

CURSO : 4691 Lenguaje de programación II
PROFESOR : YOVANI EDY QUINTEROS CAMAPAZA
SEMESTRE : 2024-33
CICLO : Cuarto
SECCION : T4FA
GRUPO : 01
FECHA : 23/10/2024

NOTA

ALUMNO (A) : Gallegos Yanarico, Jarem Joseph

EXAMEN FINAL DE LABORATORIO

Consideraciones generales:

- Se considerará el orden, la claridad de las respuestas y las buenas prácticas. En cada pregunta consigne su respuesta (Código, comentarios, observaciones y resultados en captura de imágenes) adicionalmente la URL en GitHub.
- Las preguntas deben resolverse de acuerdo con los conceptos discutidos o analizados en clase. Para ello, aplicará su propio criterio para dar una solución a los problemas planteados
- Para el desarrollo de la siguiente aplicación cree el proyecto con el nombre LP11_EF_APELLIDO_NOMBRE y cree la estructura de carpetas necesarias

LOGRO DE LA EVALUACION:

Al término de la evaluación, el alumno implementa proceso de registro y listado de una tabla empleando JPA, realizando un correcto mapeado y una vista para su funcionalidad.

Consolidado

Pregunta	Puntaje		Llenar solo en caso de Recalificación justificada	
	Máximo	Obtenido	Sustento	Puntaje
1	20			
Nota Recalificada				

Contexto:

Un sistema de gestión para una clínica odontológica es una aplicación que facilita la administración y automatización de las actividades diarias de una clínica dental. Este tipo de sistema tiene como objetivo mejorar la eficiencia en la gestión de citas, expedientes médicos, inventarios de insumos, facturación y comunicación con los pacientes. Se encarga a Ud la implementación de una parte de uno de los módulos:

Gestión de Pacientes

- **Registro de pacientes:** El sistema permite registrar y mantener un historial detallado de los pacientes, incluyendo información personal (nombre, dirección, teléfono, correo electrónico) y datos médicos (historial odontológico, alergias, tratamientos previos).
- **Historial médico:** Guarda el historial completo de cada paciente, como los tratamientos recibidos, fecha, diagnóstico, radiografías, y médico que atendió.

Pregunta 1

- A. Crear un proyecto Spring Boot; Crear el modelo y crear el repositorio con las clases necesarias para la gestión de registro de pacientes. Suba su proyecto a Github y envíe la URL.

Respuesta: <https://github.com/JaremGallegos/ExamenFinal.git>

(Explicación de Integración GitHub y Desarrollo del proyecto en las siguientes páginas)

Paso 1: Creación de un Proyecto Spring

Personalmente, tengo problemas en el uso de Spring Tool Suite al crear proyectos y desarrollarlos, por lo que decidí utilizar Spring Initializr y Eclipse.

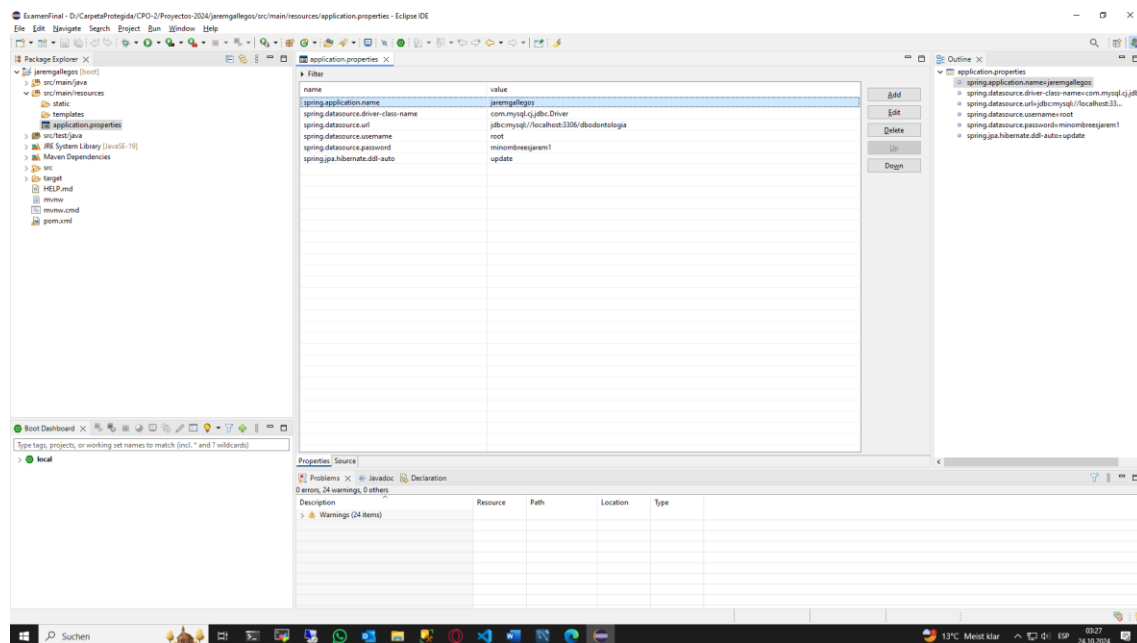
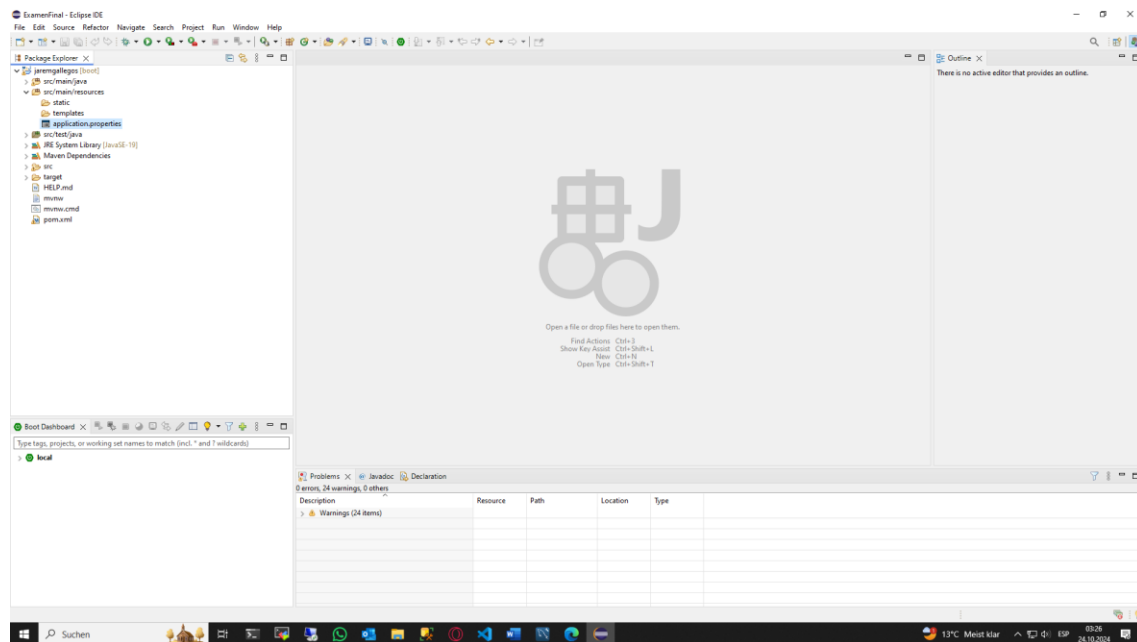
Como se puede observar en la captura, se añadieron las dependencias de Thymeleaf, Spring Web, Spring Data JPA y el conector MySQL. Además, se integró los datos del proyecto correctamente, bajo el artefacto y el nombre.

The screenshot shows the Spring Initializr web application interface. The interface is dark-themed and contains the following sections:

- Project:** Includes radio buttons for **Gradle - Groovy**, **Gradle - Kotlin**, and **Language** with options for **Java** (selected), **Kotlin**, and **Groovy**. There is also a **Maven** option.
- Spring Boot:** Includes radio buttons for versions: **3.4.0 (SNAPSHOT)**, **3.4.0 (M3)**, **3.3.5 (SNAPSHOT)**, **3.3.4** (selected), **3.2.11 (SNAPSHOT)**, and **3.2.10**.
- Project Metadata:** Includes input fields for **Group** (com.cibertec.examenfinal), **Artifact** (jaremgallegos), **Name** (jaremgallegos), **Description** (Examen Final de Spring y Jasper), and **Package name** (com.cibertec.examenfinal.jaremgallegos).
- Packaging:** Includes radio buttons for **Jar** (selected) and **War**.
- Java:** Includes radio buttons for versions: **23**, **21** (selected), and **17**.
- Dependencies:** Includes a button **ADD DEPENDENCIES... CTRL + B** and a list of dependencies: **Spring Web** (WEB), **Spring Data JPA** (SQL), **MySQL Driver** (SQL), and **Thymeleaf** (TEMPLATE ENGINES).
- Buttons:** At the bottom, there are buttons for **GENERATE CTRL + G**, **EXPLORE CTRL + SPACE**, and **SHARE...**

Paso 2: Configuración del archivo properties

Tras la realización de este primer paso, se abrió el proyecto generado en Eclipse y se realizó la configuración inicial del proyecto, teniendo que declararse en el archivo .properties, el nombre de usuario, contraseña, el conector, url y el acción hibernate para crear la base de datos.

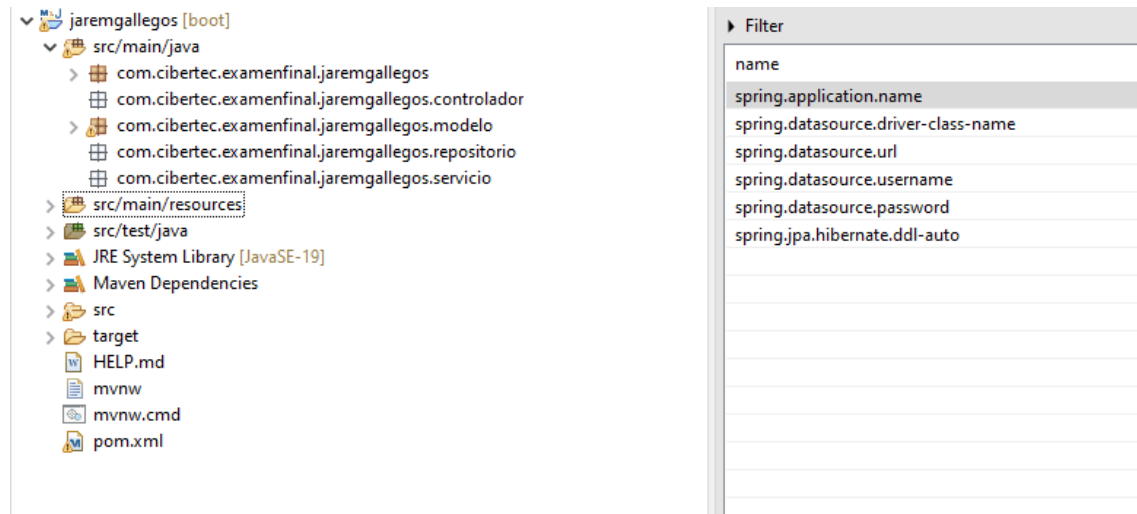


Código del archivo .properties:

```
spring.application.name=jaremgallegos
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/dbodontologia
spring.datasource.username=root
spring.datasource.password=minombreesjarem1
spring.jpa.hibernate.ddl-auto=create
```

