Trabajo Fin de Máster:

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

|  |  |  |
| --- | --- | --- |
| Titulación:  Máster Univ. en Desarrollo de  Aplicaciones y Servicios Web  Curso académico  2023 – 2024 | Alumno/a: Fernandez Albarracin, Nilxon  D.N.I: 7631825  Director/a de TFM: Francisco Gómez | Convocatoria:  Primera o Segunda |

**Índice**

[1.Introduccion 1](#_Toc199361620)

[2.Motivacion 2](#_Toc199361621)

[3.Objetivos general 3](#_Toc199361622)

[4.Objetivos especificos 3](#_Toc199361623)

[5.Marco Tecnologico 3](#_Toc199361624)

[5.1 Tecnologías Utilizadas en el Desarrollo Web 4](#_Toc199361625)

[6.Proceso de Desarrollo 5](#_Toc199361626)

[6.1 Requisitos 5](#_Toc199361627)

[6.1.1 Requerimientos Funcionales 5](#_Toc199361628)

[6.1.2 Requerimientos No Funcionales 6](#_Toc199361629)

[6.1.3 Modelo conceptual 6](#_Toc199361630)

[6.1.3.1 Modelo de estructura – Rol Administrador 7](#_Toc199361631)

[6.1.3.2 Modelo de estructura – Rol Empleado 7](#_Toc199361632)

[6.1.3.3 Modelo de estructura - Rol Medico 7](#_Toc199361633)

[6.1.3.4 Modelo de Estructura – Mi Citas 8](#_Toc199361634)

[6.1.3.5 Modelo de Estructura - Asignación de Citas Por Especialidad 9](#_Toc199361635)

[6.1.3.6 Modelo de Estructura-Asignar disponibilidad 9](#_Toc199361636)

[6.2 Diseño de la base de datos 10](#_Toc199361637)

[6.2.1 Diseño conceptual 10](#_Toc199361638)

[6.2.2 Diseño logico 10](#_Toc199361639)

[6.2.3 Diseño fisico 12](#_Toc199361640)

[6.3 Documentación del API back end 18](#_Toc199361641)

[6.3.1 Crear conexión base de datos 18](#_Toc199361642)

[6.3.2 Rutas 20](#_Toc199361643)

[6.4 Documentación Front end 21](#_Toc199361644)

[7.Conclusiones,recomendaciones y trabajos a future 22](#_Toc199361645)

[8.Bibliografia 22](#_Toc199361646)

[9.Anexos y Apendices 23](#_Toc199361647)

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 general

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 Tecnologico

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 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 framework es un conjunto de herramientas, código y estructuras predefinidas que facilitan el desarrollo de software. Para este proyecto, se utilizarán:

* **Laravel:** Laravel es un framework open-source creado con PHP, ideal para desarrollar aplicaciones web que sean potentes y escalables. Como está basado en la arquitectura MVC, el código se organiza de manera que separa claramente la lógica de la aplicación, la interfaz de usuario y el manejo de los datos. Este framework también te ofrece soluciones preconfiguradas para tareas comunes, como la autenticación de usuarios, la gestión de bases de datos, y la creación de APIs REST.
* **Angular:** Angular es un Framework de JavaScript de código abierto escrito en TypeScript. Su objetivo principal es desarrollar aplicaciones de una sola página. Google se encarga del mantenimiento y constantes actualizaciones de mejoras para este framework.

**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**,

* **MySQL** es un sistema de gestión de bases de datos relacionales (RDBMS) de código abierto que se utiliza para almacenar y gestionar datos. Su fiabilidad, rendimiento, escalabilidad y facilidad de uso hacen de MySQL una opción popular para los desarrolladores.

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

* Git será la herramienta principal de control de versiones, permitiendo un desarrollo colaborativo y organizado.

