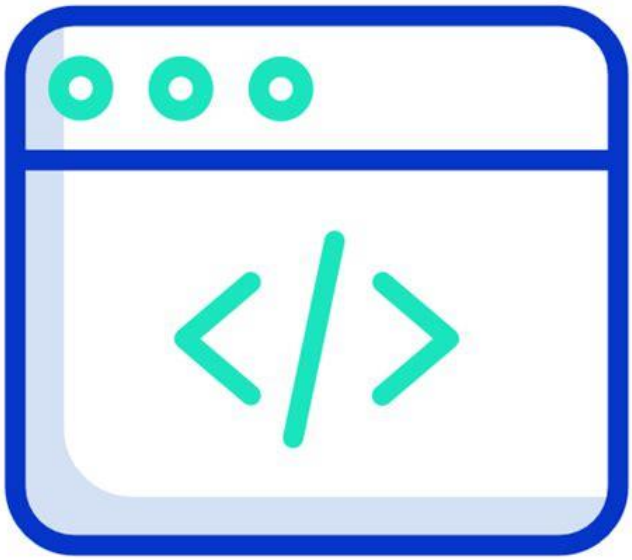


SISTEMA DE CALIFICACIONES

Presentación

Componentes descritos:

Breve presentación del proyecto fin de curso

Adrian Videira Rial

avideira@danielcastelao.com

Índice.

Introducción

En la idea inicial de este proyecto, se plantea una posible solución para un problema en el cual una pequeña escuela quiere comenzar a digitalizar parte de su proceso de administración del centro. En este caso la idea sería poder gestionar las funciones básicas que realiza el profesorado dentro de un centro de enseñanza de manera sencilla y eficaz.

Para lograr que el programa fuera lo más sencillo posible me he basado en un sistema de gestión ya existente y he intentado sintetizar las partes que consideré claves.

En los siguientes apartados se explica de una forma breve cuáles son las funciones que he querido implementar en el programa, su funcionamiento y los problemas con los que me he encontrado.

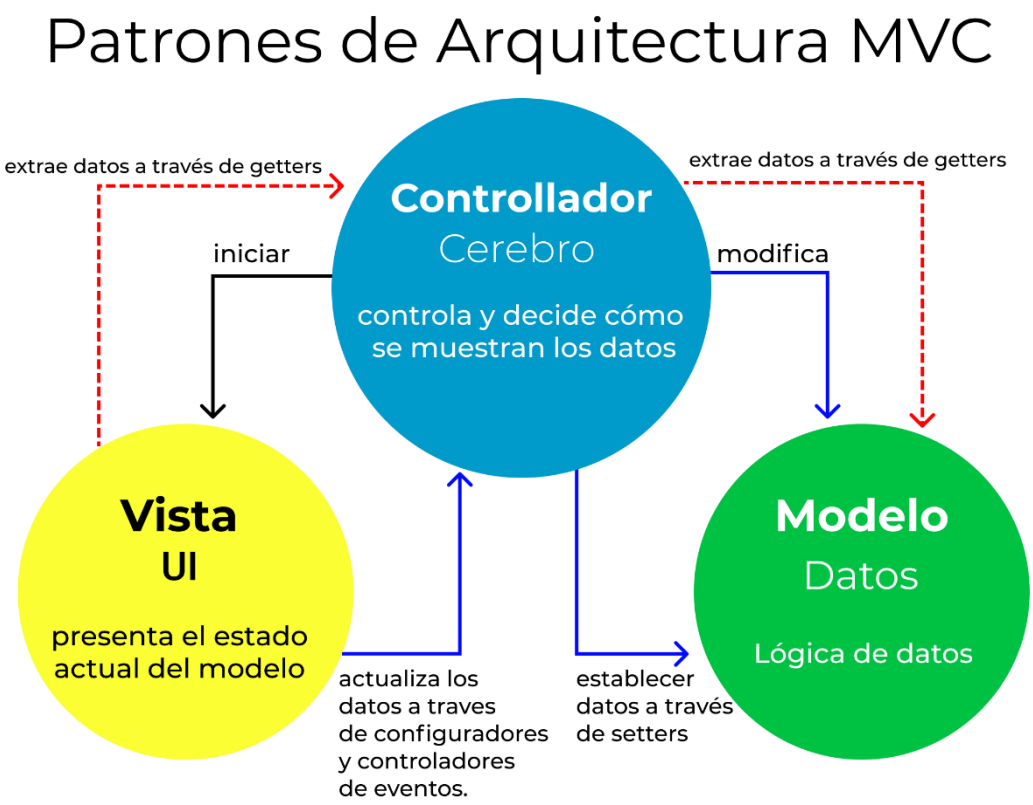
1. Objetivos del proyecto.

El sistema de gestión que se quiere implementar en el centro, deberá principalmente realizar las siguientes funciones:

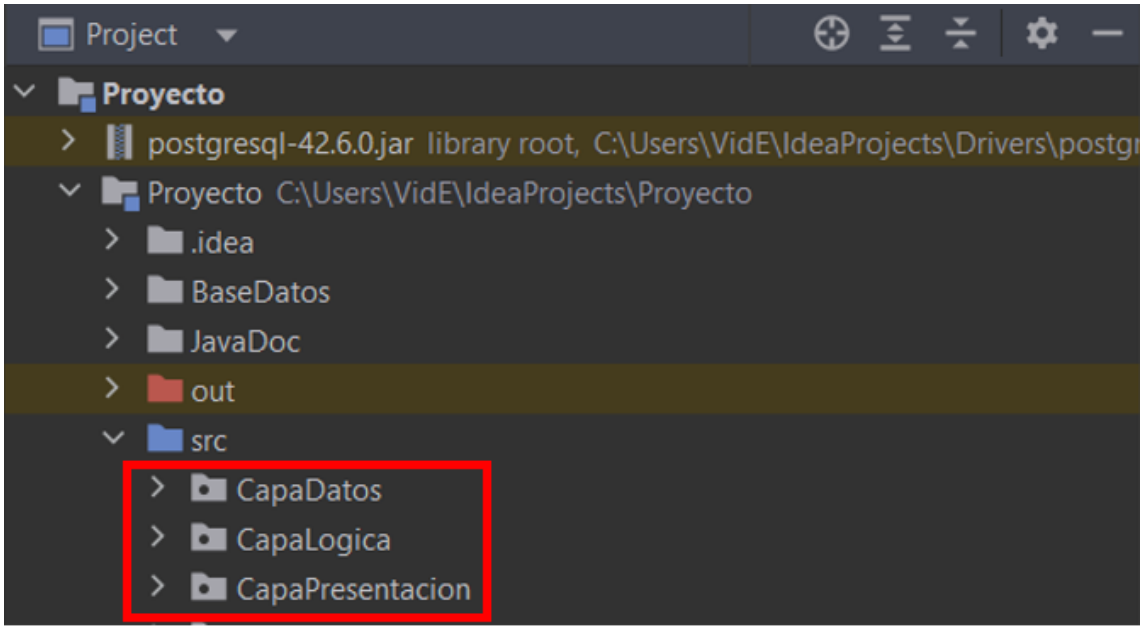
- Permitir el acceso al programa mediante un nombre de usuario y una contraseña, para que solamente puedan acceder a él el personal autorizado.
- Permitir dar de alta y baja a alumnos.
- Realizar el seguimiento de las notas de los alumnos, permitiendo el filtrado de la tabla, por cursos y materias.
- Mostrar la media de cada asignatura.

2. Desarrollo del proceso.

Para realizar esta aplicación me he basado en la arquitectura **MVC** (**M**odelo **V**ista **C**ontrolador) para organizar el código, porque al tener que trabajar con una base de datos, interfaces gráficas y la lógica para darle sentido y coordinar las funciones de la aplicación creo que es la arquitectura más apropiada.