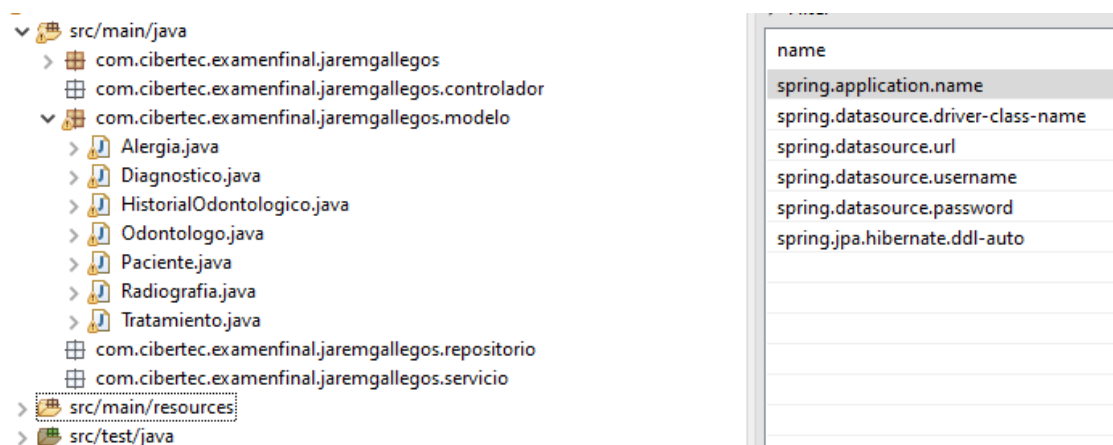
Paso 2: Creación de capas de arquitectura

Como se menciona en la propuesta se desea una arquitectura de servicio e integrado junto con un patrón mvc. De esta manera se comienza a crear los paquetes que albergarán a las clases para la creación del sistema de gestión. De esta manera definimos el paquete modelo, servicio, repositorio y controlador.



Paso 3: Creación de las clase modelo

Tras la creación de los paquetes, se necesita de poder crear las clases que servirán para el mapeo de la entidad de JPA a una tabla de la base de datos. De esta manera, através de la identificación del caso propuesto, se identificaron a 7 entidades, las cuales se lograron tras uan normalización de las entidades padres. Estas son: Alergia, Diagnostico, Historial Odontológico, Odontólogo, Paciente, Radiografía y Tratamiento.



Tras la creación de estas clases que servirán como mapeo de la base de datos, será necesario integrar dentro de las dependencias .pom la librería lombok, esto permitirá a que nos podamos enfocar más en las relaciones entre las diversas entidades y sin la necesidad de preocuparnos en la agregación de métodos de acceso debido al encapsulamiento.

```
<artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.18.30</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>com.mysql</groupId>
<artifactId>mysql-connector-j</artifactId>
<scope>runtime</scope>
</dependency>
```

De esta manera, a partir de los atributos mencionados en el caso propuesto y algunos deducidos se obtuvo el siguiente código de cada clase:

- **Clase Alergia**

```
1 package com.cibertec.examenfinal.jaremgallegos.modelo;
2
3 import java.util.List;
4
5
6 @Getter
7 @Setter
8 @NoArgsConstructor
9 @AllArgsConstructor
10 @ToString
11 @Entity
12 public class Alergia {
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Long id;
16
17     private String nombre;
18     private String descripcion;
19
20     @ManyToMany(mappedBy = "alergias")
21     private List<Paciente> pacientes;
22 }
```

```
package com.cibertec.examenfinal.jaremgallegos.modelo;
```

```
import java.util.List;
```

```
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToMany;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;
```

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Entity
public class Alergia {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nombre;
    private String descripcion;

    @ManyToMany(mappedBy = "alergias")
    private List<Paciente> pacientes;
}
```

- Clase Diagnostico

```

1 package com.cibertec.examenfinal.jaremgallegos.modelo;
2
3 import java.time.LocalDate;
4
5 @Getter
6 @Setter
7 @NoArgsConstructor
8 @AllArgsConstructor
9 @ToString
10 @Entity
11 public class Diagnostico {
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Long id;
15
16     private String descripcion;
17     private LocalDate fecha;
18
19     @ManyToOne
20     @JoinColumn(name = "paciente_id")
21     private Paciente paciente;
22
23     @ManyToOne
24     @JoinColumn(name = "odontologo_id")
25     private Odontologo odontologo;
26
27     @ManyToOne
28     @JoinColumn(name = "tratamiento_id")
29     private Tratamiento tratamiento;
30 }

```

```
package com.cibertec.examenfinal.jaremgallegos.modelo;
```

```
import java.time.LocalDate;
```

```

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

```

```

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Entity
public class Diagnostico {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String descripcion;
    private LocalDate fecha;

    @ManyToOne
    @JoinColumn(name = "paciente_id")
    private Paciente paciente;

    @ManyToOne
    @JoinColumn(name = "odontologo_id")
    private Odontologo odontologo;

    @ManyToOne
    @JoinColumn(name = "tratamiento_id")
    private Tratamiento tratamiento;
}

```

- Clase HistorialOdontologico

```

1 package com.cibertec.examenfinal.jaremgallegos.modelo;
2
3 import java.time.LocalDate;
4
5 @Getter
6 @Setter
7 @NoArgsConstructor
8 @AllArgsConstructor
9 @ToString
10 @Entity
11 public class HistorialOdontologico {
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Long id;
15
16     private String descripcion;
17     private LocalDate fecha;
18
19     @ManyToOne
20     @JoinColumn(name = "odontologo_id")
21     private Odontologo odontologo;
22
23     @ManyToOne
24     @JoinColumn(name = "paciente_id")
25     private Paciente paciente;
26
27     @OneToMany(mappedBy = "historialOdontologico")
28     private List<Radiografia> radiografias;
29 }

```

```

package com.cibertec.examenfinal.jaremgallegos.modelo;

```

```

import java.time.LocalDate;
import java.util.List;

```

```

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToMany;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

```

```

@Entity
public class HistorialOdontologico {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String descripcion;
    private LocalDate fecha;

    @ManyToOne
    @JoinColumn(name = "odontologo_id")
    private Odontologo odontologo;

    @ManyToOne
    @JoinColumn(name = "paciente_id")
    private Paciente paciente;

    @OneToMany(mappedBy = "historialOdontologico")
    private List<Radiografia> radiografias;
}

```


- Clase Odontólogo

```

1 package com.cibertec.examenfinal.jaremgallegos.modelo;
2
3 import java.util.List;
4
5
6 @Getter
7 @Setter
8 @NoArgsConstructor
9 @AllArgsConstructor
10 @ToString
11 @Entity
12 public class Odontologo {
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Long id;
16
17     private String nombre;
18     private String especialidad;
19     private String telefono;
20     private String correo;
21     private String direccion;
22
23     @OneToMany(mappedBy = "odontologo")
24     private List<HistorialOdontologico> historiales;
25
26     @OneToMany(mappedBy = "odontologo")
27     private List<Dagnostico> diagnosticos;
28 }
29
package com.cibertec.examenfinal.jaremgallegos.modelo;

import java.util.List;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Entity
public class Odontologo {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nombre;
    private String especialidad;
    private String telefono;
    private String correo;
    private String direccion;

    @OneToMany(mappedBy = "odontologo")
    private List<HistorialOdontologico> historiales;

    @OneToMany(mappedBy = "odontologo")
    private List<Dagnostico> diagnosticos;
}

```

• Clase Paciente

```

1 package com.cibertec.examenfinal.jaremgallegos.modelo;
2
3 import java.time.LocalDate;
4
5 @Getter
6 @Setter
7 @NoArgsConstructor
8 @AllArgsConstructor
9 @ToString
10 @Entity
11 public class Paciente {
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Long id;
15
16     private String nombre;
17     private String direccion;
18     private String telefono;
19     private String correo;
20     private String sexo;
21
22     @Column(name = "fecha_nacimiento")
23     private LocalDate fechaNacimiento;
24
25     @OneToMany(mappedBy = "paciente", cascade = CascadeType.ALL)
26     private List<HistorialOdontologico> historiales;
27
28     @OneToMany(mappedBy = "paciente", cascade = CascadeType.ALL)
29     private List<Diagnostico> diagnosticos;
30
31     @ManyToMany
32     @JoinTable(
33         name = "paciente_alergia",
34         joinColumns = @JoinColumn(name = "paciente_id"),
35         inverseJoinColumns = @JoinColumn(name = "alergia_id")
36     )
37     private List<Alergia> alergias;
38
39     @OneToMany(mappedBy = "paciente", cascade = CascadeType.ALL)
40     private List<Tratamiento> tratamientos;
41 }

```

```

package com.cibertec.examenfinal.jaremgallegos.modelo;

import java.time.LocalDate;
import java.util.List;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.JoinTable;
import jakarta.persistence.ManyToMany;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToOne;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Entity
public class Paciente {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nombre;
    private String direccion;
    private String telefono;
    private String correo;
    private String sexo;

    @Column(name = "fecha_nacimiento")
    private LocalDate fechaNacimiento;

    @OneToMany(mappedBy = "paciente", cascade = CascadeType.ALL)
    private List<HistorialOdontologico> historiales;

```

```

    @OneToMany(mappedBy = "paciente", cascade = CascadeType.ALL)
    private List<Diagnostico> diagnosticos;

    @ManyToMany
    @JoinTable(
        name = "paciente_alergia",
        joinColumns = @JoinColumn(name = "paciente_id"),
        inverseJoinColumns = @JoinColumn(name = "alergia_id")
    )
    private List<Alergia> alergias;

    @OneToMany(mappedBy = "paciente", cascade = CascadeType.ALL)
    private List<Tratamiento> tratamientos;
}

```

- Clase Radiografía

```

1 package com.cibertec.examenfinal.jaremgallegos.modelo;
2
3 import java.time.LocalDate;
4
5
6
7 @Getter
8 @Setter
9 @NoArgsConstructor
10 @AllArgsConstructor
11 @ToString
12 @Entity
13 public class Radiografia {
14     @Id
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private Long id;
17
18     private String url;
19     private LocalDate fecha;
20     private String tipo;
21
22     @ManyToOne
23     @JoinColumn(name = "historial_odontologico_id")
24     private HistorialOdontologico historialOdontologico;
25 }

```

```

package com.cibertec.examenfinal.jaremgallegos.modelo;

import java.time.LocalDate;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Entity
public class Radiografia {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String url;
    private LocalDate fecha;
    private String tipo;

    @ManyToOne
    @JoinColumn(name = "historial_odontologico_id")
    private HistorialOdontologico historialOdontologico;
}

```

• Clase Tratamiento

