* **Entorno de ejecución:**
  + **Java 17** (LTS) + **JavaFX 17**.
  + PostgreSQL **15+** (con extensión pgcrypto para encriptación opcional).
* **Sistemas operativos:**
  + Windows 10/11 (64-bit).
  + Linux (distribuciones basadas en Debian/Ubuntu).

### Mantenibilidad

* **Código Scala:**
  + Estructura modular (separación clara entre lógica de negocio, acceso a datos e interfaz).
  + Uso de sbt para gestión de dependencias.
* **Base de datos:**
  + Scripts SQL versionados (en Git o similar).

### Documentación

* **Técnica:**
  + Diagrama ER de la base de datos (usando draw.io).
  + Diccionario de datos (descripción de tablas/columnas clave).

# Mostrar reportes

Posibilidad de exportar en PDF o Excel

### Ficha Centro

* Nombre del centro
* Dirección postal del centro
* Logotipo del centro
* Teléfono y email de contacto
* Nombre del director general
* Nombre del jefe del departamento de recursos humanos
* Nombre del jefe del departamento de contabilidad
* Nombre del secretario general del sindicato

### Ficha Conductor

* Nombre del conductor
* Número de documento de identidad
* Dirección, teléfono, correo
* Foto
* Numero de licencia
* Categoría licencia (moto, automóvil, camión, autobús)
* Restricciones (uso de lentes, conducción diurna, etc.)
* Estado de la licencia
* Puntos en la licencia
* Infracciones registradas (tipo, fecha, puntos deducidos)
* Fecha Emisión Licencia
* Fecha Vencimiento Licencia

### Ficha Entidad Asociada

* Nombre de la entidad
* Tipo (autoescuela o clínica)
* Dirección
* Teléfono y email de contacto
* Nombre del director

### Reporte de Licencias Emitidas en un Período de Tiempo

Se requiere **FECHA INICIO – FECHA FIN**

**Presentar la información ordenada por fecha de emisión.**

* Código de la licencia
* Nombre del conductor
* Tipo de licencia
* Fecha de emisión
* Fecha de vencimiento
* Estado de la licencia

### Reporte de Exámenes Realizados en un Período de Tiempo

Se requiere **FECHA INICIO – FECHA FIN**

**Presentar la información ordenada por fecha de examen.**

* Código del examen
* Nombre del conductor
* Tipo de examen
* Fecha del examen
* Resultado del examen
* Entidad donde se realizó el examen

### Reporte de Infracciones Registradas en un Período de Tiempo

Se requiere **FECHA INICIO – FECHA FIN**

**Presentar la información ordenada por fecha de infracción.**

* Código de la infracción
* Nombre del conductor
* Numero licencia
* Tipo de infracción
* Fecha de la infracción
* Lugar de la infracción
* Puntos deducidos
* Estado de la infracción (pagada o pendiente)

### Reporte Consolidado de Infracciones por Tipo en un Año Determinado

Se requiere **Año**

**Presentar la información ordenada por tipo de infracción.**

* Tipo de infracción (leve, grave, muy grave)
* Cantidad de infracciones
* Puntos totales deducidos
* Multas totales pagadas
* Multas totales pendientes

### Reporte de Conductores con Licencias Vencidas en un Período de Tiempo

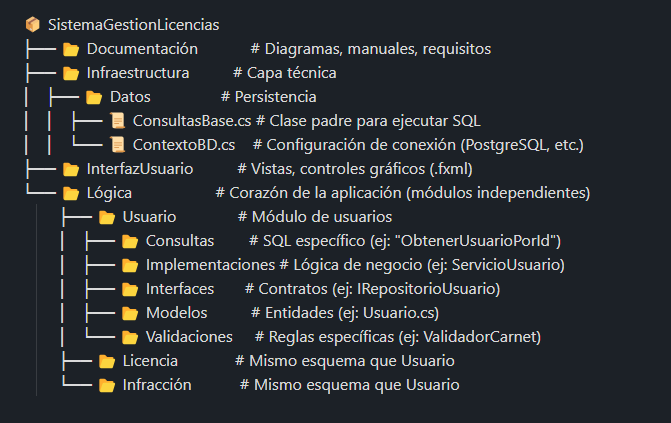
Se requiere **Tiempo que lleva vencida la licencia**