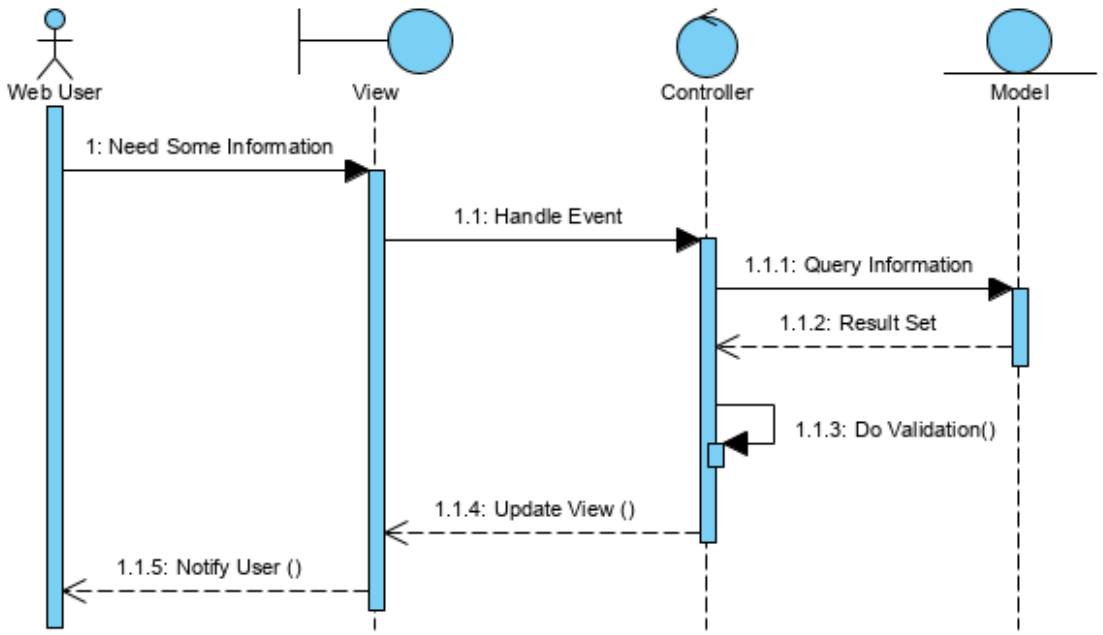
En el caso de esta aplicación está dividida en tres paquetes o (package):



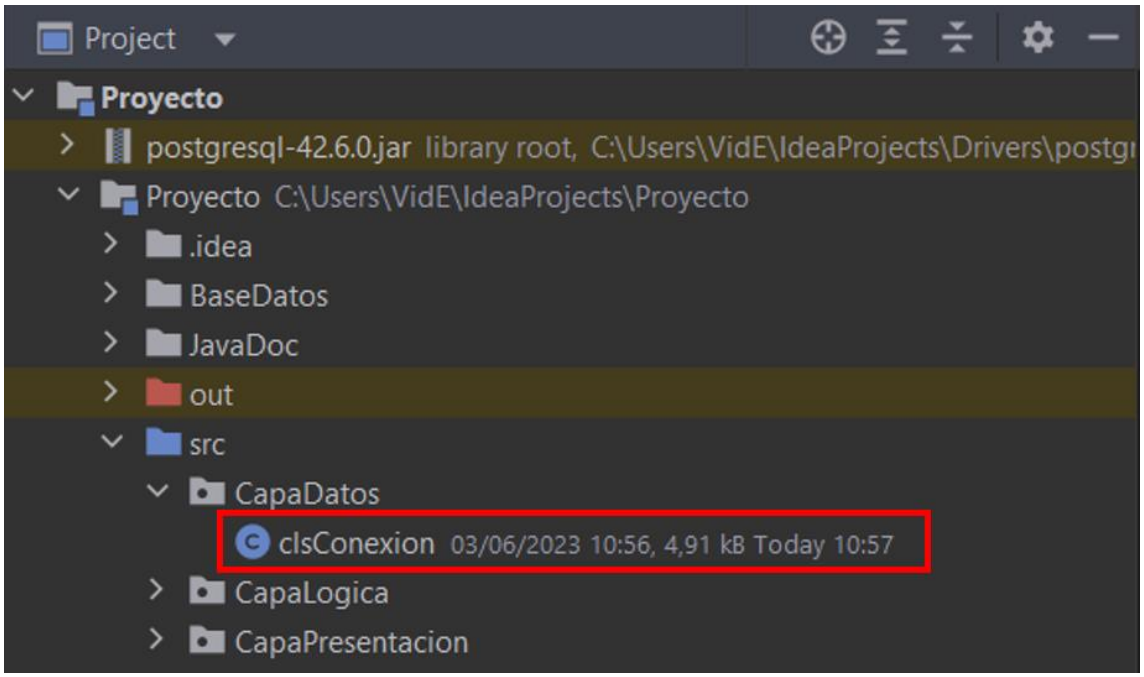
- El paquete "CapaDatos": equivale al **MODELO.**
- El paquete "CapaLógica": equivale al **CONTROLADOR.**
- El paquete "CapaPresentación": equivale a la **VISTA.**

En cada paquete se encuentran las ***clases correspondientes para el manejo de la aplicación***. A continuación, muestro más en detalle el contenido de cada paquete con las funciones que realizan.

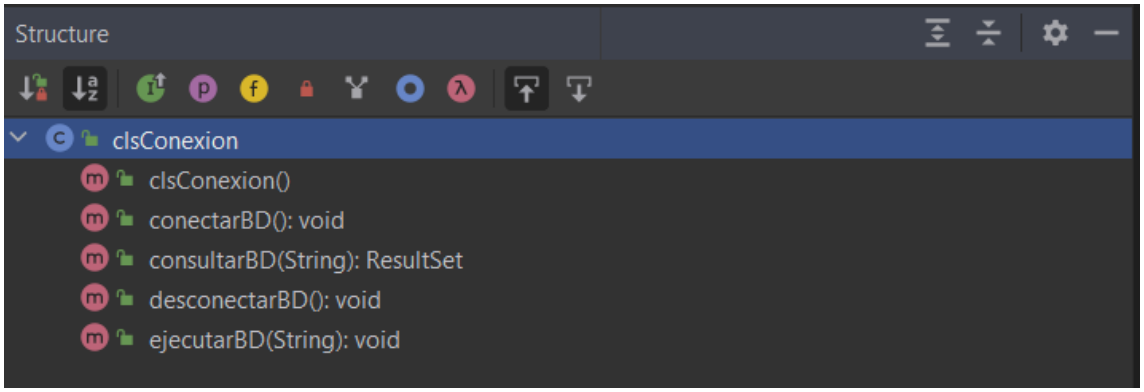
Este sería un ***diagrama genérico*** del funcionamiento de la aplicación con una arquitectura **MCV**.