```

1 package com.cibertec.examenfinal.jaremgallegos.modelo;
2
3 import java.util.List;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18 @Getter
19 @Setter
20 @NoArgsConstructor
21 @AllArgsConstructor
22 @ToString
23 @Entity
24 public class Tratamiento {
25     @Id
26     @GeneratedValue(strategy = GenerationType.IDENTITY)
27     private Long id;
28
29     private String nombre;
30     private String descripcion;
31     private String duracion;
32     private double costo;
33
34     @ManyToOne
35     @JoinColumn(name = "paciente_id")
36     private Paciente paciente;
37
38     @OneToMany(mappedBy = "tratamiento")
39     private List<Diagnostico> diagnosticos;
40 }
41
package com.cibertec.examenfinal.jaremgallegos.modelo;

import java.util.List;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToMany;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Entity
public class Tratamiento {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nombre;
    private String descripcion;
    private String duracion;
    private double costo;

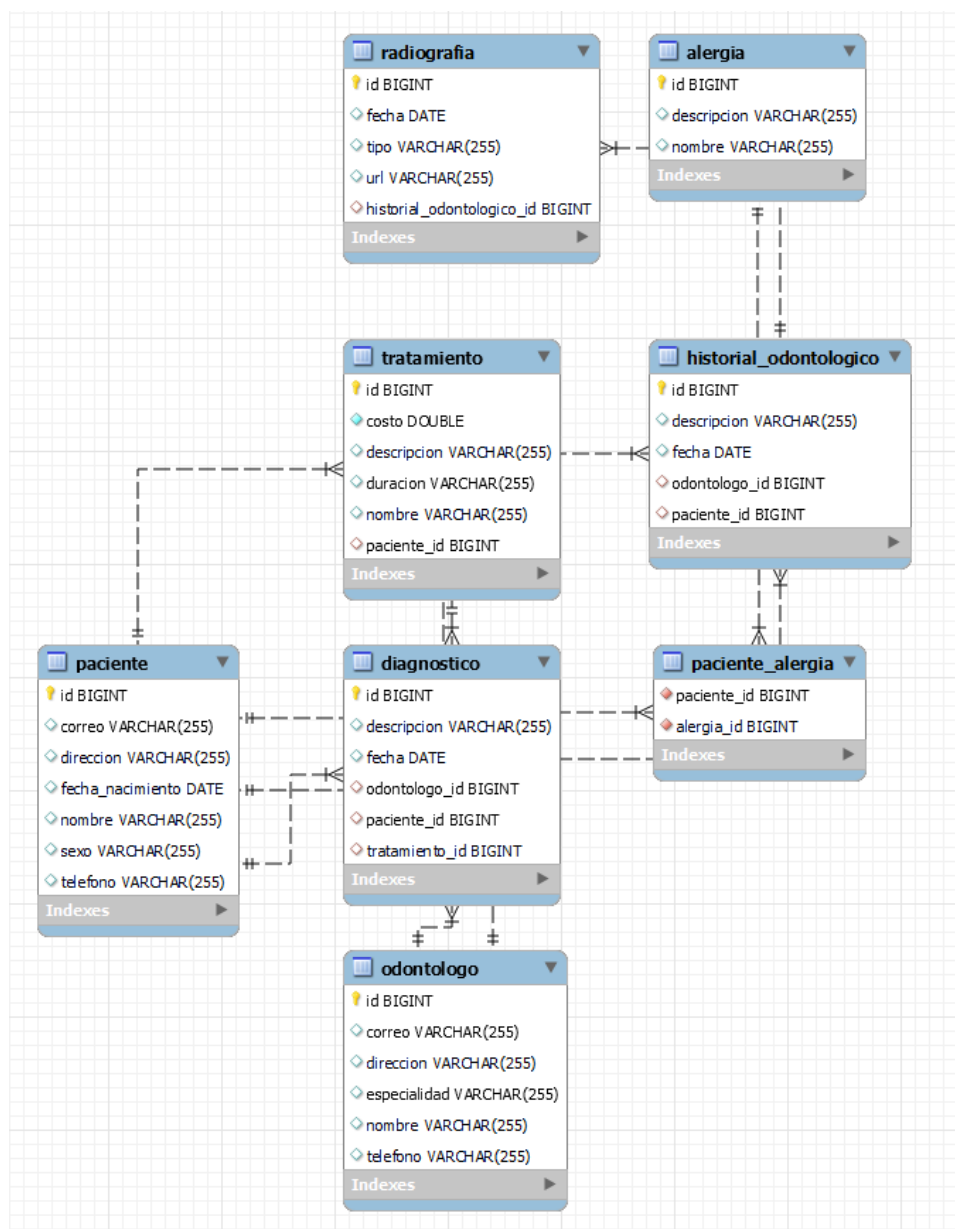
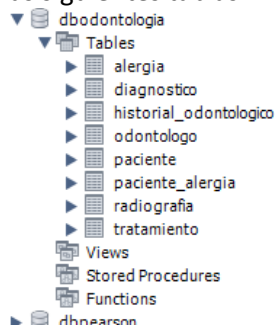
    @ManyToOne
    @JoinColumn(name = "paciente_id")
    private Paciente paciente;

    @OneToMany(mappedBy = "tratamiento")
    private List<Diagnostico> diagnosticos;
}

```

Paso 4: Generación de tablas a partir del mapeado

Tras la creación de atributos de cada una de las clases, al ejecutar hibernete, este podrá mapear todas las clases y crear tablas para la gestión en la base de datos. De esta manera se generarán las siguientes tablas:



Cabe mencionar que para poder mapear cada una de las clases en esquemas de tablas, se debió de poder haber creado previamente la base de datos, de esta manera hibernate solo se encargará de poder crear las tablas y establecer los atributos de cada entidad. Por otro lado, tras la creación de las tablas, es necesario poder cambiar el parámetro hibernate.ddl-auto a update, de esta manera se evitará en un futuro que se borren datos tras la ejecución del proyecto:

name	value
spring.application.name	jaremgallegos
spring.datasource.driver-class-name	com.mysql.cj.jdbc.Driver
spring.datasource.url	jdbc:mysql://localhost:3306/dbodontologia
spring.datasource.username	root
spring.datasource.password	minombreesjarem1
spring.jpa.hibernate.ddl-auto	update

Paso 5: Creación de Repositorio en GitHub

Dado de que se quiere poder manipular las versiones de nuestro proyecto, es importante poder crear un repositorio en GitHub para poder modificar y recuperar sin la necesidad de haber guardado de manera manual un proyecto backup.

Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * JaremGallegos / **Repository name *** ExamenFinal
ExamenFinal is available.

Great repository names are short and memorable. Need inspiration? How about [scaling-happiness](#)?

Description (optional)
Solución del Examen Final de LP2

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

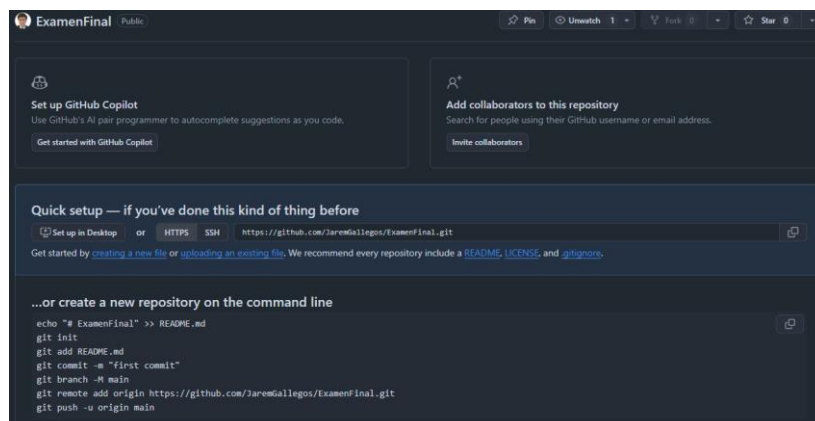
Choose a license
License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

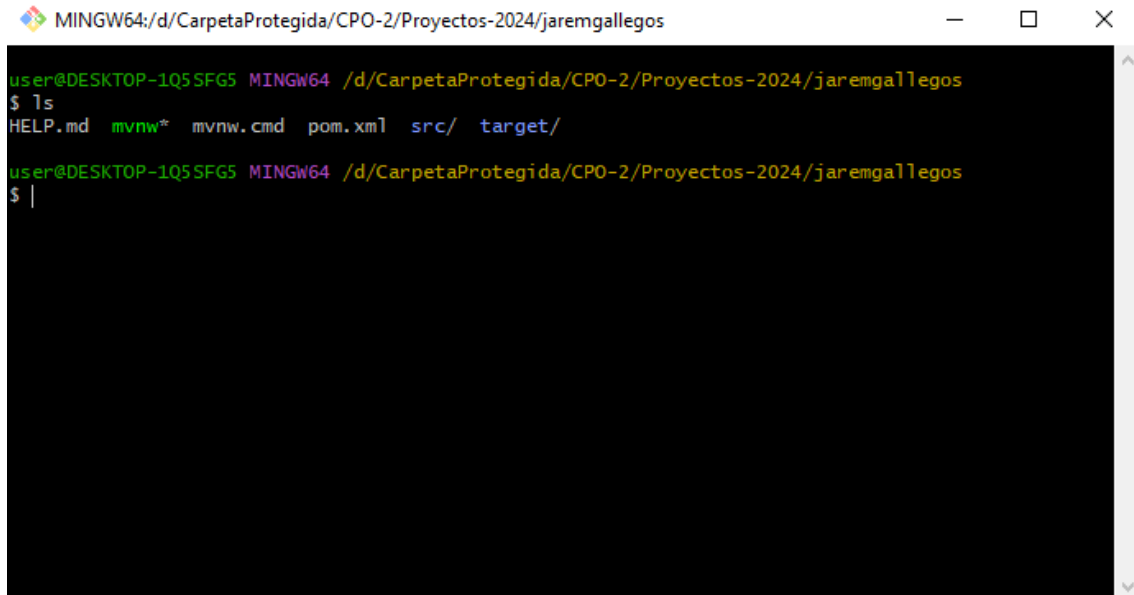
① You are creating a public repository in your personal account.

Create repository

Tras el ingreso del nombre del repositorio y una breve descripción, se crea el repositorio.

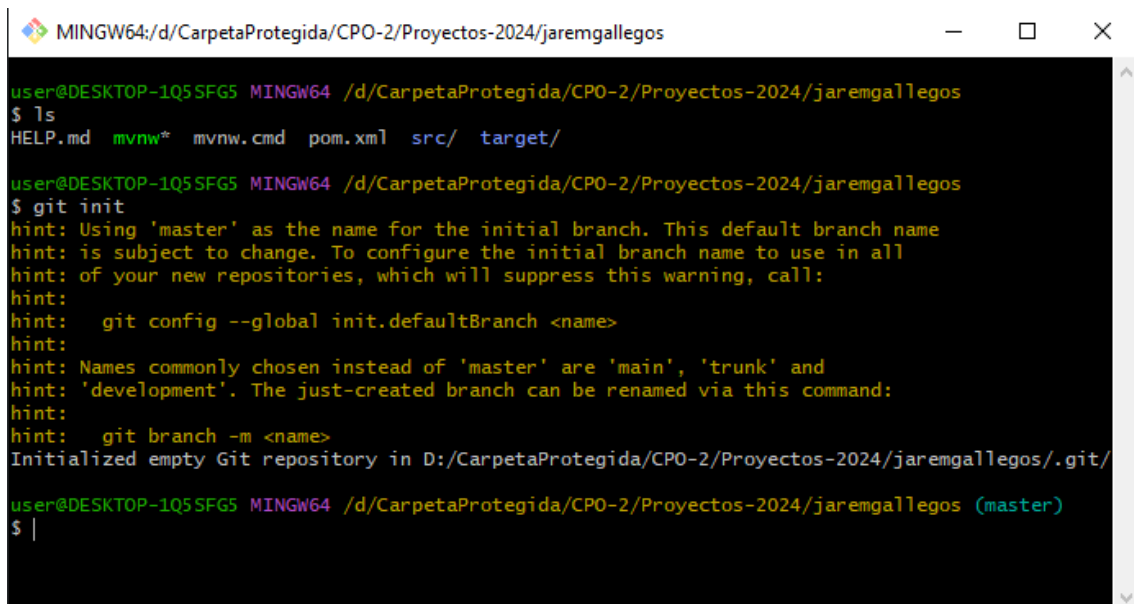


Tras la creación, se ingresa a la consola de git, para poder ejecutar los comandos y poder realizar un commit. Primeramente, buscamos el directorio raíz de nuestro proyecto.



