Trabajo Fin de Máster:

***Desarrollo de una aplicación web para la administración de citas medicas***

|  |  |  |
| --- | --- | --- |
| Titulación:  XXXX  Curso académico  2023 – 2024 | Alumno/a: Fernandez Albarracin, Nilxon  D.N.I: 7631825  Director/a de TFM: XXXX | Convocatoria:  Primera o Segunda |

**Índice**

[1.Introduccion 1](#_Toc198839829)

[2.Motivacion 3](#_Toc198839830)

[3.Objetivos generales 3](#_Toc198839831)

[4.Objetivos especificos 4](#_Toc198839832)

[5.Marco Tecnologico 4](#_Toc198839833)

[5.1 Tecnologías Utilizadas en el Desarrollo Web 5](#_Toc198839834)

[6.Proceso de Desarrollo 6](#_Toc198839835)

[6.1 Requisitos 7](#_Toc198839836)

[6.1.1 Requerimientos Funcionales 7](#_Toc198839837)

[6.1.2 Requerimientos No Funcionales 8](#_Toc198839838)

[6.1.3 Modelo conceptual 8](#_Toc198839839)

[6.1.3.1 Modelo de estructura – Rol Administrador 8](#_Toc198839840)

[6.1.3.2 Modelo de estructura – Rol Empleado 8](#_Toc198839841)

[6.1.3.3 Modelo de estructura – Rol Medico 8](#_Toc198839842)

[6.2 Diseño de la base de datos 8](#_Toc198839843)

[6.2.1 Diseño conceptual 9](#_Toc198839844)

[6.2.2 Diseño logico 9](#_Toc198839845)

[6.2.3 Diseño fisico 10](#_Toc198839846)

[7.Conclusiones,recomendaciones y trabajos a future 18](#_Toc198839847)

[8.Bibliografia 18](#_Toc198839848)

[9.Anexos y Apendices 18](#_Toc198839849)

1.Introduccion

En la actualidad, la digitalización ha transformado numerosos sectores, incluyendo la salud, donde la optimización de procesos administrativos es necesario para mejorar la calidad del servicio. Sin embargo, muchas instituciones médicas aún dependen de métodos manuales o sistemas complejos para la gestión de citas, lo que genera problemas como largos tiempos de espera, dificultad en la asignación de horarios y una administración ineficiente de la disponibilidad médica.

Ante esta problemática, surge la necesidad de desarrollar soluciones tecnológicas que faciliten la planificación y reserva de citas médicas de manera eficiente y accesible. En este contexto, este Trabajo de Fin de Máster propone el diseño y desarrollo de una aplicación web para la administración de citas médicas en una entidad de salud, empleando frameworks modernos de desarrollo web y MySQL para la gestión de bases de datos.

El objetivo principal de este proyecto es optimizar la administración de la disponibilidad médica y las reservas de pacientes mediante una plataforma digital que permita la gestión eficiente de horarios y citas, mejorando la experiencia tanto para los profesionales de la salud como para los pacientes.

La aplicación desarrollada contará con funcionalidades como la gestión de usuarios (pacientes, médicos y administradores), la asignación de citas, el control de disponibilidad médica. Además, se implementarán medidas de seguridad garantizar un acceso seguro al sistema.

Este proyecto no solo responde a una necesidad actual en el sector salud, sino que también aporta una solución escalable y adaptable a diferentes entidades médicas. Su implementación contribuirá a mejorar la eficiencia operativa, reducir los tiempos de espera y optimizar el uso del tiempo de los profesionales de la salud, promoviendo una atención médica más organizada y accesible.

2.Motivacion

Debido a mi experiencia laboral en el ámbito de la salud y al interés que ha despertado en mí y en muchas entidades de salud, la administración de citas médicas sigue dependiendo de procesos manuales y esto genera problemas como largas esperas, sobrecarga en la atención y dificultad en el seguimiento de citas.

Deseo desarrollar este proyecto con la intención de que optimice este proceso, facilitando la gestión de citas tanto para los pacientes como para los profesionales de la salud, mejorando así la eficiencia operativa de las instituciones médicas, a través de un producto amigable con el usuario que le brinde una excelente experiencia y un uso agradable.