2.1 Modelo (CapaDatos).



En este paquete nos encontramos con una única clase (**clsConexión**), en esta clase están los métodos que nos permitirán establecer la conexión con nuestro SGBD (Sistema Gestor de Bases de Datos), que en nuestro caso es **Postgres.**



Todos los métodos son públicos para poder ser accedidos desde otros paquetes y clases dentro del programa. Esta clase la usaremos básicamente para poder realizar operaciones **CRUD** sobre nuestra base de datos. Por ejemplo, el método **consultarBD**, tiene la siguiente estructura:

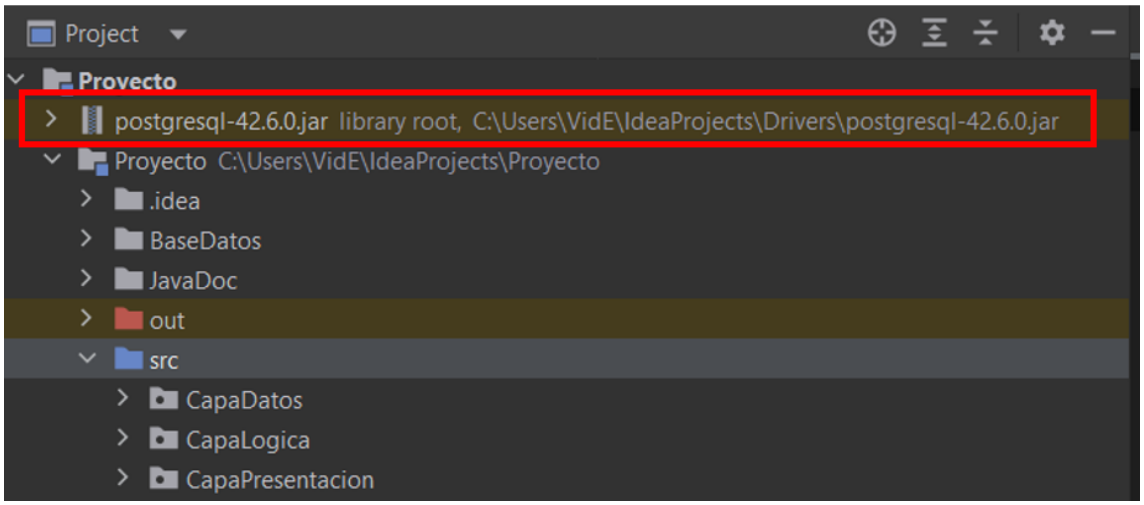
```
Adrian
public ResultSet consultarBD(String sentencia) throws Exception {
    try { //throws Exception: indica que este método debe manejar errores de la clase Exception.
        conectarBD(); // Nos conectamos a la base de datos para poder hacer la consulta con el método que hemos creado.
        pt = conexion.prepareStatement(sentencia); // Usamos el Objeto de la interfaz PreparedStatement para realizar la consulta.
        System.out.println(pt); // Mostramos la consulta por pantalla para poder checkearla.
        return pt.executeQuery();//Usamos este método para que ejecute la consulta.
    }catch (Exception e){
        throw new Exception("Error al ejecutar la consulta");
    }finally { // Este bloque de código "finally" se ejecutará siempre pase lo que pase.
        conexion.close(); // Cerramos la conexión en PostgreSQL.(Optimizamos los recursos del sitema).
    }
}
```

Crearemos un objeto con los parámetros pre-definidos, el cual usaremos para instanciarlo en nuestra capa lógica y conectarnos a la base de datos.

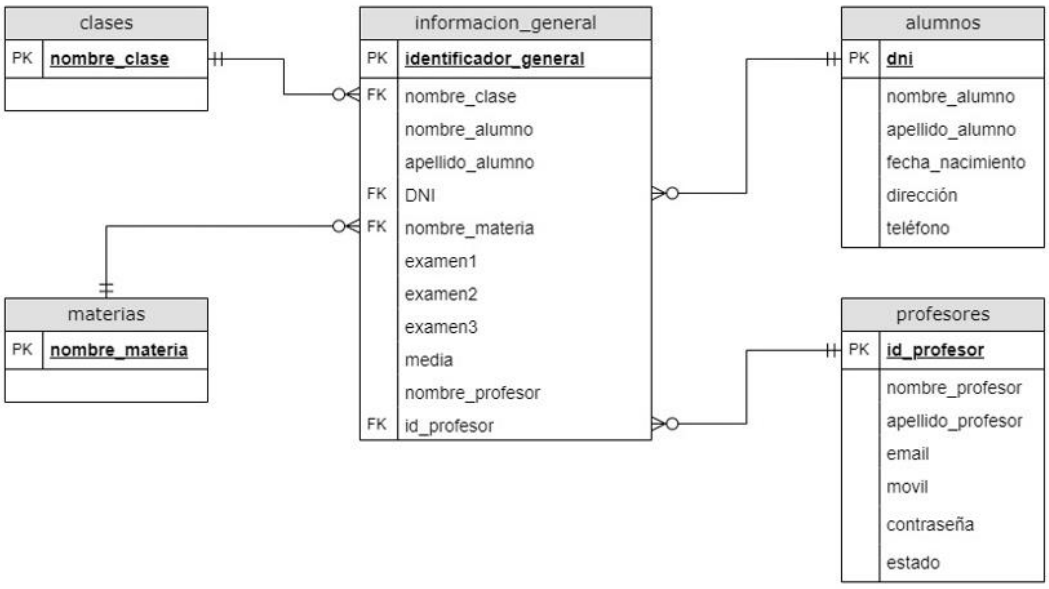
```
public clsConexion(){ // Objeto en el que predefinimos los datos necesarios para establecer la conexión.
    usuario = "postgres"; // Usuario por defecto en PostgreSQL
    contrasena = "adrian"; // Contraseña para acceder a PostgreSQL
    url = "jdbc:postgresql://localhost:5432/Escuela"; //Ruta para conectarnos a nuestra BD "Escuala".
    driver = "org.postgresql.Driver";
    conexion = null;
}
```

Por otro lado, para establecer la conexión entre Java y el SGBD, necesitamos el "driver" o controlador **JDBC** que haga de intérprete entre Java y Postgres. Para ello nos descargamos de la página oficial de Postgres dicho controlador y lo agregamos a nuestro proyecto.

https://jdbc.postgresql.org/



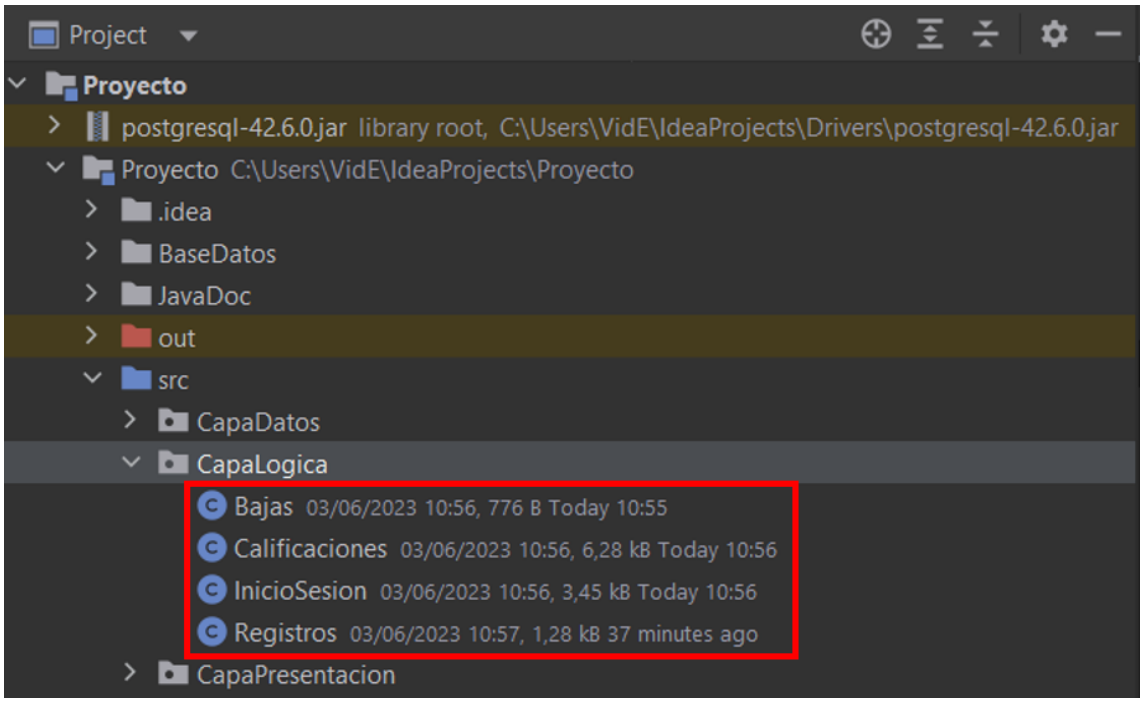
La base de datos sobre la que vamos a trabajar, se llama "Escuela" y su diseño lógico:



2.2 Controlador (CapaLógica).

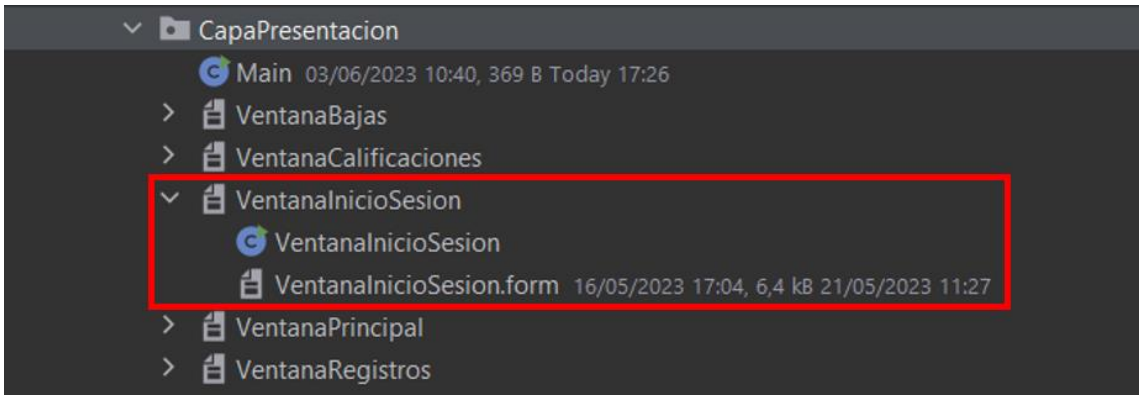
Una vez tenemos definida la capa de datos junto con nuestra base de datos, procedemos a crear la lógica que se encargará, como ya hemos dicho previamente, de **comunicarse entre la vista** (capaPresentación) **y el modelo** (capaDatos).

En este paquete, nos podemos encontrar las siguientes clases:

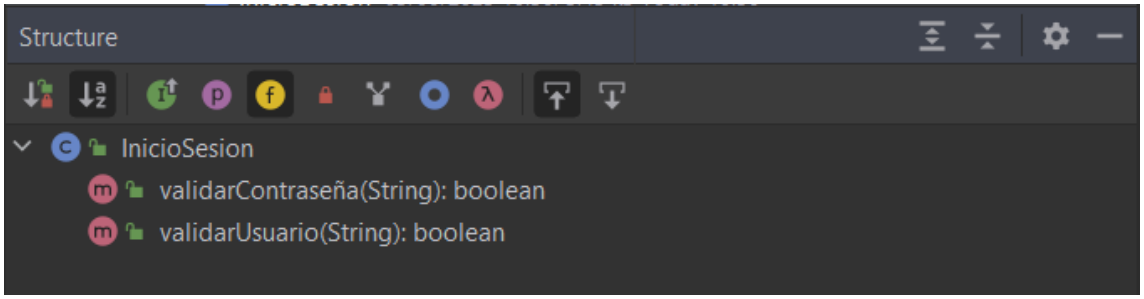


A continuación, vamos a explicar brevemente la función que cada una de estas clases tiene dentro del programa.

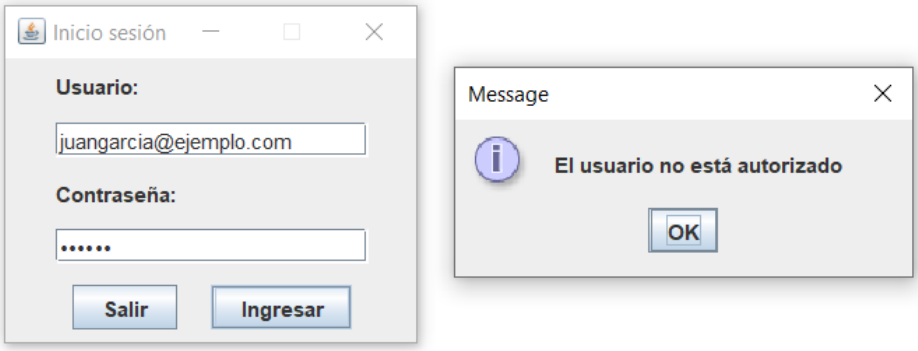
- **InicioSesion:** Esta clase se comunica con la base de datos y con la clase **VentanaInicioSesion.**



Su función será validar el nombre de usuario y la contraseña introducidas para dar acceso al programa. En caso de que un usuario no esté dado de alta en la base de datos, no podrá acceder al programa. Realiza la validación en la tabla "**public.profesores**" de la base de datos.



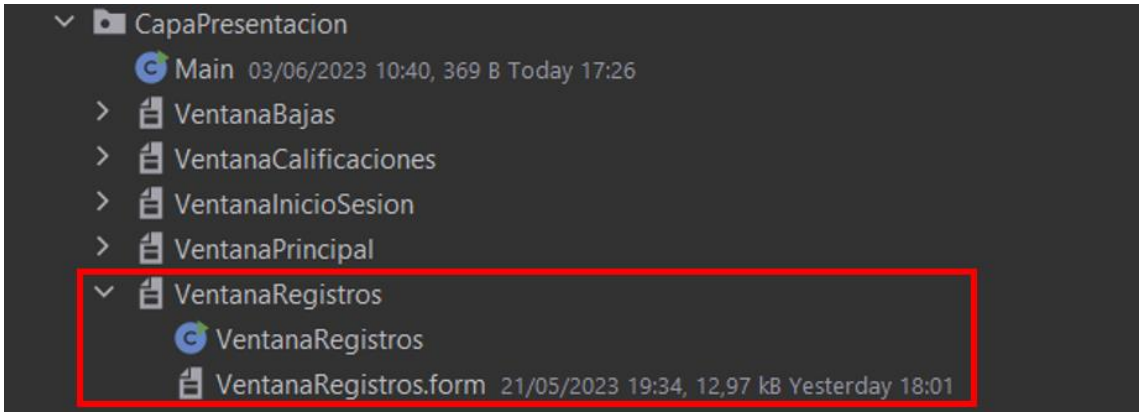
Esto lo realizará a través de los métodos **validarContraseña()** y **validarUsuario().**



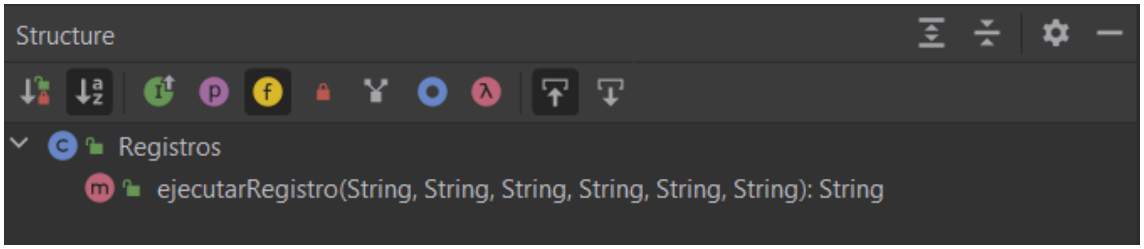
En este ejemplo el usuario no se encuentra dado de alta en la base de datos, de ahí el mensaje de error.

	id_profesor [PK] numeric (3)	nombre_profesor character varying (15)	apellido_profesor character varying (30)	email character varying (30)	movil numeric (9)	contraseña numeric (6)	estado boolean
1	111	Maria	Rodriguez	mariarodriguez@ejemplo.com	[null]	123456	true
2	222	Noelia	Alonso	noeliaalonso@ejemplo.com	[null]	123456	true

