Proyecto Final

- Programación

Francisco Javier Soler Aguado
1º DAM

Indice

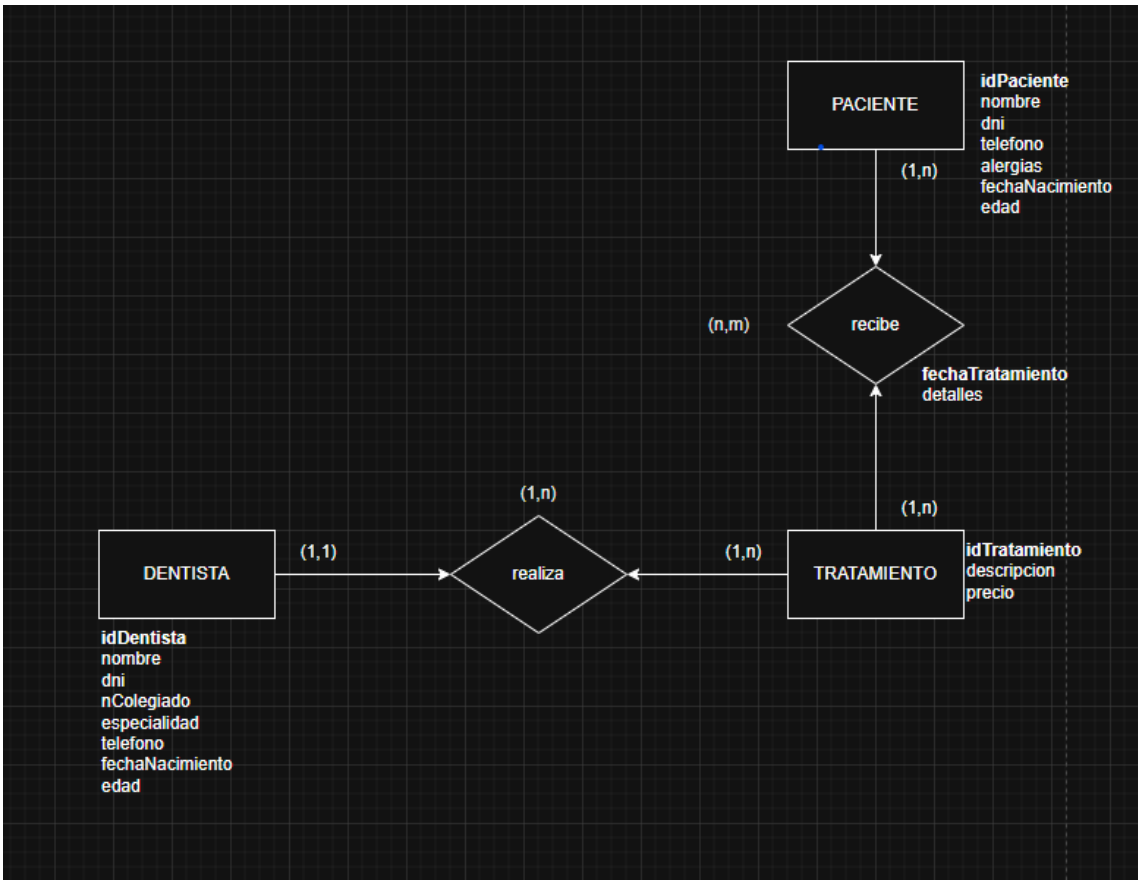
1. Fase de Diseño

1.1 Descripción del proyecto

Este proyecto consta de un programa de una Clínica Dental que quiere almacenar y gestionar todos los dentistas que trabajan para la clínica, todos los pacientes que se le hayan aplicado o no tratamientos, y los tratamientos que realiza esta clínica dental junto con sus dentistas especializados en ellos.

Cada paciente puede recibir tratamientos, a esto se le nombre Tratamientos Paciente en el código y la base de datos.

1.2 Modelo Entidad Relación



En este se muestran las entidades principales de mi proyecto que son: Dentista, Paciente y Tratamiento.