Producto(s) esperado (s) y aporte: Una aplicación que conste de los módulos para la Gestión de usuarios (pacientes, médicos, administradores).

3.Objetivos generales

Diseño y desarrollo de una aplicación web que administre la disponibilidad y reservas médicas en una entidad de salud, empleando diferentes frameworks de desarrollo web y mysql para manejo de bases de datos

4.Objetivos específicos

1. Analizar los requerimientos del sistema para definir las funcionalidades clave de la aplicación, incluyendo la gestión de usuarios, administración de citas médicas y disponibilidad de profesionales de la salud
2. Diseñar la arquitectura del sistema utilizando frameworks modernos de desarrollo web que permitan una estructura modular y escalable
3. Implementar la aplicación web utilizando Laravel como framework backend y Angular como front end para una interfaz intuitiva y amigable.
4. Desarrollar e integrar la base de datos en MySQL asegurando un modelo de datos eficiente para el almacenamiento y recuperación de información
5. **Garantizar la seguridad del sistema** mediante la implementación de autenticación, control de acceso y encriptación de datos sensibles.

5.Marco tecnológico

Los sistemas de gestión de citas médicas son plataformas digitales diseñadas para administrar la disponibilidad y reserva de consultas en instituciones de salud. Su objetivo principal es mejorar la eficiencia en la asignación de citas, reducir tiempos de espera y optimizar el uso de los recursos médicos. Este desarrollo requiere de la integración de múltiples tecnologías que nos permiten garantizar funcionalidad, escalabilidad, seguridad.

5.1 Tecnologías Utilizadas en el Desarrollo Web

El desarrollo de aplicaciones web para la gestión de citas médicas requiere la combinación de tecnologías frontend, backend y bases de datos para garantizar un sistema robusto y escalable.

**1. Frameworks de Desarrollo Web**

Los frameworks de desarrollo web permiten agilizar la creación de aplicaciones mediante estructuras predefinidas y herramientas avanzadas. Para este proyecto, se utilizarán:

* **Laravel:** Framework PHP para el desarrollo backend, que ofrece seguridad, escalabilidad y una estructura modular, que sigue el patrón MVC (Modelo-Vista-Controlador). su sistema de rutas y su potente ORM llamado Eloquent.
* **Angular:** Para la construcción del frontend, proporcionando una interfaz dinámica y responsiva.

**2. Base de Datos y Gestión de Información**

La base de datos es un componente clave para almacenar información de usuarios, citas y disponibilidad médica. Se empleará **MySQL**, un sistema de gestión de bases de datos relacional que permite manejar grandes volúmenes de información de manera eficiente.

**3. Control de Versiones Git + GitHub**

Git será la herramienta principal de control de versiones, permitiendo un desarrollo colaborativo y organizado. GitHub servirá como repositorio remoto para el almacenamiento seguro del código fuente.

**4 Entorno de Desarrollo**

* **Visual Studio Code:** Editor de código ligero y potente, compatible con múltiples extensiones para PHP, Laravel, Angular, entre otros.
* **Composer:** Gestor de dependencias para PHP/Laravel.
* **Node.js y NPM:** Necesarios para la gestión de paquetes frontend (CSS, JS, Vue, etc.).

**5. Entorno de Ejecución y Despliegue**

* **XAMPP/Laragon (entorno local):** Herramientas para levantar el servidor local con Apache, MySQL y PHP.

**6. Seguridad y Autenticación**

* **Laravel Sanctum:** Mecanismos para implementar autenticación de usuarios de forma segura y flexible.

2.7 Librerías y Complementos

* Angular Material es una biblioteca de componentes de interfaz de usuario (UI) de código abierto desarrollada por Google para ser utilizada con el framework de desarrollo web Angular.
* ngx-toastr es una librería para angular que permite crear notificaciones y alertas al usuario final

6.Proceso de Desarrollo

6.1 Requisitos

6.1.1 Requerimientos Funcionales

1. El sistema administrador de citas permitirá al usuario hacer login y acceder a la plataforma introduciendo su nombre de usuario y su contraseña.