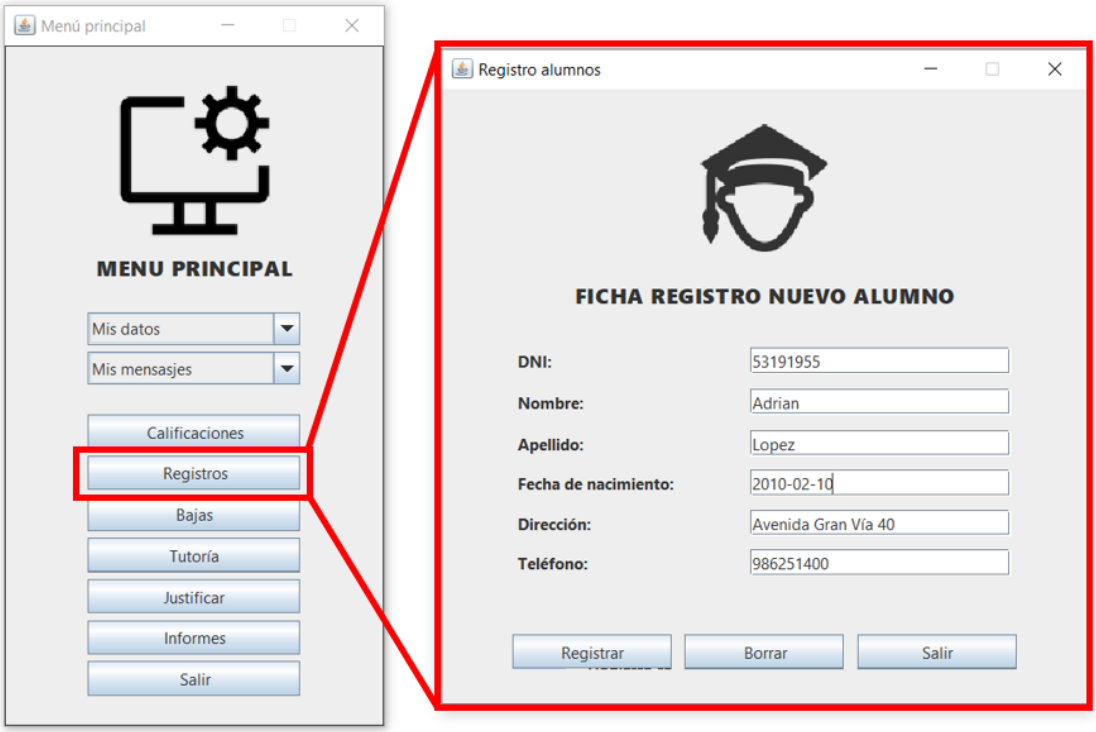
- **Registros:** Se comunica con la base de datos y con la clase **VentanaRegistros.**



Esta clase tiene un solo método el cual se encarga de ejecutar una sentencia del tipo INSERT INTO…para añadir un nuevo alumno a la tabla "**public.alumnos**" de la base de datos.



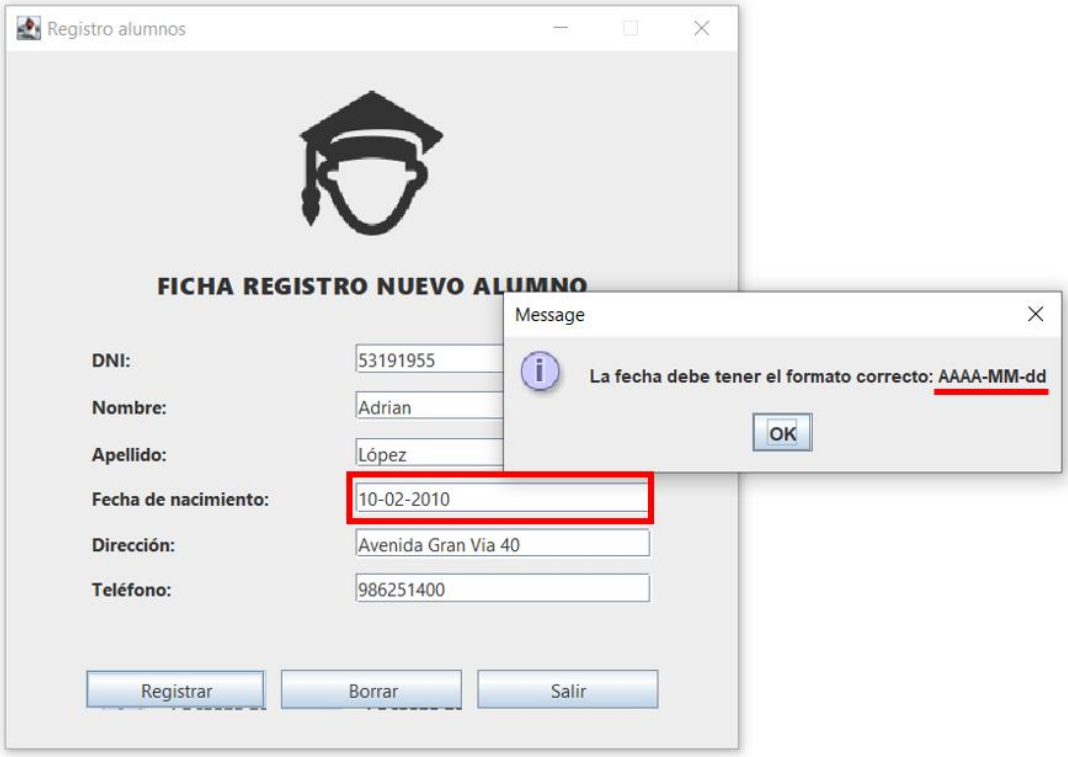
Esto lo realizará a través del método **ejecutarRegistro()**, que recibe por parámetro los los datos rellenados en el formulario de la **VentanaRegistros.**



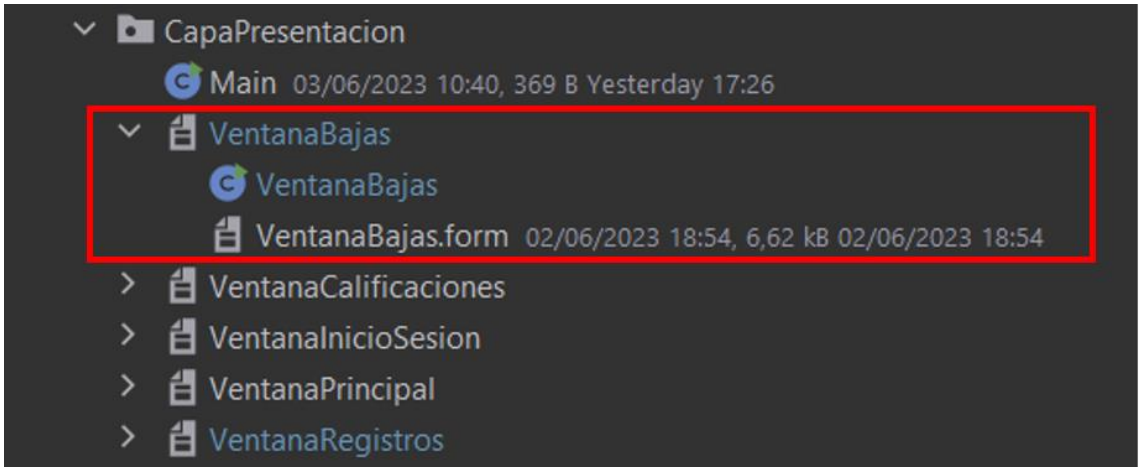
Una vez introducidos los datos correctamente en el formulario se pulsa el botón **Registrar** y si están en el formato correcto se insertará una nueva fila en la tabla "**public.alumnos"** de la base de datos.