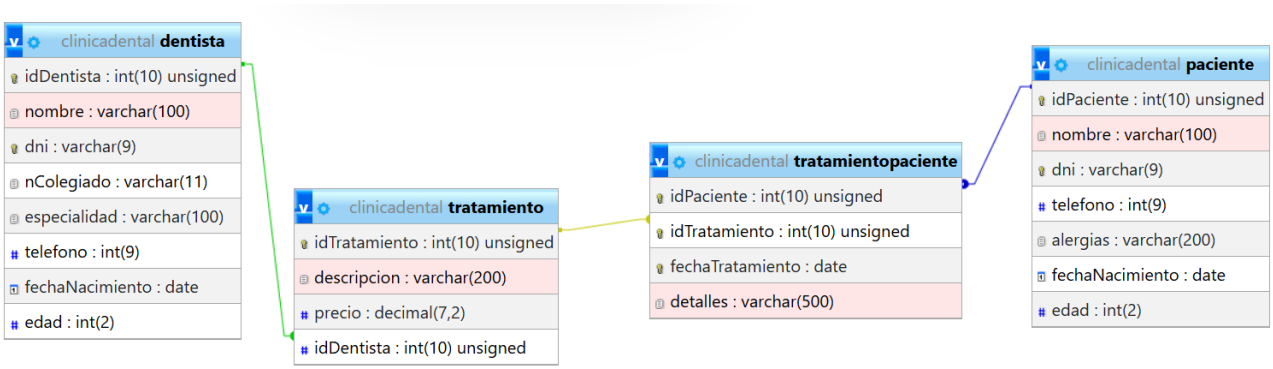
Entre otras cosas se muestran los atributos de cada clase:

- Dentista: idDentista, nombre, dni, nColegiado, especialidad, telefono, fechaNacimiento y edad.

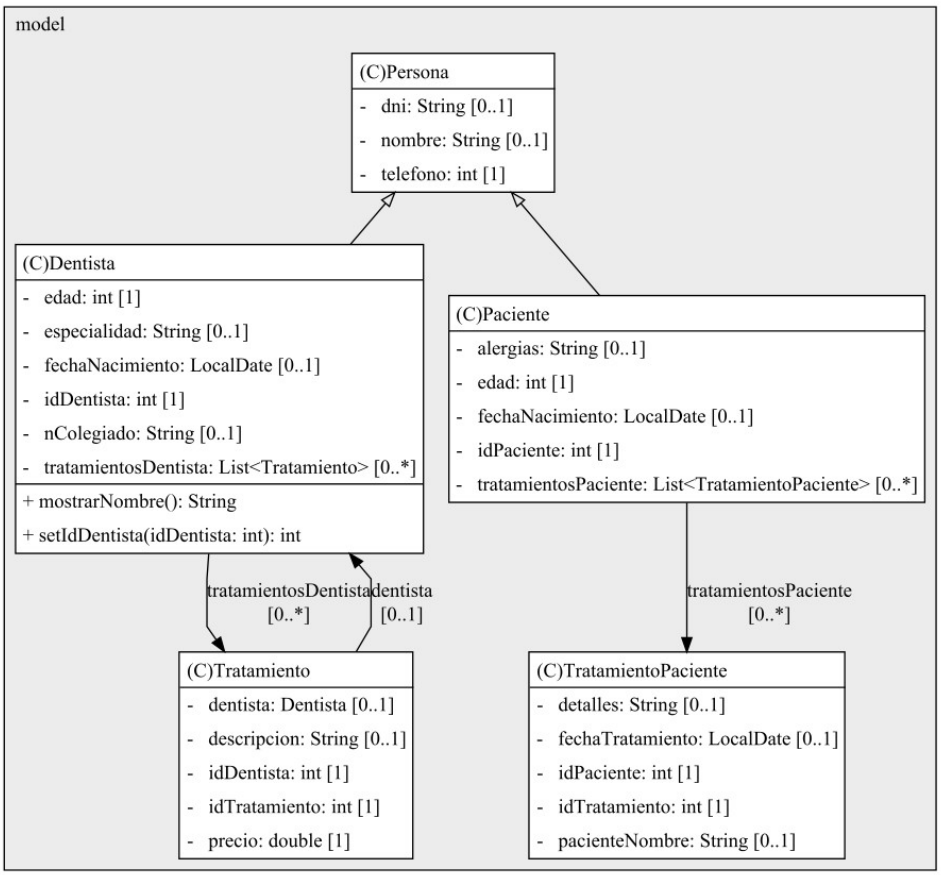
- Paciente: idPaciente, nombre, dni, telefono, alergias, fechaNacimiento y edad.
- Tratamiento: idTratamiento, descripción y precio.

En la relación recibe aparece una tabla en medio que relaciona las entidades Paciente y Tratamiento esta entidad almacena: fechaTratamiento y los detalles.