2. El sistema administrador de citas almacenará para todos los clientes registrados: nombre de usuario, contraseña, DNI, nombre, apellidos, genero, celular, ocupación, dirección, correo electrónico y fecha de nacimiento.

3. El sistema administrador de citas proporcionará un mecanismo al usuario para que pueda registrarse en la plataforma ingresando: nombre de usuario, contraseña, DNI, nombre, apellidos, genero, celular, ocupación, dirección, correo electrónico y fecha de nacimiento.

4. El sistema administrador de citas permitirá al usuario empleado de la clínica (recepcionista), reservar una cita seleccionando el día y médicos disponibles.

5. El sistema administrador de citas permitirá al usuario empleado de la clínica (recepcionista), reservar una cita seleccionando el médico y su disponibilidad.

6. El sistema administrador de citas permitirá al usuario empleado de la clínica (recepcionista), crear la disponibilidad del médico.

7. El sistema administrador de citas permitirá usuario empleado de la clínica (recepcionista), cancelar las citas de médica.

8. El sistema administrador de citas proporcionará, al usuario médico de la clínica, un mecanismo para visualizar sus citas médicas del día

6.1.2 Requerimientos No Funcionales

RNF.1. El sistema debe validar que los datos nombre usuario y contraseña son correctos.

RNF.2. El sistema debe utilizar el encriptado a 128 bits para almacenar los datos nombre usuario y contraseña

RNF.3. El sistema debe ser responsive

6.1.3 Modelo conceptual

Para mostrar de forma fácil y mejor comprensión de la aplicación web en los diferentes roles; se presentó un modelo de estructura (Bocetos). Para ello utilizamos la herramienta “Diagram.drawio”.

6.1.3.1 Modelo de estructura – Rol Administrador

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

6.1.3.2 Modelo de estructura – Rol Empleado

Interfaz de usuario gráfica, Texto, Aplicación

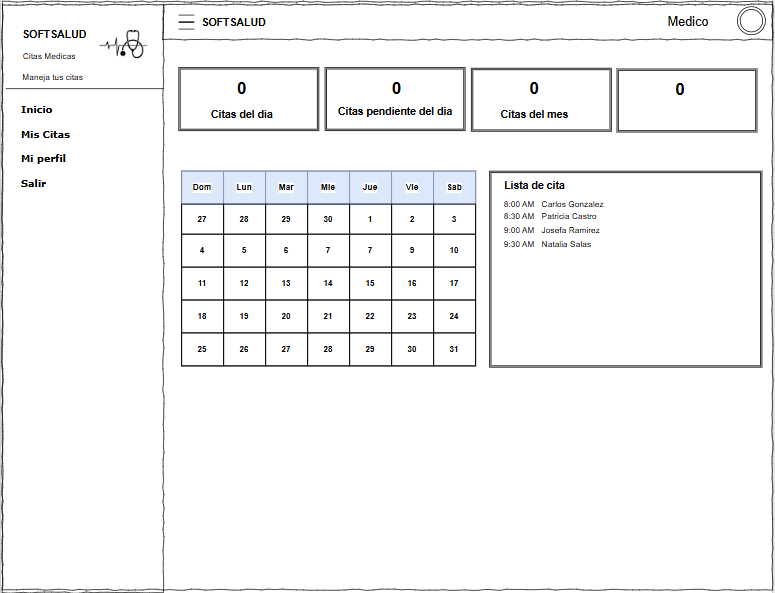
El contenido generado por IA puede ser incorrecto.