	dni [PK] numeric (8)	nombre_alumno character varying (15)	apellido_alumno character varying (30)	fecha_nacimiento date	direccion character varying (50)	telefono numeric (9)
8	89012345	Marta	Fernández	2011-02-18	Plaza 159	986890123
9	90123456	David	Ramírez	2011-03-20	Ronda 357	986901234
10	10234567	Elena	Torres	2011-04-22	Boulevard 246	986012345
11	53191955	Adrian	Lopez	2010-02-10	Avenida Gran Vía 40	986251400

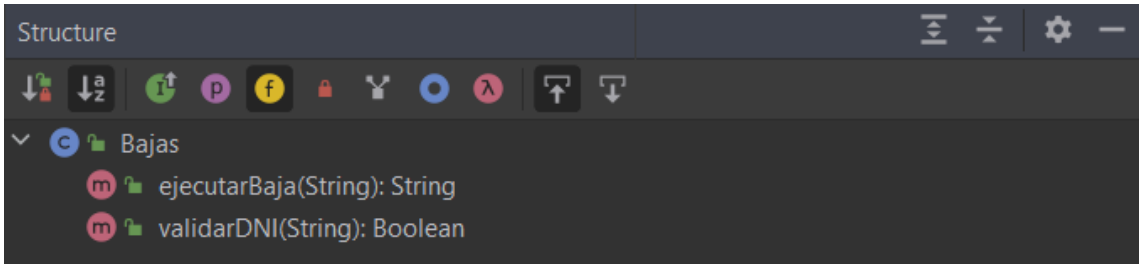
En caso de que alguna casilla este sin cubrir o no tenga un formato apropiado, saltará una ventana avisando del error.



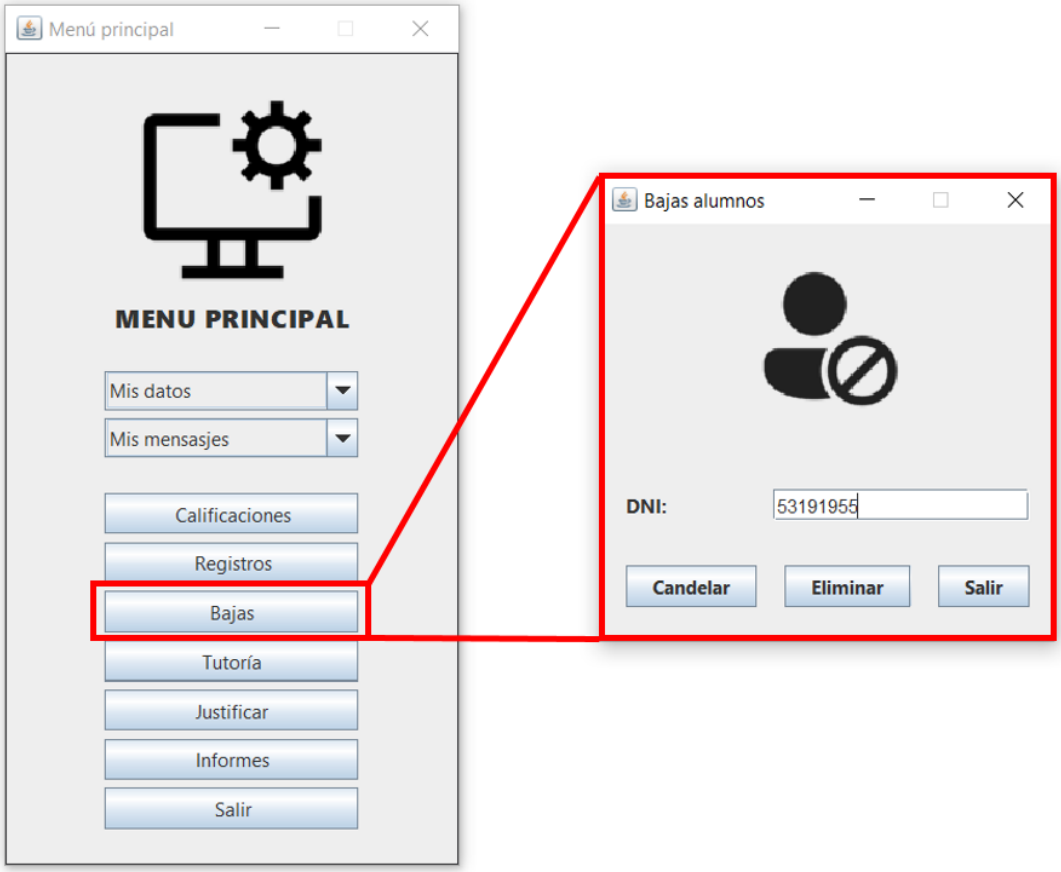
- **Bajas:** Esta clase se comunica con la base de datos y con la clase **VentanaBajas.**



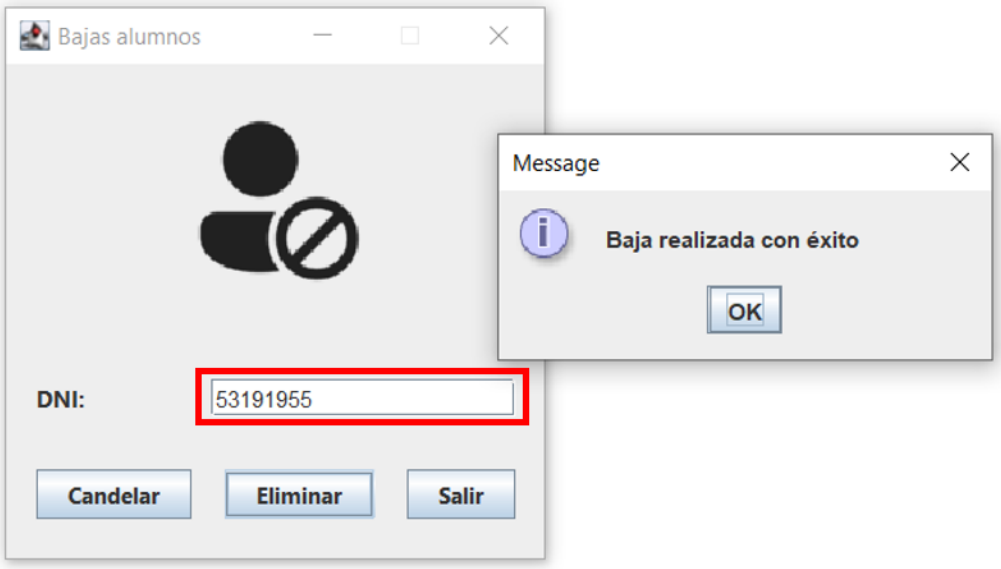
Esta clase se encargará de eliminar el registro de la tabla "**public.alumnos**" que contenga el DNI espeficicado. Lo hará con el método **ejecutarBaja()**. El método validarDNI servirá para verificar que el DNI introducido existe en la base de datos.



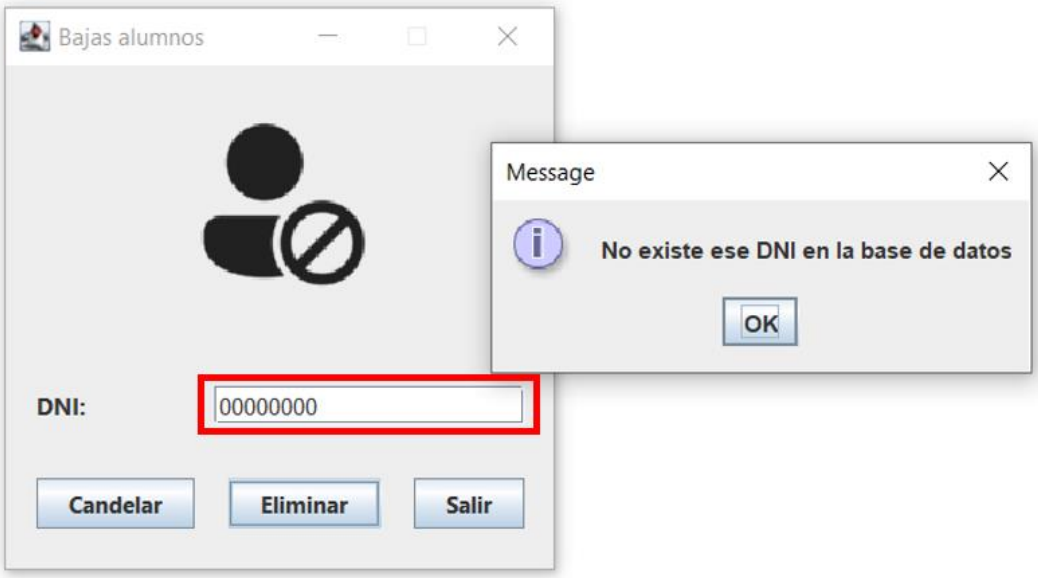
Para eliminar un registro, se pulsa el botón "**Bajas**" en el menú principal y en la ventana emergente, se introduce el dni del registro a eliminar.