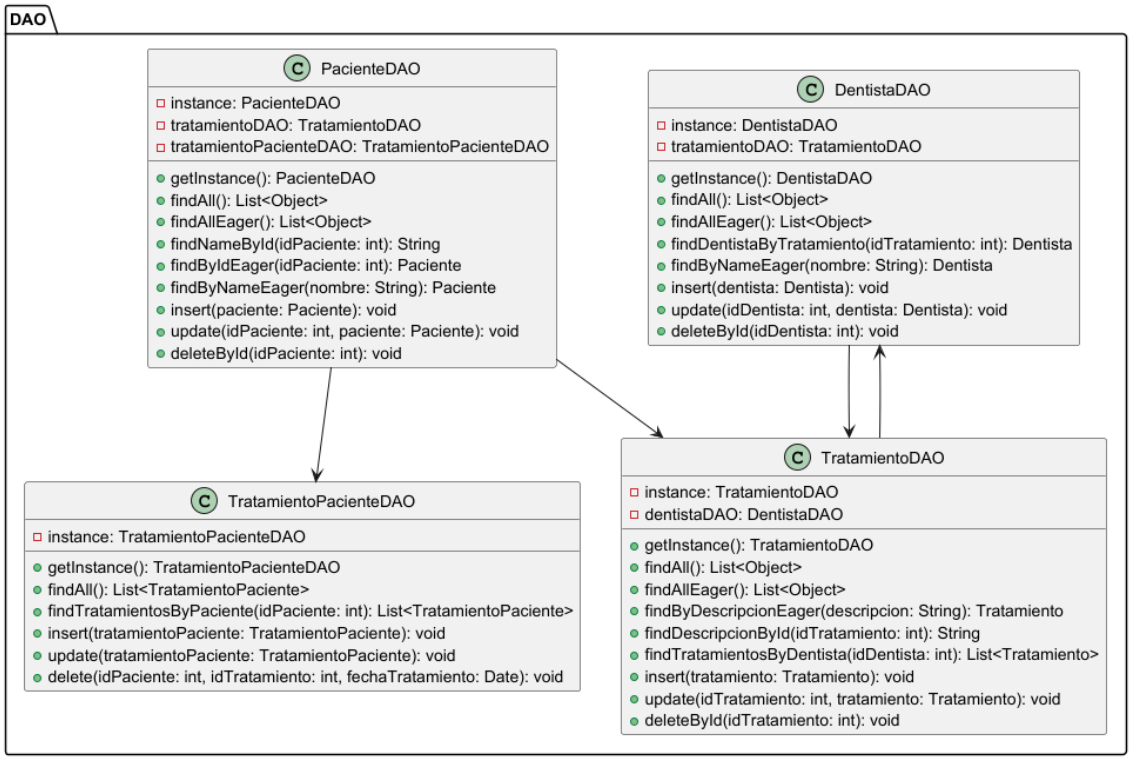
1.2.1 Archivo SQL de la base de datos



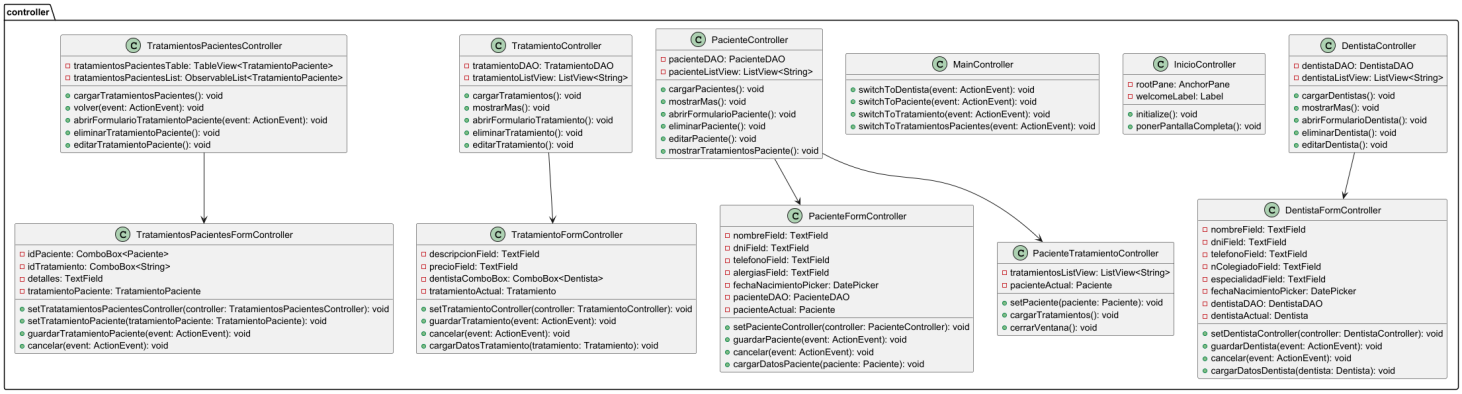
1.3 Diagrama de clases



Este es el diagrama de clases del paquete model generado a partir del código, en este se muestra lo visto anteriormente en el entidad-relación pero adaptado al código en java.

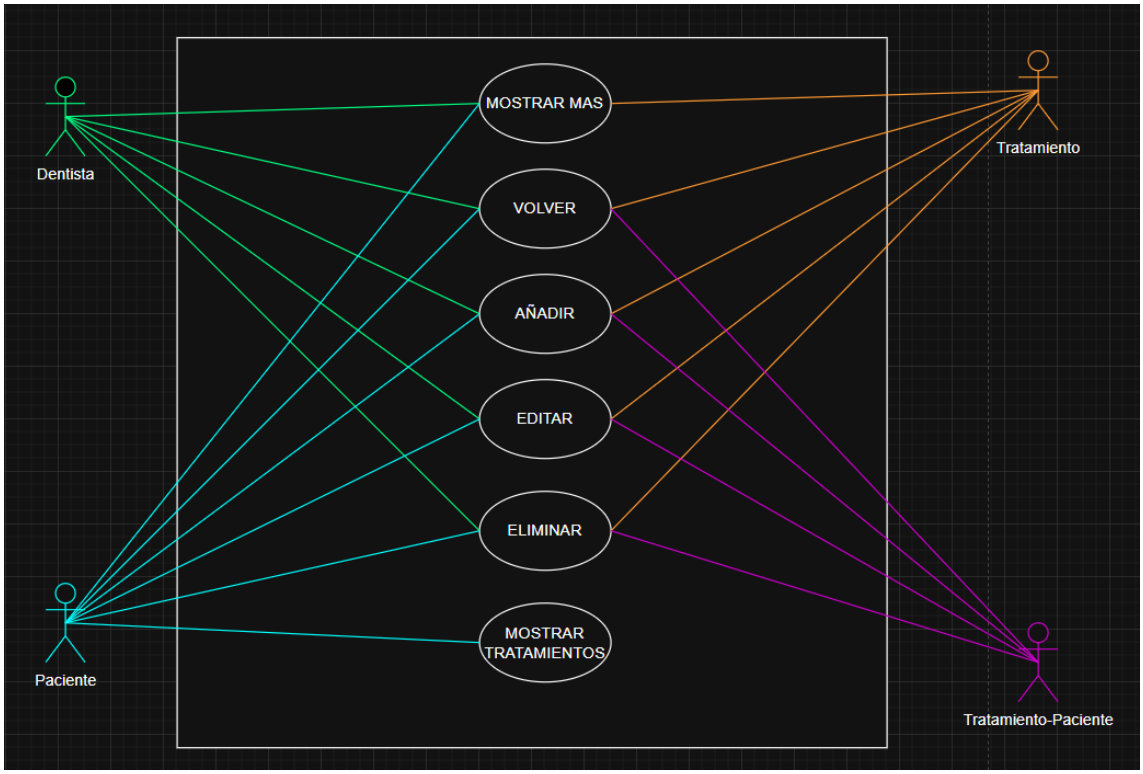


Este es el diagrama de clases del paquete DAO (encargado de las consultas de la base de datos), en él vemos los métodos utilizados en cada DAO que han sido utilizados en el programa final.



Este diagrama de clases pertenece al paquete controller, donde se encuentran los métodos necesarios en la interfaz gráfica, en estas clases es donde se hace uso de los métodos del paquete DAO.