**Presentar la información ordenada por fecha de vencimiento.**

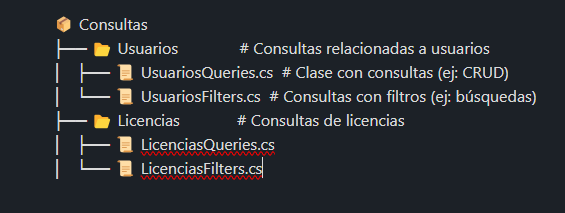
* Nombre del conductor
* Número de documento de identidad
* Número licencia
* Tipo de licencia
* Fecha de vencimiento
* Estado de la licencia

# Arquitectura

## Organización de carpetas

****

## Consultas



### Ejemplo

**// 📜 UsuariosQueries.cs**

**public static class UsuariosQueries**

**{**

**public static string InsertUsuario =>**

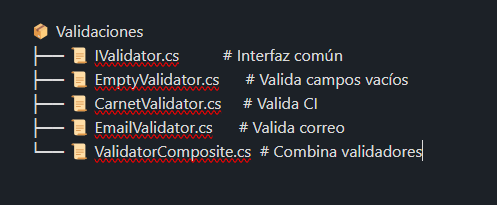
**@"INSERT INTO Usuarios (Nombre, CI) VALUES (@Nombre, @CI)";**

**public static string GetUsuarioById =>**

**@"SELECT \* FROM Usuarios WHERE Id = @Id";**

**}**

## Validaciones



### Ejemplo como usar esta arquitectura

**public class NotEmptyValidator : IValidator**

El ValidatorComposite es un patrón que permite **agrupar múltiples validaciones** y ejecutarlas como si fueran una sola. Es como un "paquete de validadores" que simplifica la lógica cuando necesitas aplicar **varias reglas a un mismo campo** (ej: que un texto no esté vacío **y** que tenga un formato válido).

**var ciValidator = new ValidatorComposite(**

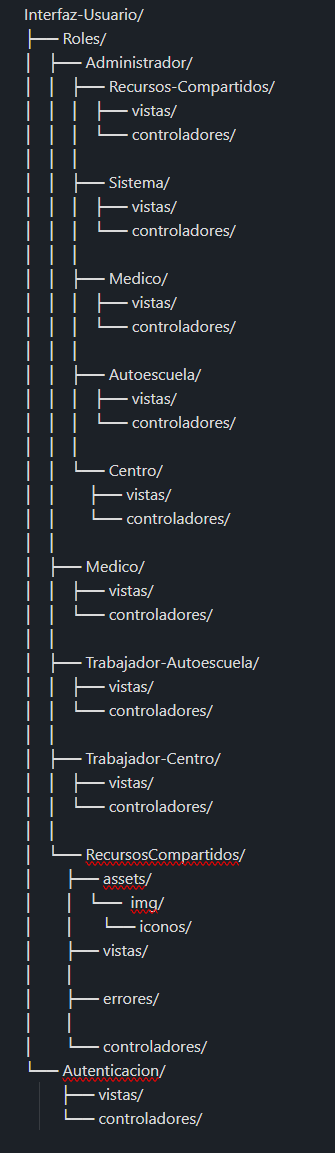
**new NotEmptyValidator(),**

**new NumericValidator(),**

**new LengthValidator(5, 10)**

**);**

## Interfaz Usuario

****

# Extras

### Notaciones

|  |  |
| --- | --- |
| **Pascal\_Snake** | Base datos |
| **PascalCase** | (Clases,atributos,métodos) |
| **kebab-case** | (URLs, carpetas) |