Git es un sistema de control de versiones de código fuente distribuido, diseñado para gestionar el desarrollo de software de manera colaborativa y eficiente. Git se utiliza para realizar un seguimiento de los cambios realizados en el código fuente a lo largo del tiempo lo que permite a los desarrolladores trabajar en equipo en el mismo proyecto, en paralelo, sin perder información.

* GitHub servirá como repositorio remoto para el almacenamiento seguro del código fuente.

GitHub es un servicio basado en la nube que aloja un sistema de control de versiones (VCS) llamado Git. Éste permite a los desarrolladores colaborar y realizar cambios en proyectos compartidos, a la vez que mantienen un seguimiento detallado de su progreso.

**4 Entorno de Desarrollo**

* **Visual Studio Code:** es un editor de código para programadores gratuito, de código abierto y multiplataforma. Está desarrollado por Microsoft, una compañía con una dilatada experiencia en la creación de IDEs (entornos de desarrollo integrados), que ha conseguido plasmar su larga tradición en el sector para ofrecer una herramienta ligera y práctica que la comunidad ha adoptado en masa.
* **Composer:** es un sistema de gestión de dependencias específico para PHP. Fue lanzado en 2012 por Nils Adermann y Jordi Boggiano, inspirándose en herramientas como npm y Bundler. Su objetivo es claro: facilitar la inclusión de librerías de terceros en cualquier proyecto PHP.
* **NPM:** es un gestor dependencias de JavaScript, de hecho, es el gestor por defecto de todo el ecosistema de NodeJS. Viene empaquetado con NodeJS, por lo que en cuanto lo instalemos, de cualquiera de las formas posibles, vamos a tener **automáticamente instalado NPM.**

**5. Entorno de Ejecución y Despliegue**

* **XAMPP (entorno local):** es un paquete de software gratuito y de código abierto que proporciona un entorno de servidor local para el desarrollo web. El nombre XAMPP representa los componentes principales: X (para cualquier sistema operativo), Apache, MySQL, PHP, y Perl. Este conjunto de herramientas permite a los desarrolladores crear y probar sus sitios web de manera eficiente antes de llevarlos a un entorno de producción.

**6. Seguridad y Autenticación**

* **Laravel Sanctum:** es una API paquete de autenticación para Laravel las aplicaciones, proporcionando un peso ligero, sencillo de usar un sistema de autenticación para una sola página de aplicaciones (SPAs), aplicaciones para móviles, y otras API. Ofrece token de autenticación basada en el uso de JSON Web de Tokens (JWT) o API tokens, la habilitación de la autenticación segura sin la sobrecarga de sesión tradicional basado en la autenticación. Sanctum simplifica la configuración de token de autenticación, permitiendo a los desarrolladores centrarse en la construcción de sus aplicaciones en lugar de lidiar con la autenticación de complejidades.

2.7 Librerías y Complementos

* Angular Material es un sistema de diseño UI diseñada por Google, basado en la guía de diseño de Material Design, la cual proporciona una amplia variedad de componentes listos para integrarse perfectamente con cualquier proyecto de Angular.
* NGX-TOASTR es una biblioteca de notificaciones de Toast para Angular. Es una herramienta fácil de usar que se utiliza para mostrar notificaciones en pantalla en una aplicación Angular. Proporciona una interfaz fácil de usar para personalizar la apariencia de las notificaciones y controlar su comportamiento. Puede utilizarse para mostrar mensajes de éxito, error, advertencia, etc.

Con ngx-toastr, se puede personalizar la posición donde se mostrarán las notificaciones, así como el tiempo de duración. Además, se pueden crear notificaciones con diferentes niveles de importancia, como “información”, “éxito”, “error” y “advertencia”. También se pueden configurar las notificaciones para que se cierren automáticamente después de un tiempo específico o para que el usuario las cierre manualmente.