1.4 Diagrama de Casos de Uso

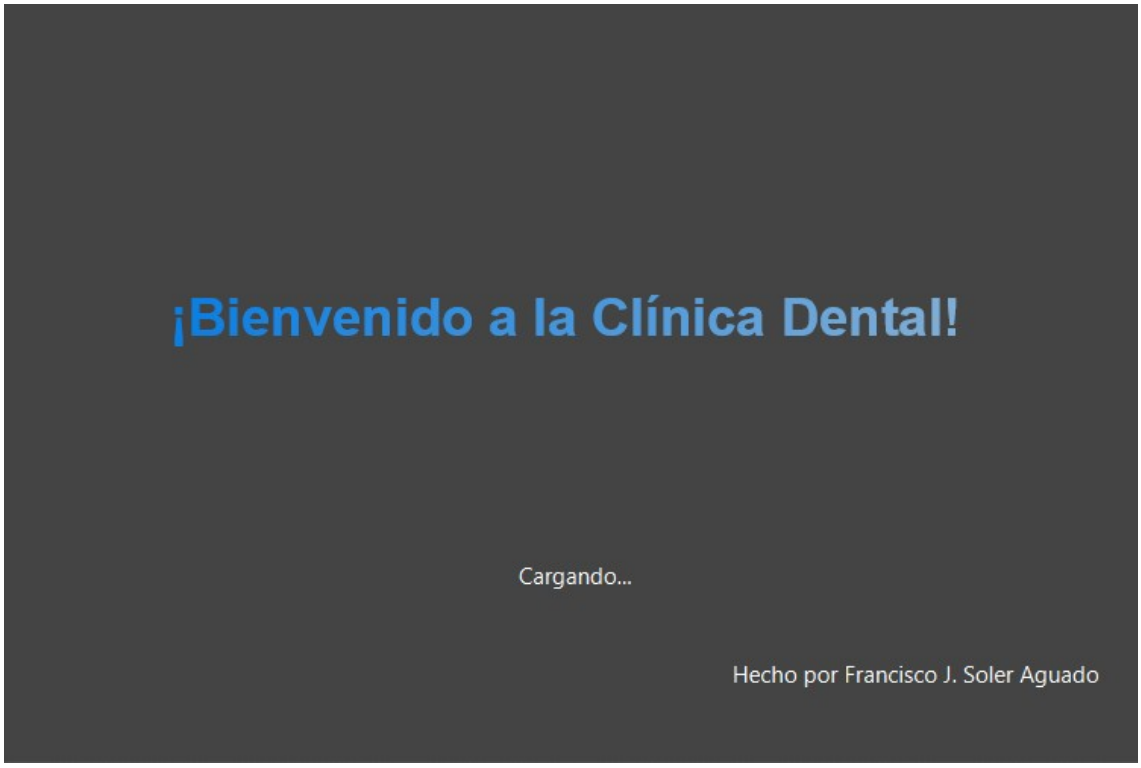


En este diagrama se muestra, las opciones que tiene cada menú del programa, según cada menú tendrás estas opciones al usar el programa:

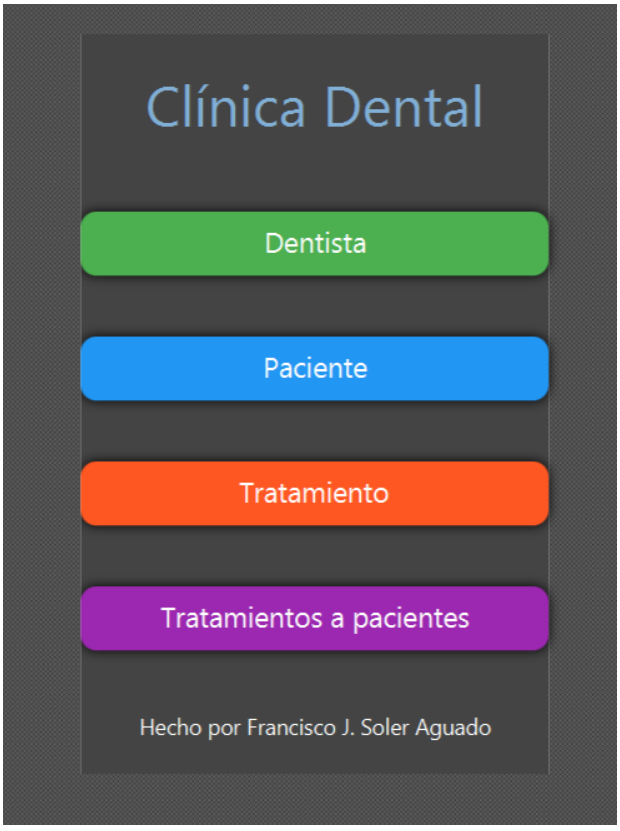
- Dentista: Añadir, Editar, Eliminar, Mostrar Mas y Volver.
- Paciente: Añadir, Editar, Eliminar, Mostrar Mas, Mostrar Tratamientos y Volver
- Tratamiento: Añadir, Editar, Eliminar, Mostrar Mas, Editar y Volver
- Tratamiento-Paciente: Añadir, Editar, Eliminar y Volver.