6.1.3.3 Modelo de estructura - Rol Medico

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

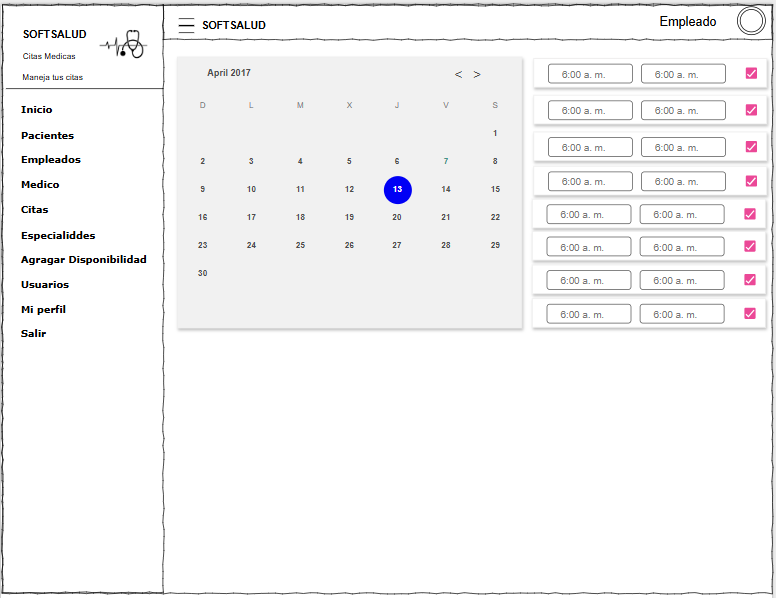
6.1.3.4 Modelo de Estructura – Mi Citas



6.1.3.5 Modelo de Estructura - Asignación de Citas Por Especialidad

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

6.1.3.6 Modelo de Estructura-Asignar disponibilidad

6.2 Diseño de la base de datos

La base de datos ha sido diseñada siguiendo una estructura relacional. A continuación, se presentan las principales entidades:

Users: almacena información general del usuario (nombre, correo, rol).

Patients: datos específicos del paciente.

Doctors: información del personal médico (especialidad, horarios).

Appointmens: representa las reservas médicas entre pacientes y profesionales.

Availabilities: define los días y horarios habilitados por cada profesional.

Cada entidad está relacionada mediante claves primarias y foráneas, asegurando la integridad de los datos. El modelo de datos ha sido representado gráficamente a través de un diagrama entidad-relación (ER), que detalla las tablas, atributos y relaciones, y ha sido implementado mediante migraciones en Laravel.

**6.2.1 Diseño conceptual**

Diagrama

El contenido generado por IA puede ser incorrecto.

6.2.2 Diseño logico

appointments (id,date\_appointments,start\_hour,doctor\_id,patient\_id,availabily\_id)CP:{id} Caje: appointments.doctor\_id es clave ajena doctors.id ¿NULOS?NO Actualizacion PROPAGAR Borrar:PROPAGAR Caje2: appointments.patient\_id es clave ajena patients.id ¿NULOS?NO Actualizacion PROPAGAR Borrar:PROPAGAR Caje3: appointments.availabily\_id es clave ajena availabilities.id ¿NULOS?NO Actualizacion PROPAGAR Borrar:PROPAGAR

availabilities(id,doctor\_id,date\_availabilities,start\_hour,end\_hour,status)CP:{id} Caje: availabilities.doctor\_id es clave ajena doctors.id ¿NULOS?NO Actualizacion PROPAGAR Borrar:PROPAGAR

doctors(id,name,surname,dni,genre,photo,phone,email,password,user\_id,specialty\_id)CP:{id} Caje: doctors.user\_id es clave ajena users.id ¿NULOS?NO Actualizacion PROPAGAR Borrar:PROPAGAR Caje2: doctors.specialty\_id es clave ajena specialties.id ¿NULOS?NO Actualizacion PROPAGAR Borrar:PROPAGAR

employees (id,name,surname,dni,occupation,gross\_salary,email,password,genre,photo,phone,user\_id)CP:{id} Caje: employees.user\_id es clave ajena users.id ¿NULOS?NO Actualizacion PROPAGAR Borrar:PROPAGAR

medical\_offices(id,name,estado)CP:{id}

patients(id,name,surname,dni,genre,photo,occupation,phone,email,password,user\_id)CP:{id} Caje: patients.user\_id es clave ajena users.id ¿NULOS?NO Actualizacion PROPAGAR Borrar:PROPAGAR

specialties (id,name)CP:{id}

users (id,email,password,role,name,surname)

6.2.3 Diseño fisico

CREATE TABLE `appointments` (

`id` bigint(20) UNSIGNED NOT NULL,

`date\_appointments` date NOT NULL,

`start\_hour` varchar(255) NOT NULL,

`doctor\_id` bigint(20) UNSIGNED NOT NULL,

`patient\_id` bigint(20) UNSIGNED NOT NULL,

`availabily\_id` bigint(20) UNSIGNED NOT NULL,

`status` varchar(255) NOT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE `availabilities` (