```
MINGW64:/d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos
user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos
$ ls
HELP.md  mvnw*  mvnw.cmd  pom.xml  src/  target/
user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos
$ |
```

Tras esto, se ingresa el comando de git init



```
MINGW64:/d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos
user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos
$ ls
HELP.md  mvnw*  mvnw.cmd  pom.xml  src/  target/
user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos
$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in D:/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos/.git/
user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos (master)
$ |
```

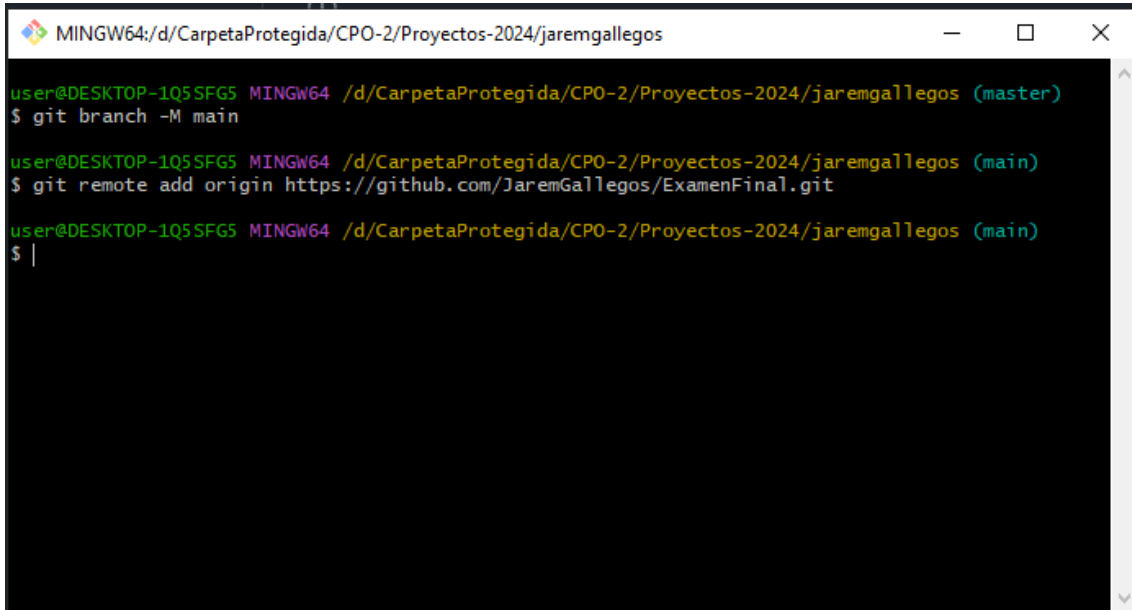

Luego ejecutamos el comando “git add .” para incluir todos nuestros archivos al commit.

```
MINGW64:/d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos
user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos (master)
$ git add .
warning: in the working copy of '.gitattributes', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of '.mvn/wrapper/maven-wrapper.properties', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'mvnw.cmd', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'pom.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main/java/com/cibertec/examenfinal/jaremgallegos/JaremgallegosApplication.java', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main/resources/application.properties', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/test/java/com/cibertec/examenfinal/jaremgallegos/JaremgallegosApplicationTests.java', LF will be replaced by CRLF the next time Git touches it
user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos (master)
$ |
```

Luego ejecutamos el comando git commit -m “mensaje” para poder lanzar nuestro primer commit.

```
MINGW64:/d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos
user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos (master)
$ git commit -m "Primer commit"
[master (root-commit) 21aad3c] Primer commit
16 files changed, 855 insertions(+)
create mode 100644 .gitattributes
create mode 100644 .gitignore
create mode 100644 .mvn/wrapper/maven-wrapper.properties
create mode 100644 mvnw
create mode 100644 mvnw.cmd
create mode 100644 pom.xml
create mode 100644 src/main/java/com/cibertec/examenfinal/jaremgallegos/JaremgallegosApplication.java
create mode 100644 src/main/java/com/cibertec/examenfinal/jaremgallegos/modelo/Alergia.java
create mode 100644 src/main/java/com/cibertec/examenfinal/jaremgallegos/modelo/Diagnostico.java
create mode 100644 src/main/java/com/cibertec/examenfinal/jaremgallegos/modelo/HistorialOdontologico.java
create mode 100644 src/main/java/com/cibertec/examenfinal/jaremgallegos/modelo/Odontologo.java
create mode 100644 src/main/java/com/cibertec/examenfinal/jaremgallegos/modelo/Paciente.java
create mode 100644 src/main/java/com/cibertec/examenfinal/jaremgallegos/modelo/Radiografia.java
create mode 100644 src/main/java/com/cibertec/examenfinal/jaremgallegos/modelo/Tratamiento.java
create mode 100644 src/main/resources/application.properties
create mode 100644 src/test/java/com/cibertec/examenfinal/jaremgallegos/JaremgallegosApplicationTests.java
```

Tras realizar esto, se ejecuta el comando de `git branch -M main`, para poder cambiar el nombre de la rama principal “master” a “main”. Además, indicamos con `git remote add origin` el repositorio en donde se desea guardar el proyecto.

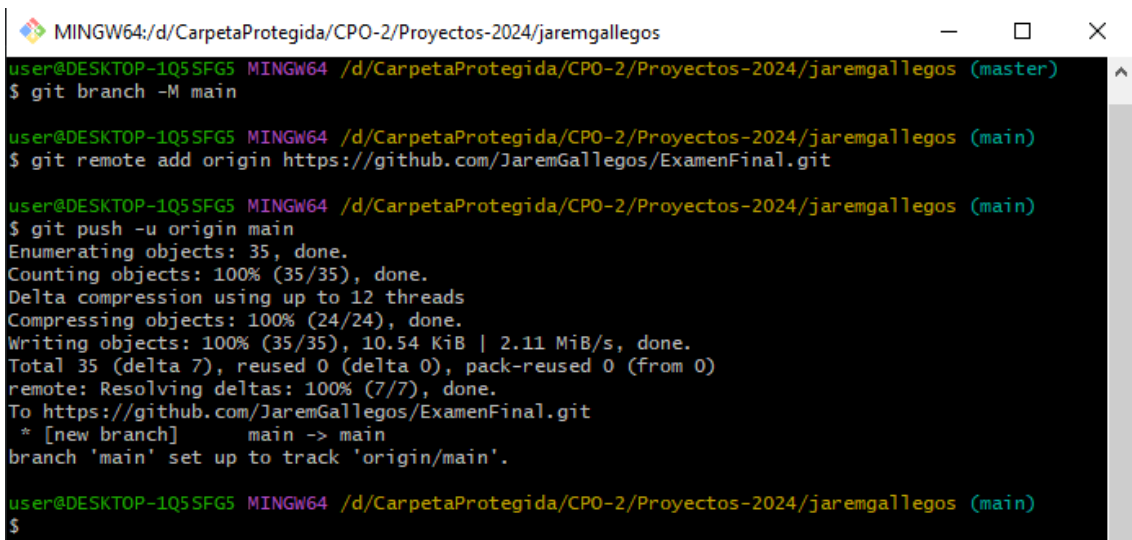


```
MINGW64:/d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos
user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos (master)
$ git branch -M main

user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos (main)
$ git remote add origin https://github.com/JaremGallegos/ExamenFinal.git

user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos (main)
$ |
```

Finalmente, para poder agregar nuestro commit a nuestro repositorio, utilizamos el comando “git push” para poder subir nuestro proyecto a la rama principal



```
MINGW64:/d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos
user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos (master)
$ git branch -M main

user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos (main)
$ git remote add origin https://github.com/JaremGallegos/ExamenFinal.git

user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos (main)
$ git push -u origin main
Enumerating objects: 35, done.
Counting objects: 100% (35/35), done.
Delta compression using up to 12 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (35/35), 10.54 KiB | 2.11 MiB/s, done.
Total 35 (delta 7), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/JaremGallegos/ExamenFinal.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