1.5 Diseño de pantallas

Inicio.fxml

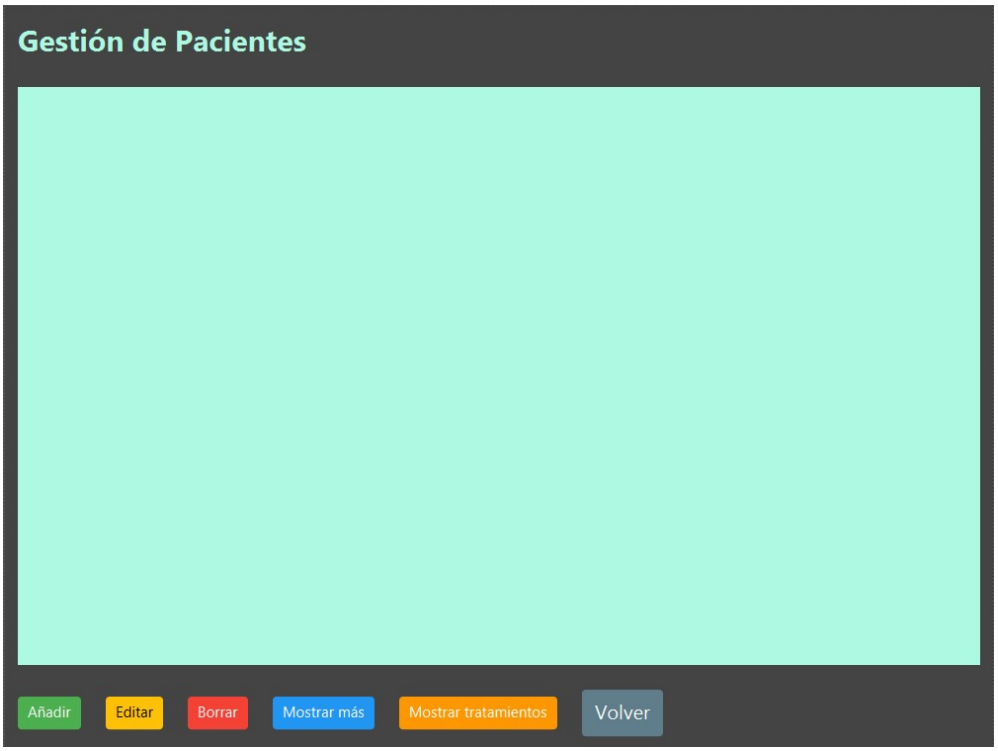


main.fxml



dentista.fxml

paciente.fxml



tratamiento.fxml

tratamientosPacientes.fxml

pacienteEditForm.fxml (esta pantalla se usa varias veces para cada formulario ya sea de añadir o editar adaptada a cada clase y sus atributos)

2. Fase de Desarrollo

En el desarrollo del proyecto, primero me encargue de tener una idea de lo que quería realizar, para ello primero diseñé el modelo entidad-relación, a partir de esa base, escribí las primeras líneas de código, lo primero fue crear las clases básicas y sus atributos en código para tener una idea de como plasmar la idea en el código.

Una vez llegado a este punto creé la base de datos del proyecto y la conecté con el programa, diseñe el diagrama de casos de uso para conocer que consultas a la base de datos serán necesarias, añadí el paquete DAO a mi proyecto y empezé a implementar las consultas necesarias en métodos que posteriormente serán llamados desde el paquete controller en la interfaz gráfica.

Lo ideal hubiera sido crear primero de todo las interfaces, pero como no disponía de conocimientos de JavaFX tuve que hacerlo en este orden.

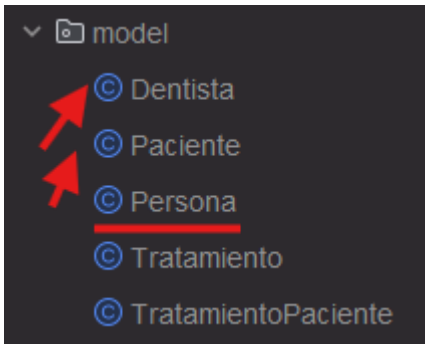
Cuando ya tuve todo el proyecto acabado prácticamente implementé la interfaz gráfica y su paquete controller, los archivos fxml etc...