`id` bigint(20) UNSIGNED NOT NULL,

`doctor\_id` bigint(20) UNSIGNED NOT NULL,

`date\_availabilities` date NOT NULL,

`start\_hour` varchar(255) NOT NULL,

`end\_hour` varchar(255) NOT NULL,

`status` tinyint(1) NOT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE `doctors` (

`id` bigint(20) UNSIGNED NOT NULL,

`name` varchar(255) NOT NULL,

`surname` varchar(255) NOT NULL,

`dni` varchar(255) NOT NULL,

`genre` varchar(255) NOT NULL,

`photo` varchar(255) NOT NULL,

`phone` varchar(255) NOT NULL,

`birthdate` date NOT NULL,

`address` varchar(255) NOT NULL,

`email` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`user\_id` bigint(20) UNSIGNED NOT NULL,

`specialty\_id` bigint(20) UNSIGNED NOT NULL,

`remember\_token` varchar(100) DEFAULT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE `employees` (

`id` bigint(20) UNSIGNED NOT NULL,

`name` varchar(255) NOT NULL,

`surname` varchar(255) NOT NULL,

`dni` int(11) NOT NULL,

`occupation` varchar(255) NOT NULL,

`gross\_salary` double NOT NULL,

`email` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`genre` varchar(255) NOT NULL,

`photo` varchar(255) NOT NULL,

`phone` varchar(255) NOT NULL,

`birthdate` date NOT NULL,

`address` varchar(255) NOT NULL,

`user\_id` bigint(20) UNSIGNED NOT NULL,

`remember\_token` varchar(100) DEFAULT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE `patients` (

`id` bigint(20) UNSIGNED NOT NULL,

`name` varchar(255) NOT NULL,

`surname` varchar(255) NOT NULL,

`dni` varchar(255) NOT NULL,

`genre` varchar(255) NOT NULL,

`photo` varchar(255) NOT NULL,

`occupation` varchar(255) NOT NULL,

`phone` varchar(255) NOT NULL,

`birthdate` date NOT NULL,

`address` varchar(255) NOT NULL,

`email` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`user\_id` bigint(20) UNSIGNED NOT NULL,

`remember\_token` varchar(100) DEFAULT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE `specialties` (

`id` bigint(20) UNSIGNED NOT NULL,

`name` varchar(255) NOT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE `users` (

`id` bigint(20) UNSIGNED NOT NULL,

`email` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`role` varchar(255) NOT NULL,

`name` varchar(255) NOT NULL,

`surname` varchar(255) NOT NULL,

`remember\_token` varchar(100) DEFAULT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

ALTER TABLE `appointments`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `appt\_date\_medical\_start\_unique` (`date\_appointments`,`doctor\_id`,`start\_hour`),

ADD KEY `appointments\_doctor\_id\_foreign` (`doctor\_id`),

ADD KEY `appointments\_patient\_id\_foreign` (`patient\_id`),

ADD KEY `appointments\_availabily\_id\_foreign` (`availabily\_id`);

ALTER TABLE `availabilities`

ADD PRIMARY KEY (`id`),

ADD KEY `availabilities\_doctor\_id\_foreign` (`doctor\_id`);

ALTER TABLE `doctors`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `doctors\_dni\_unique` (`dni`),

ADD UNIQUE KEY `doctors\_email\_unique` (`email`),

ADD KEY `doctors\_user\_id\_foreign` (`user\_id`),

ADD KEY `doctors\_specialty\_id\_foreign` (`specialty\_id`);

ALTER TABLE `employees`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `employees\_dni\_unique` (`dni`),

ADD UNIQUE KEY `employees\_email\_unique` (`email`),

ADD KEY `employees\_user\_id\_foreign` (`user\_id`);

ALTER TABLE `patients`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `patients\_dni\_unique` (`dni`),

ADD UNIQUE KEY `patients\_email\_unique` (`email`),

ADD KEY `patients\_user\_id\_foreign` (`user\_id`);

ALTER TABLE `specialties`

ADD PRIMARY KEY (`id`);