user@DESKTOP-1Q5SFG5 MINGW64 /d/CarpetaProtegida/CPO-2/Proyectos-2024/jaremgallegos (main)
$
```

De esta manera se comprueba el cargado de nuestro proyecto a nuestro repositorio

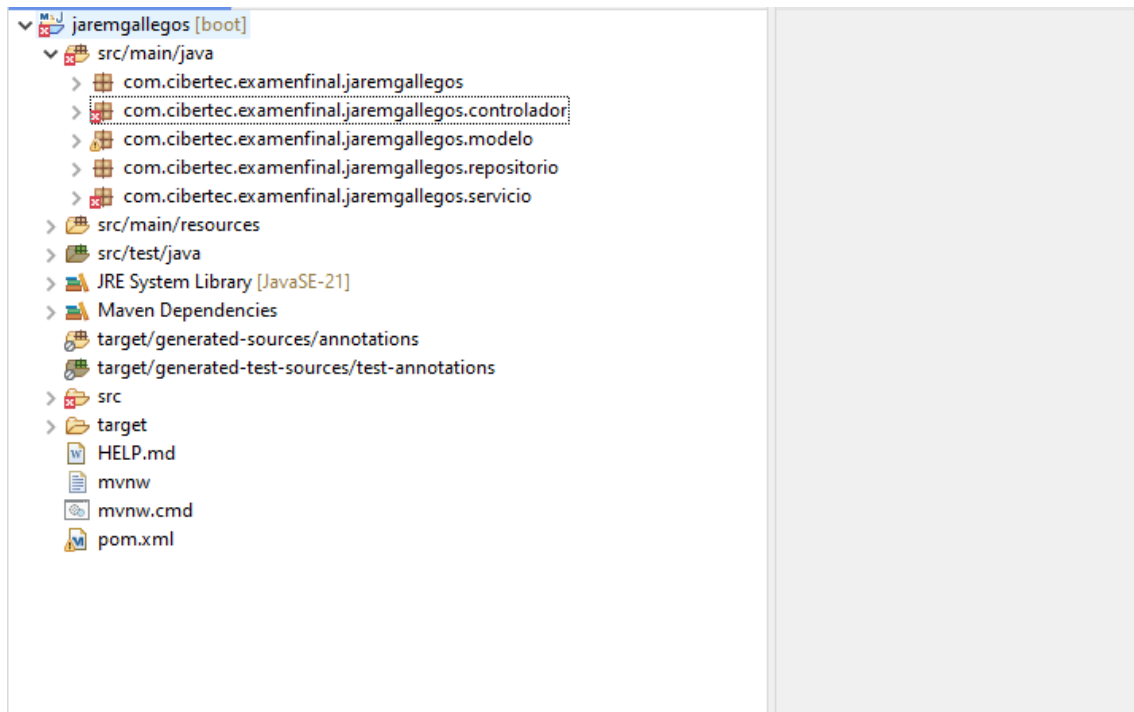
The screenshot shows a GitHub repository named 'ExamenFinal' by user 'JaremGallegos'. The repository is public and has 1 branch (main) and 1 commit. The commit history shows a 'Primer commit' at 21aad3c, 7 minutes ago. The file list includes: .mvn/wrapper, src, .gitattributes, .gitignore, mvnw, mvnw.cmd, and pom.xml, all from the 'Primer commit' 7 minutes ago. The README section prompts to 'Add a README' with a button 'Add a README'. The right sidebar shows repository statistics: 0 stars, 1 watching, 0 forks. It also lists 'Releases' (no releases published), 'Packages' (no packages published), and 'Languages' (Java 100.0%). Under 'Suggested workflows', it shows a 'Clojure' workflow with a 'Configure' button.

<https://github.com/JaremGallegos/ExamenFinal.git>

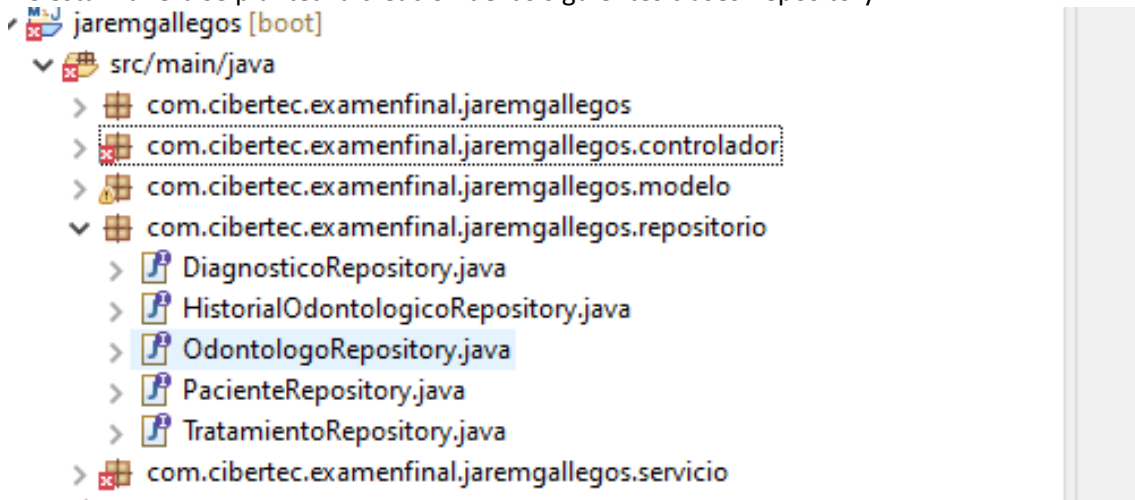
- B. Crear el controlador y la capa servicio que permita realizar las siguientes gestiones:
- Registro de pacientes y lista de pacientes
 - Registro de atenciones y listas de atenciones
 - Registro de historial medico de pacientes y por paciente

Paso 1: Creación de capas de arquitectura

Dado que ya se creó el paquete de modelo, para poder desarrollar con la arquitectura planteada, será necesario poder crear los paquetes Repository, Service y Controller para poder crear la lógica y acciones dentro del sistema planteado.



De esta manera se plantea la creación de las siguientes clases Repository:



Clases Repository creadas:

```
package com.cibertec.examenfinal.jaremgallegos.repositorio;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.cibertec.examenfinal.jaremgallegos.modelo.Diagnostico;

@Repository
public interface DiagnosticoRepository extends JpaRepository<Diagnostico, Long> {

}

public interface OdontologoRepository extends JpaRepository<Odontologo, Long> {

}

@Repository
public interface PacienteRepository extends JpaRepository<Paciente, Long> {

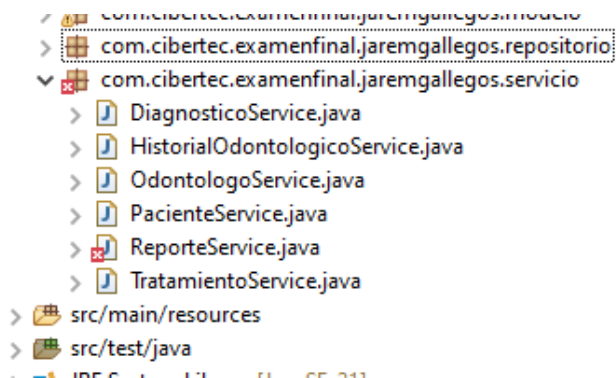
}

@Repository
public interface TratamientoRepository extends JpaRepository<Tratamiento, Long>{

}

@Repository
public interface HistorialOdontologicoRepository extends
JpaRepository<HistorialOdontologico, Long> {
    List<HistorialOdontologico> findByPacienteId(Long pacienteId);
}
```

De esta manera se plantea la creación de las siguientes clases Service:



Clases Service creadas:

```
package com.cibertec.examenfinal.jaremgallegos.servicio;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.cibertec.examenfinal.jaremgallegos.modelo.Diagnostico;
import com.cibertec.examenfinal.jaremgallegos.repositorio.DiagnosticoRepository;

@Service
public class DiagnosticoService {
    @Autowired
    private DiagnosticoRepository diagnosticoRepository;

    public List<Diagnostico> listarDiagnosticos() {
        return diagnosticoRepository.findAll();
    }

    public Diagnostico registrarDiagnostico(Diagnostico diagnostico) {
        return diagnosticoRepository.save(diagnostico);
    }
}

@Service
public class HistorialOdontologicoService {
    @Autowired
    private HistorialOdontologicoRepository historialRepository;

    public List<HistorialOdontologico> listarHistoriales() {
        return historialRepository.findAll();
    }

    public List<HistorialOdontologico> listarHistorialesPorPaciente(Long pacienteId)
    {
        return historialRepository.findByPacienteId(pacienteId);
    }

    public HistorialOdontologico registrarHistorial(HistorialOdontologico historial)
    {
        return historialRepository.save(historial);
    }
}

@Service
public class OdontologoService {
    @Autowired
    private OdontologoRepository odontologoRepository;

    public List<Odontologo> listarTodos() {
        return odontologoRepository.findAll();
    }

    public Optional<Odontologo> obtenerPorId(Long id) {
        return odontologoRepository.findById(id);
    }

    public Odontologo guardar(Odontologo odontologo) {
        return odontologoRepository.save(odontologo);
    }

    public void eliminar(Long id) {
        odontologoRepository.deleteById(id);
    }
}
```

```

    }
}

@Service
public class PacienteService {
    @Autowired
    private PacienteRepository pacienteRepository;

    public List<Paciente> listarPacientes() {
        return pacienteRepository.findAll();
    }

    public Paciente registrarPaciente(Paciente paciente) {
        return pacienteRepository.save(paciente);
    }

    public Optional<Paciente> obtenerPacientePorId(Long id) {
        return pacienteRepository.findById(id);
    }
}

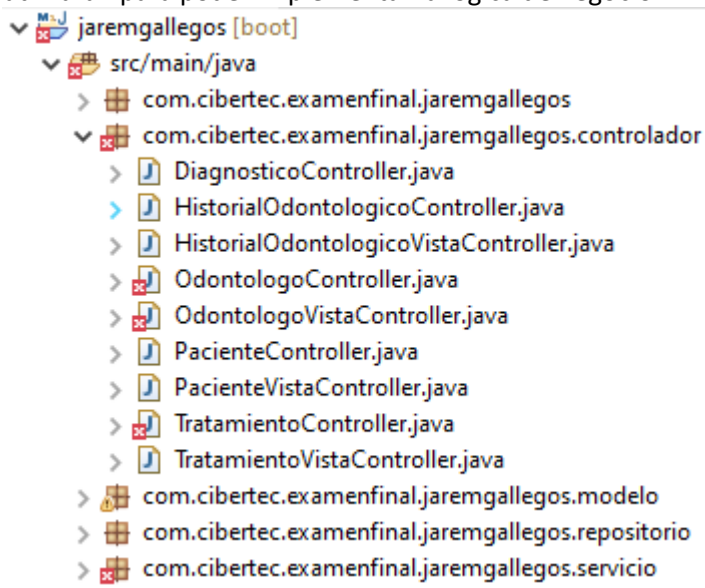
@Service
public class TratamientoService {
    @Autowired
    private TratamientoRepository tratamientoRepository;

    public List<Tratamiento> listarTratamientos() {
        return tratamientoRepository.findAll();
    }

    public Tratamiento registrarTratamiento(Tratamiento tratamiento) {
        return tratamientoRepository.save(tratamiento);
    }
}

```

De esta manera se plantea la creación de las siguientes clases Controller Rest y Vista, que se utilizarán para poder implementar la lógica de negocio:



Clases Controller Rest creadas:

```
@RestController
@RequestMapping("/api/diagnosticos")
public class DiagnosticoController {
    @Autowired
    private DiagnosticoService diagnosticoService;

    @GetMapping
    public List<Diagnostico> listarDiagnosticos() {
        return diagnosticoService.listarDiagnosticos();
    }

    @PostMapping
    public Diagnostico registrarDiagnostico(@RequestBody Diagnostico diagnostico) {
        return diagnosticoService.registrarDiagnostico(diagnostico);
    }
}

@RestController
@RequestMapping("/api/historiales")
public class HistorialOdontologicoController {
    @Autowired
    private HistorialOdontologicoService historialService;

    @GetMapping
    public List<HistorialOdontologico> listarHistoriales() {
        return historialService.listarHistoriales();
    }

    @GetMapping("/paciente/{pacienteId}")
    public List<HistorialOdontologico> listarHistorialesPorPaciente(@PathVariable
Long pacienteId) {
        return historialService.listarHistorialesPorPaciente(pacienteId);
    }

    @PostMapping
    public HistorialOdontologico registrarHistorial(@RequestBody
HistorialOdontologico historial) {
        return historialService.registrarHistorial(historial);
    }
}

@RestController
@RequestMapping("/api/odontologos")
public class OdontologoController {
    @Autowired
    private OdontologoService odontologoService;

    @GetMapping
    public List<Odontologo> listarOdontologos() {
        return odontologoService.listarTodos();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Odontologo> obtenerOdontologoPorId(@PathVariable Long id) {
        Optional<Odontologo> odontologo = odontologoService.obtenerPorId(id);
        return odontologo.map(ResponseEntity::ok)
            .orElseGet(() -> ResponseEntity.notFound().build());
    }

    @PostMapping
    public Odontologo crearOdontologo(@RequestBody Odontologo odontologo) {
        return odontologoService.guardar(odontologo);
    }
}
```