2.1 URL del repositorio del proyecto en GitHub

FranciscoSolerAguado/ClinicaDental: Proyecto final de programacion 1ºDAM de una clinica dental, en java

3. Lista de Comprobación

3.1 ¿Cómo se usa la herencia?



La herencia se usa con la clase Persona la cual tiene unos atributos únicos que heredan Paciente y Dentista, estos atributos son:

- Nombre
- Dni
- Teléfono

```
public class Persona {  2 usages  2 inheritors  FranciscoSolerAguado
    private String nombre;  4 usages
    private String dni;  4 usages
    private int telefono;  4 usages

    public Persona(String nombre, String dni, int telefono) {
        this.nombre = nombre;
        this.dni = dni;
        this.telefono = telefono;
    }
```

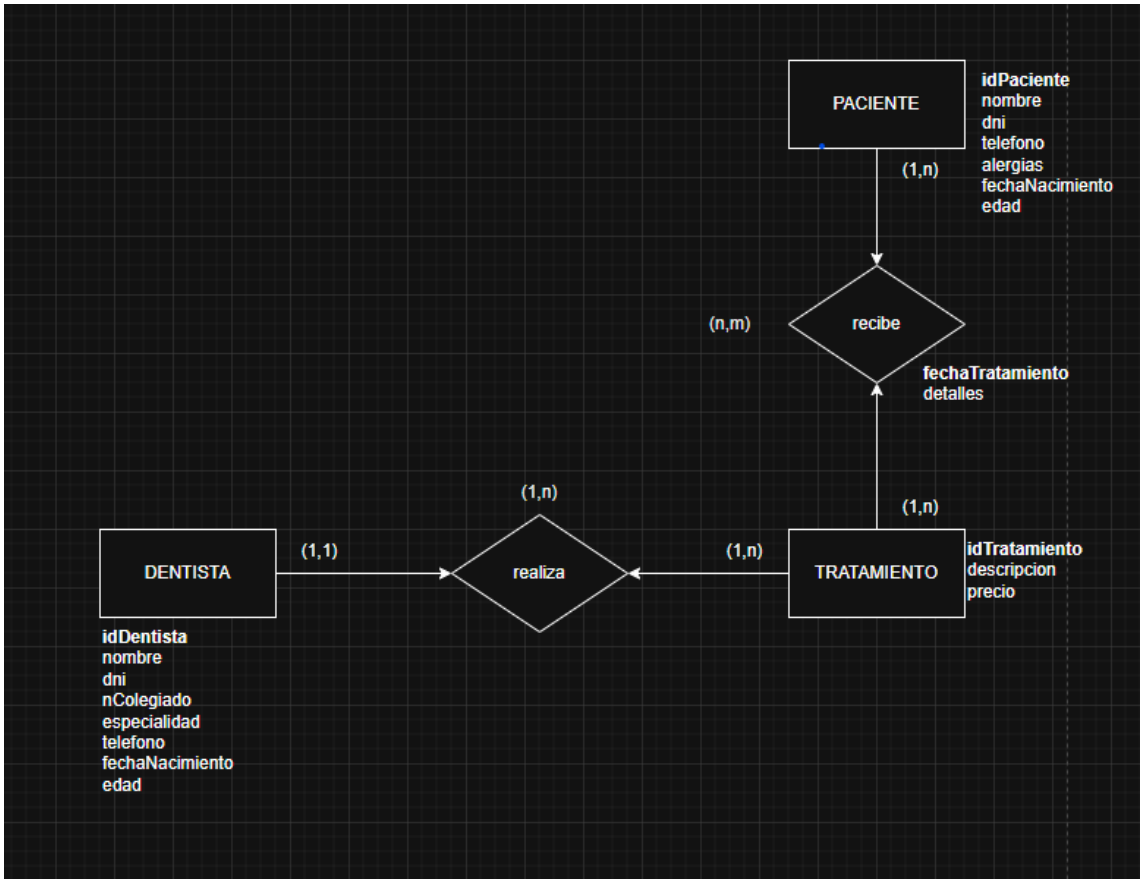
```
public Dentista(String nombre, Stri
    super(nombre, dni, telefono);
```

```
public Paciente(String nombre, String dni, int telefono,
    super(nombre, dni, telefono);
```

3.2 Relaciones de tablas

Las relaciones de tablas en mi modelo entidad-relación son las siguientes:

- Un paciente recibe (1,n) tratamientos y un tratamiento es recibido por (1,n) pacientes, esta relación es n,m y crea una tabla en medio
- Un dentista realiza (1,n) tratamientos, y un tratamiento es realizado por (1,1) un único dentista, esta relación es (1,n).