ALTER TABLE `appointments`

MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT;

ALTER TABLE `availabilities`

MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT;

ALTER TABLE `doctors`

MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=2;

ALTER TABLE `employees`

MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

ALTER TABLE `failed\_jobs`

MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT;

ALTER TABLE `jobs`

MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT;

ALTER TABLE `medical\_offices`

MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT;

ALTER TABLE `migrations`

MODIFY `id` int(10) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=12;

ALTER TABLE `patients`

MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT;

ALTER TABLE `personal\_access\_tokens`

MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

ALTER TABLE `specialties`

MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=2;

ALTER TABLE `users`

MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=10;

ALTER TABLE `appointments`

ADD CONSTRAINT `appointments\_availabily\_id\_foreign` FOREIGN KEY (`availabily\_id`) REFERENCES `availabilities` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `appointments\_doctor\_id\_foreign` FOREIGN KEY (`doctor\_id`) REFERENCES `doctors` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `appointments\_patient\_id\_foreign` FOREIGN KEY (`patient\_id`) REFERENCES `patients` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `availabilities`

ADD CONSTRAINT `availabilities\_doctor\_id\_foreign` FOREIGN KEY (`doctor\_id`) REFERENCES `doctors` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `doctors`

ADD CONSTRAINT `doctors\_specialty\_id\_foreign` FOREIGN KEY (`specialty\_id`) REFERENCES `specialties` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `doctors\_user\_id\_foreign` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `employees`

ADD CONSTRAINT `employees\_user\_id\_foreign` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `patients`

ADD CONSTRAINT `patients\_user\_id\_foreign` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

COMMIT;

6.3 Documentación del API back end

Partiendo del concepto framework para backend: conjunto de herramientas y bibliotecas que simplifican el desarrollo de las aplicaciones web proporcionando estructuras para su organización. Este conjunto de herramientas suele incluir funcionalidades y componentes esenciales como integración de bases de datos, autenticación de usuarios y funcionalidades de seguridad facilitando así las tareas del desarrollador para que pueda centrarse en la lógica específica de la aplicación. Los frameworks incorporan unas estructuras definidas que permiten un orden y un control del código de los archivos, la mayoría conteniendo utilidades para la arquitectura de Desarrollo MVC (Modelo, Vista Controlador).

En concreto, el framework que se va a utilizar para este proyecto es Laravel, uno de los frameworks de PHP más demandados por las empresas digitales. Éste se traba de uno de los frameworks más jóvenes y más fáciles de usar, caracterizado por su gran potencia y simplicidad.

Una vez definidos las tabla y campos a utilizar en el proyecto se crearan las modelos, migraciones y controladores por medio de los siguientes comandos:

php artisan make:model employee -mcr

php artisan make:model patient -mcr

php artisan make:model doctor -mcr

php artisan make:model specialty -mcr

php artisan make:model medical\_office -mcr

php artisan make:model appointment -mcr

php artisan make:model availability -mcr

php artisan make:controller AuthController

Para cada una de las entidades del proyecto, se han definido en el archivo api.php los siguientes métodos:

• GET: Obtiene los datos. Se llamará a una función donde se hará una SELECT en la base de datos que se utilice.

• POST: Se guarda un nuevo dato. Se llamará a una función donde se hará un INSERT en la base de datos que se utilice.

• PATCH: Modificación parcial de un recurso. Se llamará a una función donde se hará un UPDATE en la base de datos que se utilice.

• DELETE: Se borra un dato especificado por parámetro. Se llamará a una función donde se hará un DELETE en la base de datos que se utilice.

También utilizamos **Laravel Sanctum**, un paquete de autenticación liviano para API en Laravel que se instala por medio del siguiente comando:

**composer require laravel/sanctum**

Este comando instala el paquete **Sanctum** en el proyecto Laravel mediante Composer **y**  proporciona un sistema de autenticación basado en tokens API.

php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"

Este comando publica los archivos de configuración del paquete Sanctum en el proyecto,

al ejecutarlo, Laravel copiará la configuración predeterminada de Sanctum a config/sanctum.php.

7.Conclusiones, recomendaciones y trabajos a future

8.Bibliografia

9.Anexos y Apendices