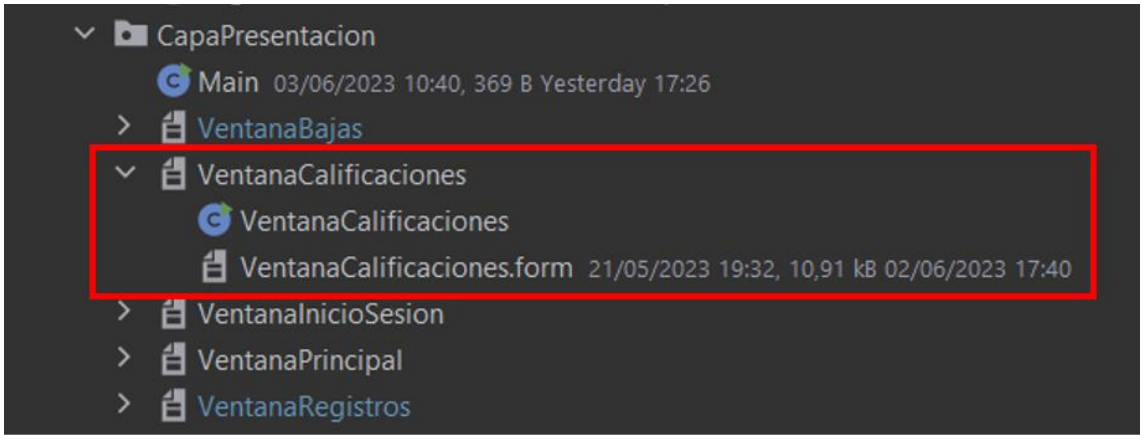
Si el número de DNI se encuentra en la tabla alumnos, procederá a eliminar el registro y enviará un aviso de confirmación.



En caso de que el DNI introducido no conste en la base de datos, nos dará un aviso como este:

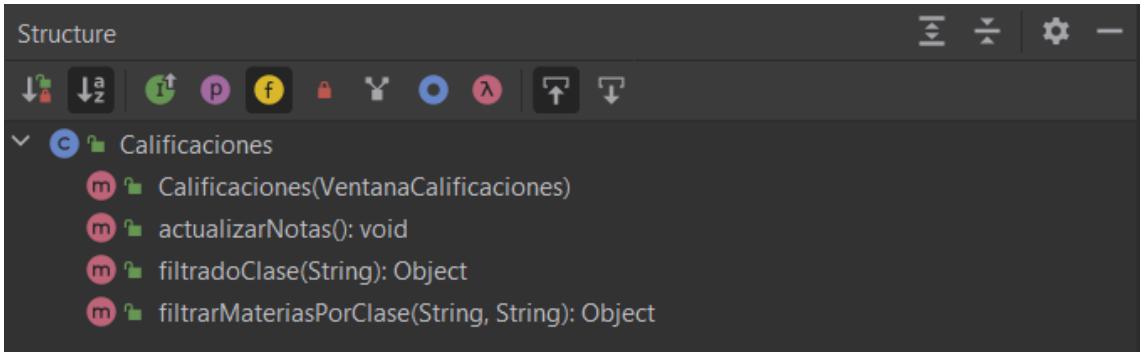


- **Calificaciones:** Esta clase se comunica con la base de datos y con la clase **VentanaCalificaciones.**



Su función será acceder a la tabla "**public.información_general**" de la base de datos y mostrar los alumnos correspondientes al curso y la materia filtrada por el usuario. Una vez los datos están filtrados, se mostrarán en una tabla, la cual tiene 3 columnas llamadas "examen1", "examen2", "examen3", que son editables y permiten al profesor introducir las notas de cada alumno.

Lugo en la columna "Media" se mostrará la media de estos tres exámenes correspondientes a cada evaluación que dará na nota final la "Media".



Esto lo realizará a través de los métodos **filtradoClase()**, **filtradoMateriasPorClase()** y **actualizarNotas()**.

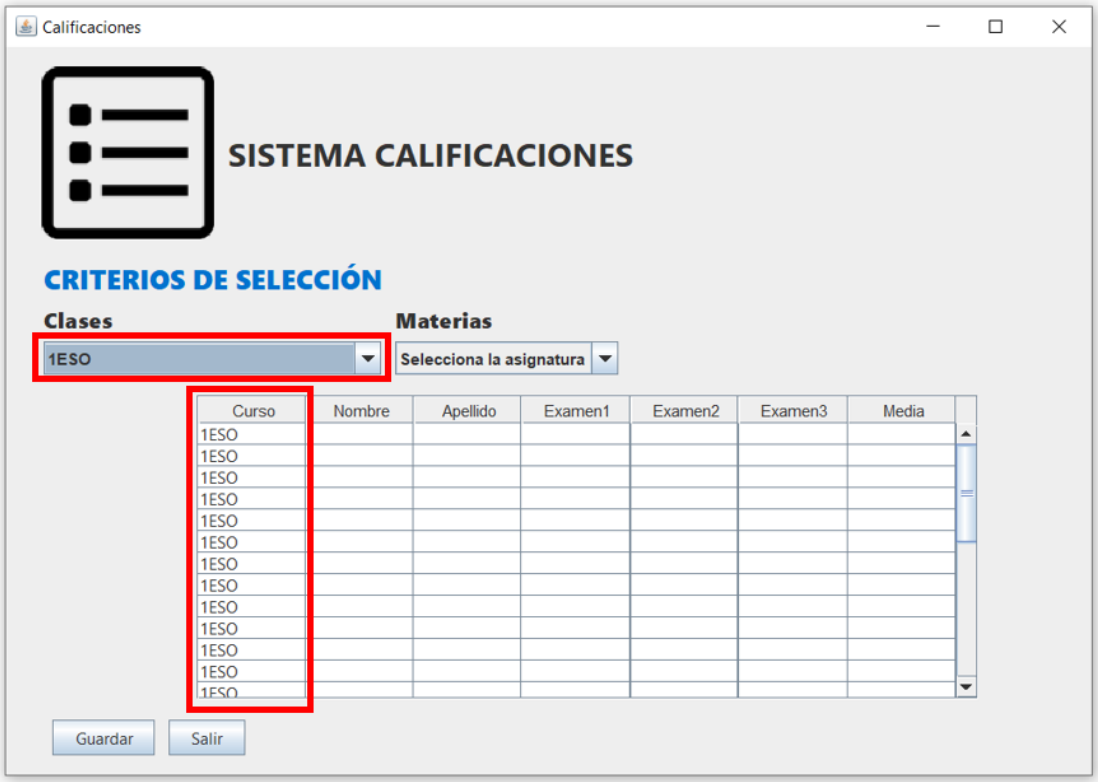
El método **filtradoClase()**, es el encargado de seleccionar todos los registros de la tabla información_general que cumplan el criterio seleccionado por el usuario.