3.3 Construcción de DAOs

En el paquete DAO, he creado una clase para cada entidad de la base de datos, lo que resulta en 4 clases en estas se crea una nueva instancia singleton de otra clase DAO (si es necesario), se definen las consultas como atributos finales y estáticos y a continuación se crean los métodos que hacen uso de esas consultas.

```
/**
 * Consultas SQL
 */
private final static String SQL_CHECK = "SELECT COUNT(*) FROM Dentista WHERE idDentista = ?";  2 usages
private final static String SQL_ALL = "SELECT * FROM Dentista";  2 usages
private final static String SQL_FIND_BY_NAME = "SELECT * FROM Dentista WHERE nombre = ?";  1 usage
private final static String SQL_INSERT = "INSERT INTO Dentista (nombre, dni,nColegiado, especialidad, telefono, fechaNacimiento, edad) VALUES(?, ?, ?, ?, ?, ?, ?)";
private final static String SQL_UPDATE = "UPDATE Dentista SET nombre = ?, dni = ?, telefono = ?, nColegiado = ?, especialidad = ?, fechaNacimiento = ?, edad = ? WHE
private final static String SQL_DELETE_BY_ID = "DELETE FROM Dentista WHERE idDentista = ?";  1 usage
private final static String SQL_SELECT_BY_TRATAMIENTO = "SELECT * FROM Dentista WHERE idDentista IN (SELECT idDentista FROM Tratamiento WHERE idTratamiento = ?)";  1
```

```java
/**
 * Version lazy para obtener todos los dentistas en un list
 *
 * @return un list de dentistas
 */
@Override  FranciscoSolerAguado +1
public List<Object> findAll() {
    List<Dentista> dentistas = new ArrayList<~>();
    Connection con = ConnectionDB.getConnection();
    try {
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(SQL_ALL);
        while (rs.next()) {
            Dentista dentista = new Dentista();
            dentista.setIdDentista(rs.getInt( columnLabel: "idDentista"));
            dentista.setNombre(rs.getString( columnLabel: "nombre"));
            dentista.setDni(rs.getString( columnLabel: "dni"));
            dentista.setnColegiado(rs.getString( columnLabel: "nColegiado"));
            dentista.setEspecialidad(rs.getString( columnLabel: "especialidad"));
            dentista.setTelefono(rs.getInt( columnLabel: "telefono"));
            dentista.setFechaNacimiento(rs.getDate( columnLabel: "fechaNacimiento").toLocalDate());
            dentista.setEdad(rs.getInt( columnLabel: "edad"));

            //Version LAZY
            dentistas.add(dentista);
        }
    } catch (SQLException e) {
        logger.severe( msg: "Error al obtener todos los dentistas: " + e.getMessage());
        e.printStackTrace();
    }
    return new ArrayList<>(dentistas);
}
```

3.4 Funcionamiento de los botones

Los botones Añadir, editar, eliminar etc. funcionan desde una llamada desde el archivo fxml donde esta el botón a su método que realiza la acción en el controller de ese archivo fxml

Dentista.fxml

```
<HBox alignment="CENTER_LEFT" spacing="20.0">
    <Button fx:id="btnAdd" onAction="#abrirFormularioDentista"
    <cursor>
```

DentistaController.java

```
@FXML  Fran
private void abrirFormularioDentista() {
    try {
        FXMLLoader loader = new FXMLLoader(getClass().getResource( name: "/dentistaForm.fxml"));
        Parent root = loader.load();

        // Obtener el controlador del formulario
        DentistaFormController formController = loader.getController();
        formController.setDentistaController(this); // Pasar referencia del controlador actual

        Stage stage = new Stage();
        stage.setTitle("Añadir Dentista");
        stage.setScene(new Scene(root));
        stage.show();
```

Tanto el botón añadir como el botón editar llevan a otra pantalla donde se introducen los datos del dentista que se va a añadir/editar, mientras que el boto eliminar simplemente borra el dentista y actualiza la lista una vez borrado.