```

        @PutMapping("/{id}")
        public ResponseEntity<Odontologo> actualizarOdontologo(@PathVariable Long id,
@RequestBody Odontologo detallesOdontologo) {
            Optional<Odontologo> odontologo = odontologoService.obtenerPorId(id);

            if (odontologo.isPresent()) {
                Odontologo odontologoActualizado = odontologo.get();
                odontologoActualizado.setNombre(detallesOdontologo.getNombre());

odontologoActualizado.setEspecialidad(detallesOdontologo.getEspecialidad());
                odontologoActualizado.setTelefono(detallesOdontologo.getTelefono());
                odontologoActualizado.setCorreo(detallesOdontologo.getCorreo());
                odontologoActualizado.setDireccion(detallesOdontologo.getDireccion());

                return
ResponseEntity.ok(odontologoService.guardar(odontologoActualizado));
            } else {
                return ResponseEntity.notFound().build();
            }
        }

        @DeleteMapping("/{id}")
        public ResponseEntity<Void> eliminarOdontologo(@PathVariable Long id) {
            if (odontologoService.obtenerPorId(id).isPresent()) {
                odontologoService.eliminar(id);
                return ResponseEntity.noContent().build();
            } else {
                return ResponseEntity.notFound().build();
            }
        }
    }
}

```

```

@RestController
@RequestMapping("/api/pacientes")
public class PacienteController {
    @Autowired
    private PacienteService pacienteService;

    @Autowired
    private ReporteService reporteService;

    @GetMapping
    public List<Paciente> listarPacientes() {
        return pacienteService.listarPacientes();
    }

    @PostMapping
    public ResponseEntity<String> registrarPaciente(@ModelAttribute Paciente
paciente) {
        Paciente nuevoPaciente = pacienteService.registrarPaciente(paciente);
        String mensaje = reporteService.exportReportePDF(nuevoPaciente);
        return ResponseEntity.ok("Paciente registrado" + mensaje);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Paciente> obtenerPacientePorId(@PathVariable Long id) {
        Optional<Paciente> pacienteOpt = pacienteService.obtenerPacientePorId(id);
        return pacienteOpt.map(ResponseEntity::ok)
            .orElse(ResponseEntity.notFound().build());
    }
}

```

```

@RestController

```

```

@RequestMapping("/api/atenciones")
public class TratamientoController {
    @Autowired
    private TratamientoService tratamientoService;

    @Autowired
    private PacienteService pacienteService;

    @GetMapping
    public List<Tratamiento> listarTratamientos() {
        return tratamientoService.listarTratamientos();
    }

    @PostMapping
    public Tratamiento registrarTratamiento(@ModelAttribute Tratamiento tratamiento)
    {
        Optional<Paciente> paciente =
pacienteService.obtenerPacientePorId(tratamiento.getPaciente().getId());
        if (paciente.isPresent()) {
            tratamiento.setPaciente(paciente.get());
            return tratamientoService.registrarTratamiento(tratamiento);
        } else {
            throw new RuntimeException("Paciente no encontrado");
        }
    }
}

```

Clases Controller Vista creadas:

```

@Controller
@RequestMapping("/historiales")
public class HistorialOdontologicoVistaController {
    @Autowired
    private DiagnosticoService diagnosticoService;

    @GetMapping
    public String listarHistoriales(Model model) {
        model.addAttribute("historialOdontologico", new HistorialOdontologico());
        model.addAttribute("diagnosticos", diagnosticoService.listarDiagnosticos());
        return "historial";
    }
}

@Controller
@RequestMapping("/odontologos")
public class OdontologoVistaController {
    @Autowired
    private OdontologoService odontologoService;

    @GetMapping
    public String listarOdontologos(Model model) {
        List<Odontologo> odontologos = odontologoService.listarTodos();
        model.addAttribute("odontologos", odontologos);
        return "odontologos/lista";
    }

    @GetMapping("/nuevo")
    public String mostrarFormularioDeNuevoOdontologo(Model model) {
        model.addAttribute("odontologo", new Odontologo());
        return "odontologos/formulario";
    }

    @PostMapping
    public String guardarOdontologo(@ModelAttribute Odontologo odontologo) {
        odontologoService.guardar(odontologo);
        return "redirect:/odontologos";
    }
}

```

```

        @GetMapping("/editar/{id}")
        public String mostrarFormularioDeEditarOdontologo(@PathVariable Long id, Model
model) {
            Odontologo odontologo = odontologoService.obtenerPorId(id).orElseThrow(() ->
new IllegalArgumentException("Odontologo no encontrado: " + id));
            model.addAttribute("odontologo", odontologo);
            return "odontologos/formulario";
        }

        @PostMapping("/editar/{id}")
        public String actualizarOdontologo(@PathVariable Long id, @ModelAttribute
Odontologo odontologo) {
            odontologo.setId(id);
            odontologoService.guardar(odontologo);
            return "redirect:/odontologos";
        }

        @GetMapping("/eliminar/{id}")
        public String eliminarOdontologo(@PathVariable Long id) {
            odontologoService.eliminar(id);
            return "redirect:/odontologos";
        }
    }

@Controller
@RequestMapping("/pacientes")
public class PacienteVistaController {
    @Autowired
    private PacienteService pacienteService;

    @GetMapping
    public String listarPacientes(Model model) {
        List<Paciente> pacientes = pacienteService.listarPacientes();
        model.addAttribute("pacientes", pacientes);
        model.addAttribute("paciente", new Paciente());
        return "pacientes";
    }
}

@Controller
@RequestMapping("/atenciones")
public class TratamientoVistaController {
    @Autowired
    private TratamientoService tratamientoService;
    @Autowired
    private PacienteService pacienteService;