El método **filtrarMateriasPorClase()**, se encarga del segundo filtro para seleccionar los datos que se mostrarán en la tabla, en este caso seleccionara los alumnos de la materia seleccionada por el usuario y que además estén en el curso previamente seleccionado.

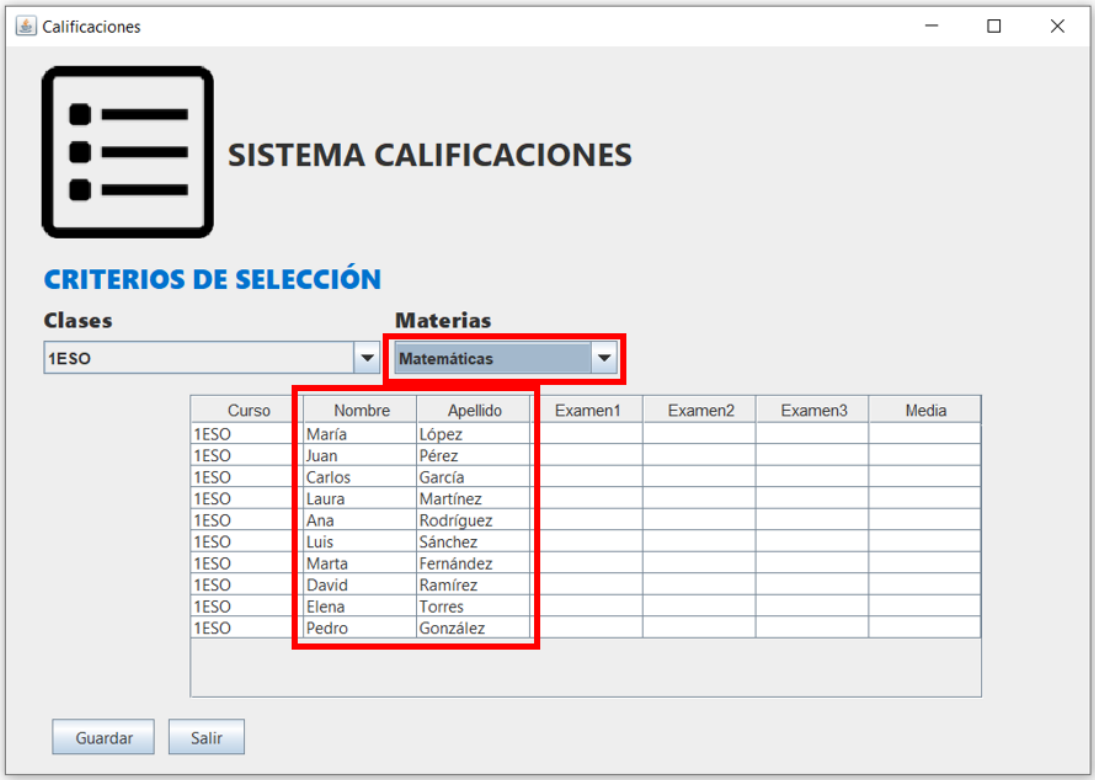
El método **actualizarNotas()** será el encargado de una vez puestas las notas en la tabla por el usuario actualizar la base de datos con dicha información.

Sí pulsamos en el botón “Calificaciones”, se nos abre la ventana en la cual vamos trabajar.

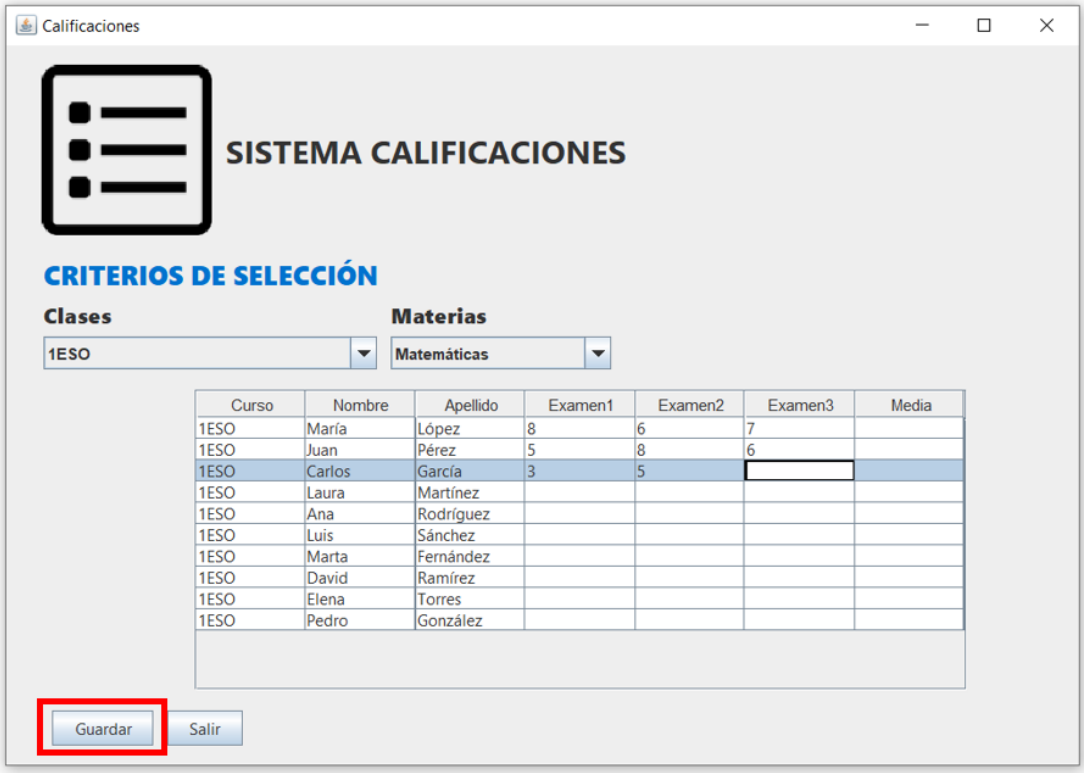
1º Filtrado: por curso.



2º Filtrado: por materia.



3º Puesta de notas a cada alumno.



Una vez cubiertas las celdas, se pulsaría en el botón guardar, para enviar los dato y actualizar la tabla información_general de la base de datos.

4. Problemas y errores.

Los principales problemas los he encontrado en la **VentanaCalificaciones**, no he sido capaz de que los datos introducidos en la tabla de la interfaz gráfica fueran insertados en la tabla de la base de datos, quedando así este apartado del programa sin acabar.

5. Conclusiones personales y proyecto a futuro.

5.1 Conclusiones personales

Un sistema de calificaciones o cualquier otro sistema de gestión son proyectos grandes que abarcan gran cantidad de elementos y funcionalidades, con este pequeño proyecto he intentado replicar una pequeña parte de uno de estos sistemas de gestión.

Dado mi nivel principiante en la programación, el tiempo personal del que disponía para realizar este trabajo y el resto de tareas del curso saco las siguientes conclusiones:

Por un lado, he hecho lo mejor que he podido con lo que he aprendido y el tiempo de que disponía.

También creo que ha sido una buena experiencia para ver en conjunto cómo funciona lo aprendido a lo largo del curso.

Por otro lado, ha sido algo frustrante el no haber podido conseguir que el programa funcionara de una forma correcta en su totalidad.

5.2 Proyecto a futuro.

En este proyecto creo que se debería mejorar las siguientes funcionalidades:

- Lograr que la ventantaCalificaciones funcione correctamente.
- Que cada profesor solo pueda acceder a los cursos y materias que tiene asignados dentro del centro educativo, para evitar posibles errores ya que cada profesor solo tendrá acceso a sus datos.
- En la base de datos se deberían implementar varios triggers los cuales una vez añadido o eliminado un dato lo actualicen en el resto de tablas para que se muestre.
- En la ventana registro, se debería de añadir el curso en el cual se matricula el alumno.