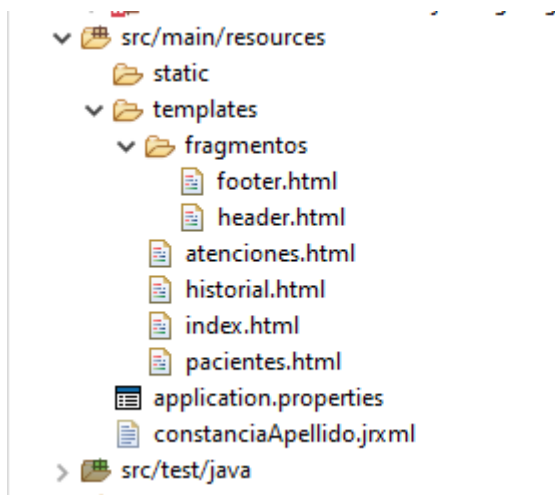
    @GetMapping
    public String listarTratamientos(Model model) {
        model.addAttribute("tratamientos", tratamientoService.listarTratamientos());
        model.addAttribute("nuevoTratamiento", new Tratamiento());
        model.addAttribute("pacientes", pacienteService.listarPacientes());
        return "atenciones";
    }
}

```

La creación de todos los controladores modelos y demás se crearon en el proyecto, si no se muestran se debe a la gran cantidad de elementos

Paso 2: Creación de vista para poder registrar y listar un paciente

Para el logro de la creación de la vista, se integró archivos html con Bootstrap dentro de la carpeta templates. Por otro lado también se fragmentó la parte de cabecera y footer para poder integrarlo como componente.



De esta manera se creó el siguiente dashboard:

Pacientes Historial Odontológico Atenciones

Registrar Paciente

Nombre:

Dirección:

Teléfono:

Correo:

Nombre	Dirección	Teléfono	Correo	Acciones
Jarem Gallegos	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<input type="button" value="Ver"/>
Cuzcano	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<input type="button" value="Ver"/>
Gutierrez	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<input type="button" value="Ver"/>
Cuzano Gutierrez	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<input type="button" value="Ver"/>
Jarem Gallegos	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<input type="button" value="Ver"/>
Cuzcano	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<input type="button" value="Ver"/>
Felipe	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<input type="button" value="Ver"/>
Cuzcano	Urb. Pedro Diez Canseco	946333421	jaremgallegosp@gmail.com	<input type="button" value="Ver"/>

En este caso se integrarán datos para poder registrar a un paciente

Registrar Paciente

Nombre:

cuzcano

Dirección:

Urb. Pedro Diez Canseco

Teléfono:

946333421

Correo:

jaremgallegosp@gmail.com

Registrar

Nombre	Dirección	Teléfono	Correo	Acciones
Jarem Gallegos	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Cuzcano	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Gutierrez	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Cuzano Gutierrez	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>

Correo:

Registrar


Nombre	Dirección	Teléfono	Correo	Acciones
Jarem Gallegos	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Cuzcano	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Gutierrez	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Cuzano Gutierrez	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Jarem Gallegos	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Cuzcano	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Felipe	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Cuzcano	Urb. Pedro Diez Canseco	946333421	jaremgallegosp@gmail.com	<div>Ver</div>
Cuzcano	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Felipe	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Jarem Gallegos	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Jarem Gallegos	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	<div>Ver</div>
Jose Maria	Urb. Miraflores	9421233422	josemaria@gmail.com	<div>Ver</div>
cuzcano	Urb. Pedro Diez Canseco	946333421	jaremgallegosp@gmail.com	<div>Ver</div>

© 2024 Gestión de Pacientes - Jarem Galleqos

Paso 3: Creación de vista para poder registrar y listar una atención



De la misma forma con el caso anterior, para poder registrar una atención, se tiene que haber creado un paciente, de esta manera para mayor facilidad, se estableció una barra de opciones.

Registrar Atención

Cita:	<input type="text"/>
Descripción:	<input type="text"/>
Duración:	<input type="text"/>
Costo:	<input type="text" value="0,0"/>
Fecha:	<input type="text" value="dd/mm/aaaa"/> 
Paciente:	<input type="text" value="Jarem Gallegos"/> 
<input type="button" value="Registrar Atención"/>	

Para continuar con el caso anterior, se seleccionará al paciente recientemente registrado

Registrar Atención

Cita:	<input type="text" value="Tramamiento de Dental"/>
Descripción:	<input type="text" value="Después de las 12:00"/>
Duración:	<input type="text" value="2"/>
Costo:	<input type="text" value="200"/>
Fecha:	<input type="text" value="17/10/2024"/> 
Paciente:	<input type="text" value="cuzcano"/> 
<input type="button" value="Registrar Atención"/>	

Como se puede observar tras el ingreso y asignación del paciente se crea la cita del paciente

Registrar Atención

Cita:	<input type="text"/>
Descripción:	<input type="text"/>
Duración:	<input type="text"/>
Costo:	<input type="text" value="0,0"/>
Fecha:	<input type="text" value="dd/mm/aaaa"/>
Paciente:	<input type="text" value="Jarem Gallegos"/>
<input type="button" value="Registrar Atención"/>	

Descripción	Fecha	Paciente
Tos seca	2024-10-18	Jarem Gallegos
Después de las 12:00	2024-10-17	cuzcano
Después de las 11:00	2024-10-10	cuzcano

Paso 4: Registrar historial médico de pacientes por pacientes

Para poder realizar la gestión del historial médico del paciente, primeramente, se deberá de poder registrar un odontólogo y el diagnóstico.

Agregar Historial Odontológico

Descripción:	<input type="text" value="Problemas de caries en los dientes"/>
Fecha:	<input type="text" value="10/10/2024"/>
Paciente:	<input type="text" value="cuzcano"/>
Diagnóstico:	<input type="text" value="Caries Dentarias"/>
Odontólogo:	<input type="text" value="Dr Mirando Garcia"/>
Radiografía (URL):	<input type="text" value="https://www.bladegrup.com/wp-content/uploads/2015/08/rads.jpg"/>
<input type="button" value="Registrar Historial"/>	

Descripción	Fecha	Diagnóstico	Odontólogo	Radiografía	
Problemas en encías	2024-10-12	Sin Paciente	asas	Jarem Gallegos	<input type="button" value="Ver Radiografía"/>
Problemas en encías	2024-10-12	Jarem Gallegos	asas	Jarem Gallegos	<input type="button" value="Ver Radiografía"/>

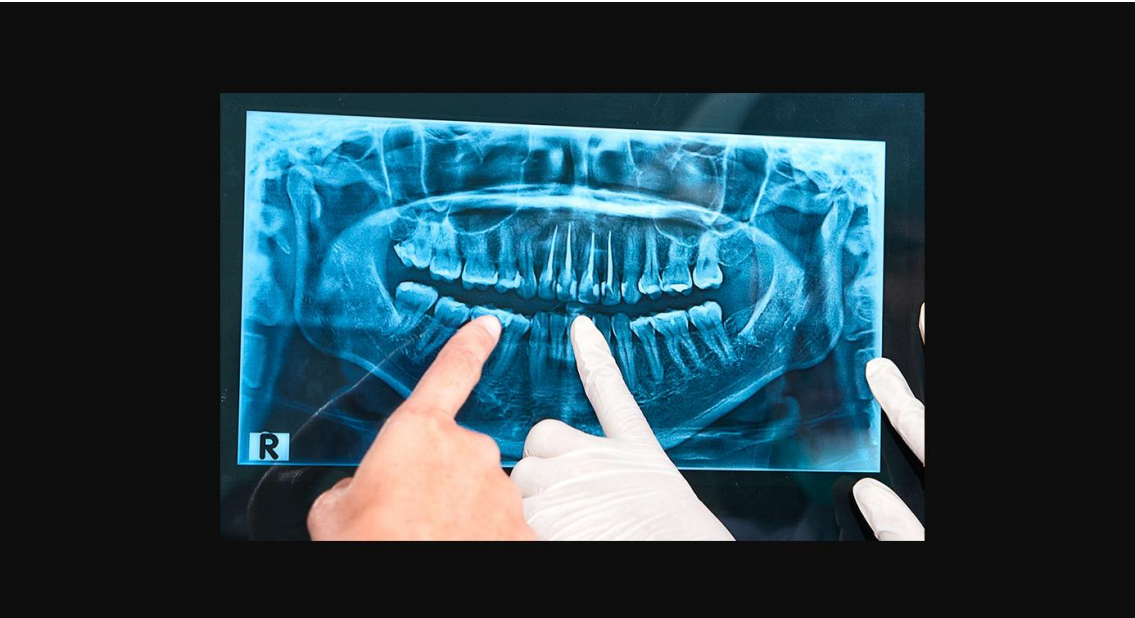
Lo que se pretende es poder registrar el historial médico del paciente al ingresar una descripción, seleccionar una fecha y asignar a un paciente, diagnóstico, odontólogo a cargo y la dirección de una radiografía

Registrar Historial

Descripción	Fecha	Paciente	Diagnóstico	Odontólogo	Radiografía
Problemas en encías	2024-10-12	Sin Paciente	asas	Jarem Gallegos	Ver Radiografía
Problemas en encías	2024-10-12	Jarem Gallegos	asas	Jarem Gallegos	Ver Radiografía
Problemas de caries en los dientes	2024-10-10	cuzcano	Caries Dentarias	Dr Mirando García	Ver Radiografía

© 2024 Gestión de Pacientes - Jarem Gallegos

Tras haber realizado esto, se podrá mostrar el registro del historial médico, juntamente también será capaz de poder ingresar a la radiografía.



[Extras]

Dado a que es necesario asignar un odontólogo y un diagnóstico, se crearon módulos para cada uno

Registrar Odontólogo

Nombre:

Especialidad:

Teléfono:

Correo:

Dirección:

Registrar

Nombre	Especialidad	Teléfono	Correo	Dirección	Acciones
Jarem Gallegos	Diente	946333421	jaremgallegosp@gmail.com	Urb. Pedro Diez Canseco	Ver
Dr Mirando García	Dentadura	9421233422	mirandagarcia@gmail.com	Urb. Miraflores	Ver

Registrar Diagnóstico

Descripción:

Fecha:

dd/mm/aaaa

Registrar

Lista de Diagnósticos

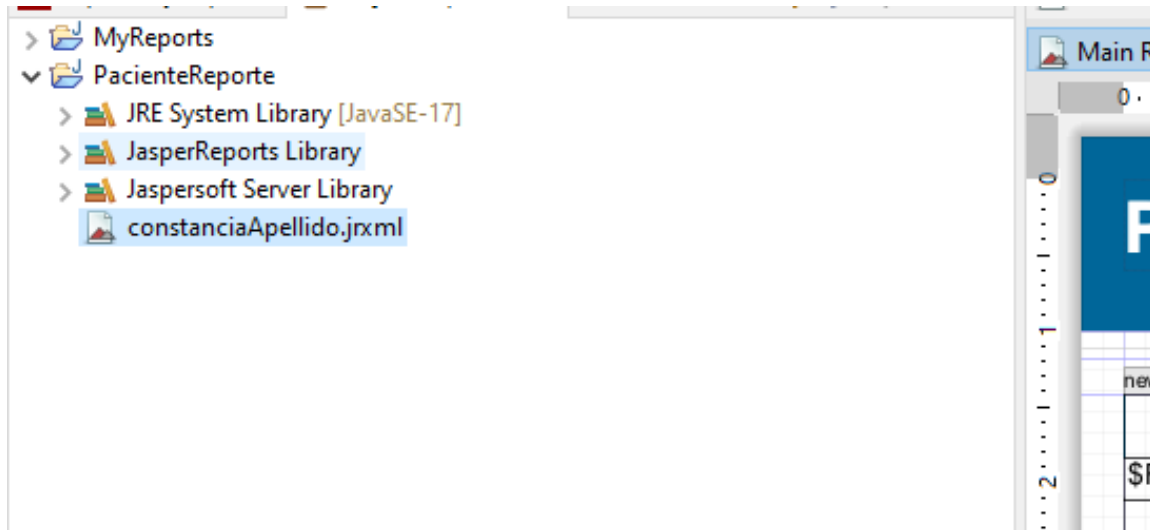
Descripción	Fecha	Paciente	Odontólogo	Tratamiento
asas	2024-10-11			
Caries Dentarias	2024-10-12			

© 2024 Gestión de Pacientes - Jarem Gallegos

- C. Al pulsar la opción **registrar** deberá guardar los datos de los pacientes y en caso de éxito, se debe generar una **constancia PDF** (constanciaApellido.jasper) del paciente registrado (mascota).

Paso 1: Creación de Reporte Jasper

Para poder realizar un reporte Jasper, primeramente, se debe conectar con la base de datos, tras la realización de este paso, se crea el proyecto Jasper y se integra la sentencia SQL para el listado de los pacientes (SELECT * FROM paciente)



Tras esto se diseñó los datos que se quiere poner en el reporte, en este caso se desea integrar los datos de ID, Nombre, Dirección y Correo

PACIENTE			
Constancia de Registro de Paciente			
new java.util.Date() "Page " + \$V{PAGE_NUMBER} + " of " + \$V			
ID	NOMBRE	DIRECCIÓN	CORREO
\$F{id}	\$F{nombre}	\$F{direccion}	\$F{correo}
Summary			

MySQL PDF Document (.pdf) Page 1 of 1 100%

PACIENTE

Constancia de Registro de Paciente

ID	NOMBRE	DIRECCIÓN	CORREO
1	Jarem Gallegos	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
2	Cuzcano	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
3	Gutierrez	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
4	Cuzano	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
5	Jarem Gallegos	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
6	Cuzcano	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
7	Felipe	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
8	Cuzcano	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
9	Cuzcano	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
10	Felipe	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
11	Jarem Gallegos	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com

Tras esto se copia el código XML generado del reporte diseñado, teniendo que integrarse dentro de un archivo plane para poder generar reportes

```

1 <!-- Created with JasperSoft Studio version 7.0.0.final using JasperReports Library version 7.0.0-b478feaa9aab4375eba71de77b4ca138ad21 -->
2 <jasperReport name="constanciaApellido" language="java" columnCount="1" pageWidth="595" pageHeight="842" orientation="Landscape" when
3   <property name="template.engine" value="tabular_template"/>
4   <property name="com.jaspersoft.studio.data.defaultdataadapter" value="MySQL"/>
5   <property name="com.jaspersoft.studio.data.sql.tables" value="" />
6   <property name="com.jaspersoft.studio.unit." value="pixel"/>
7   <property name="com.jaspersoft.studio.unit.pageHeight" value="pixel"/>
8   <property name="com.jaspersoft.studio.unit.pageWidth" value="pixel"/>
9   <property name="com.jaspersoft.studio.unit.topMargin" value="pixel"/>
10  <property name="com.jaspersoft.studio.unit.bottomMargin" value="pixel"/>
11  <property name="com.jaspersoft.studio.unit.leftMargin" value="pixel"/>
12  <property name="com.jaspersoft.studio.unit.rightMargin" value="pixel"/>
13  <property name="com.jaspersoft.studio.unit.columnWidth" value="pixel"/>
14  <property name="com.jaspersoft.studio.unit.columnSpacing" value="pixel"/>
15  <style name="Table">
16    <box>
17      <pen lineWidth="1.0" lineColor="#000000"/>
18      <topPen lineWidth="1.0" lineColor="#000000"/>
19      <leftPen lineWidth="1.0" lineColor="#000000"/>
20      <bottomPen lineWidth="1.0" lineColor="#000000"/>
21      <rightPen lineWidth="1.0" lineColor="#000000"/>
22    </box>
23  </style>
24  <style name="Table_TH" mode="Transparent" backgroundColor="#FFFFFF">
25    <box>
26      <pen lineWidth="0.5" lineColor="#000000"/>
27      <topPen lineWidth="0.5" lineColor="#000000"/>
28      <leftPen lineWidth="0.5" lineColor="#000000"/>
29      <bottomPen lineWidth="0.5" lineColor="#000000"/>
30      <rightPen lineWidth="0.5" lineColor="#000000"/>
31    </box>
32  </style>
33  <style name="Table_Ch" mode="Transparent" foreground="#8B9F7D" backgroundColor="#70A9C6">
34    <box>
35      <pen lineWidth="0.5" lineColor="#000000"/>
36      <topPen lineWidth="0.5" lineColor="#000000"/>
37      <leftPen lineWidth="0.5" lineColor="#000000"/>
38      <bottomPen lineWidth="0.5" lineColor="#000000"/>
39      <rightPen lineWidth="0.5" lineColor="#000000"/>
40    </box>
41  </style>
42  <style name="Table_TD" mode="Transparent" backgroundColor="#FFFFFF">
  
```

Para poder lograr esto, es necesario poder integrar las dependencias Jasper para poder generar documentos pdf del reporte e integrarlo en Spring

```

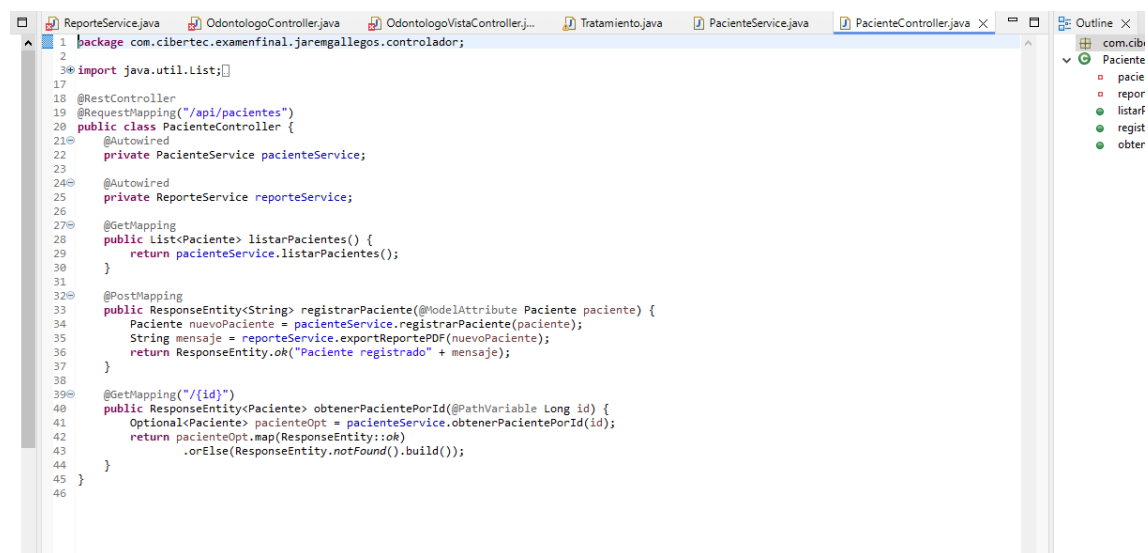
45 <dependency>
46   <groupId>org.projectlombok</groupId>
47   <artifactId>lombok</artifactId>
48   <version>1.18.30</version>
49   <scope>provided</scope>
50 </dependency>
51 <dependency>
52   <groupId>net.sf.jasperreports</groupId>
53   <artifactId>jasperreports</artifactId>
54   <version>7.0.1</version>
55 </dependency>
56 <dependency>
57   <groupId>net.sf.jasperreports</groupId>
58   <artifactId>jasperreports-functions</artifactId>
59   <version>7.0.1</version>
60 </dependency>
61 <dependency>
62   <groupId>net.sf.jasperreports</groupId>
63   <artifactId>jasperreports-pdf</artifactId>
64   <version>7.0.1</version>
65 </dependency>
  
```

Tras esto, es necesario poder crear la clase de ReporteService para poder integrarlo dentro del método post que realizará el registro del paciente.



```
1 package com.cibertec.examenfinal.jaremgallegos.servicio;
2
3 import com.cibertec.examenfinal.jaremgallegos.modelo.Paciente;
4
5 @Service
6 public class ReporteService {
7
8     public String exportReportePDF(Paciente paciente) {
9         try {
10             List<Paciente> pacientes = Collections.singletonList(paciente);
11             File archivo = ResourceUtils.getFile("classpath:constanciaApellido.jrxml");
12             JasperReport jasperReport = JasperCompileManager.compileReport(archivo.getAbsolutePath());
13             JRBeanCollectionDataSource dataSource = new JRBeanCollectionDataSource(pacientes);
14             Map<String, Object> parametros = new HashMap<>();
15             parametros.put("creadoPor", "Jarem Gallegos");
16             JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, parametros, dataSource);
17             String rutaArchivo = "D:\\CarpetaProtegida\\CPD-2\\Proyectos-2024\\jaremgallegos\\" + paciente.getId() + "_paciente.pdf";
18             JasperExportManager.exportReportToPdfFile(jasperPrint, rutaArchivo);
19
20             return "Reporte PDF generado exitosamente en: " + rutaArchivo;
21         } catch (Exception e) {
22             e.printStackTrace();
23             return "Error al generar el reporte PDF: " + e.getMessage();
24         }
25     }
26 }
```

Dado que tratamos de apartar la lógica de negocio con la vista, en nuestro controlador Rest establecemos la verificación de creación



```
1 package com.cibertec.examenfinal.jaremgallegos.controlador;
2
3 import java.util.List;
4
5 @RestController
6 @RequestMapping("/api/pacientes")
7 public class PacienteController {
8
9     @Autowired
10     private PacienteService pacienteService;
11
12     @Autowired
13     private ReporteService reporteService;
14
15     @GetMapping
16     public List<Paciente> listarPacientes() {
17         return pacienteService.listarPacientes();
18     }
19
20     @PostMapping
21     public ResponseEntity<String> registrarPaciente(@ModelAttribute Paciente paciente) {
22         Paciente nuevoPaciente = pacienteService.registrarPaciente(paciente);
23         String mensaje = reporteService.exportReportePDF(nuevoPaciente);
24         return ResponseEntity.ok("Paciente registrado" + mensaje);
25     }
26
27     @GetMapping("/{id}")
28     public ResponseEntity<Paciente> obtenerPacientePorId(@PathVariable Long id) {
29         Optional<Paciente> pacienteOpt = pacienteService.obtenerPacientePorId(id);
30         return pacienteOpt.map(ResponseEntity::ok)
31             .orElse(ResponseEntity.notFound().build());
32     }
33 }
```

Paso 2: Prueba de registro de paciente y generación de reporte

Para uso prácticos se integran los datos del paciente Jose Maria, para poder generar un reporte, al registrar, deberá mostrar un mensaje de éxito y mostrarnos la dirección en donde se guardó el reporte pdf.

PacientesHistorial OdontológicoAtenciones

Registrar Paciente

Nombre:
Jose Maria

Dirección:
Urb. Miraflores

Teléfono:
9421233422

Correo:
josemaria@gmail.com

Registrar

Nombre	Dirección	Teléfono	Correo	Acciones
Jarem Gallegos	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	Ver
Cuzcano	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	Ver
Gutierrez	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	Ver
Cuzano Gutierrez	Urb. Pedro Diez Canseco	946333420	jaremgallegosp@gmail.com	Ver

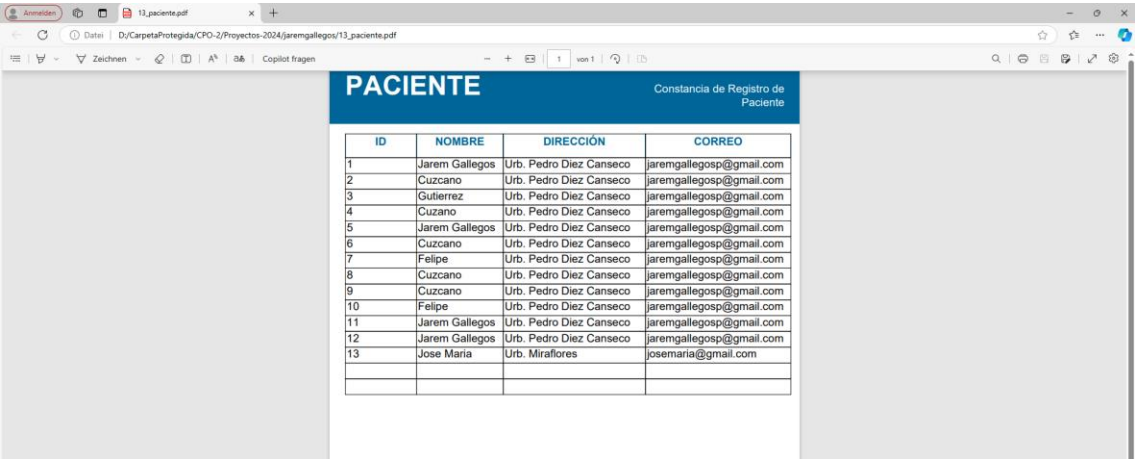
Se generó con éxito

Paciente registradoReporte PDF generado exitosamente en: D:\CarpetaProtegida\CPO-2\Proyectos-2024\jaremgallegos\13_paciente.pdf

Dirección del Reporte generado

	Name	Änderungsdatum	Typ	Größe
	.apt_generated	25.10.2024 06:18	Dateiordner	
	.apt_generated_tests	25.10.2024 06:18	Dateiordner	
	.git	25.10.2024 06:20	Dateiordner	
	.mvn	23.10.2024 22:20	Dateiordner	
	.settings	25.10.2024 05:07	Dateiordner	
	.vscode	24.10.2024 02:53	Dateiordner	
	src	23.10.2024 22:20	Dateiordner	
	target	23.10.2024 22:22	Dateiordner	
	.classpath	25.10.2024 06:18	CLASSPATH-Datei	3 KB
	.gitattributes	23.10.2024 22:20	Textdokument	1 KB
	.gitignore	23.10.2024 22:20	Textdokument	1 KB
	.project	25.10.2024 05:07	PROJECT-Datei	2 KB
	12_paciente.pdf	25.10.2024 06:31	Microsoft Edge P...	2 KB
	13_paciente.pdf	25.10.2024 06:35	Microsoft Edge P...	4 KB
	HELP.md	23.10.2024 22:20	Markdown-Quell...	2 KB
	mvnw	23.10.2024 22:20	Datei	11 KB
	mvnw.cmd	23.10.2024 22:20	Windows-Befehls...	7 KB
	pom.xml	25.10.2024 06:00	Microsoft Edge H...	3 KB

Tras haber generado el reporte con éxito se muestra los datos actualizados de la constancia de registro del paciente.



The screenshot shows a PDF document titled "PACIENTE" and "Constancia de Registro de Paciente". It contains a table with 13 rows of patient data. The table has four columns: ID, NOMBRE, DIRECCIÓN, and CORREO. The data is as follows:

ID	NOMBRE	DIRECCIÓN	CORREO
1	Jarem Gallegos	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
2	Cuzcano	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
3	Gutierrez	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
4	Cuzano	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
5	Jarem Gallegos	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
6	Cuzcano	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
7	Felipe	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
8	Cuzcano	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
9	Cuzcano	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
10	Felipe	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
11	Jarem Gallegos	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
12	Jarem Gallegos	Urb. Pedro Diez Canseco	jaremgallegosp@gmail.com
13	Jose Maria	Urb. Miraflores	josemaria@gmail.com

[Problema]
Al tratar de generar una constancia de registro personal, este no muestra los datos, generando un documento en blanco del paciente que se a registrado, por lo que se pretende poder corregir en ese sentido el uso de este reporte.



The screenshot shows a PDF document titled "PACIENTE" and "Constancia de Registro de Paciente". The table area is blank, indicating that the patient data is not being displayed correctly.

Rúbrica				
Pregunta	Excelente	Bueno	Regular	Deficiente
1	Implementa adecuadamente el proceso de registro de datos, muestra la constancia en PDF (18p) y El reporte gráfico (20 p)	Implementa los proceso de registro, listados con la estructura solicitada. (14 p)	Implementa sólo la página principal, y el modelo, repositorio; mostrando los datos de la tabla (10p)	No implementa ninguna funcionalidad.