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

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

RNF.E.S.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

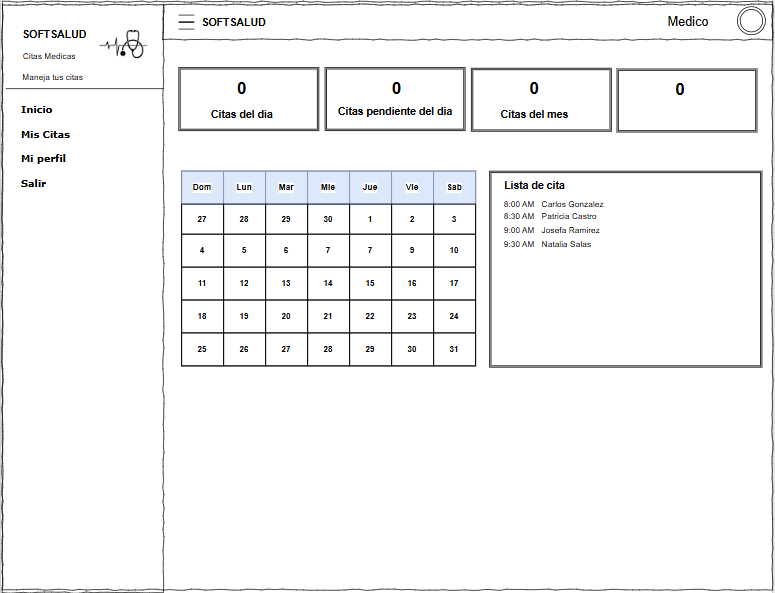
El contenido generado por IA puede ser incorrecto.

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

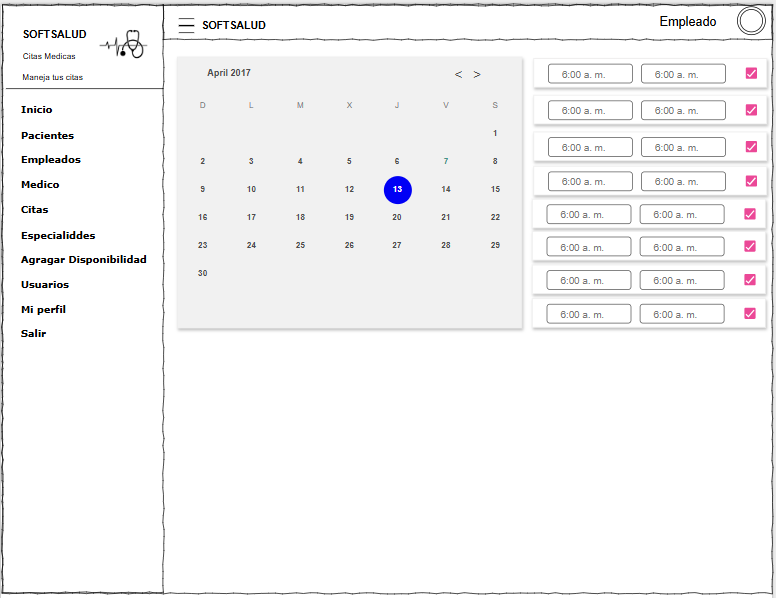


6.1.3.5 Modelo de Estructura - Asignación de Citas Por Especialidad

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 `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

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 es uno de los frameworks más jóvenes y más fáciles de usar, caracterizado por su gran potencia y simplicidad.

6.3.1 Crear conexión base de datos

Para crear nuestra base de datos debemos ir a la raíz del proyecto, en el archivo llamado ”.env” y utilizar los siguientes parámetros:

Texto

El contenido generado por IA puede ser incorrecto.

• “DB\_CONNECTION”: - Indica que la aplicación usará MySQL como motor de base de datos.

• “DB\_HOST”: Define la dirección del servidor de la base de datos.

• “DB\_PORT”: Especifica el puerto en el que MySQL escucha conexiones.

• “DB\_DATABASE”: Es el nombre de la base de datos utilizada en la aplicación.

• “DB\_USERNAME”: Define el usuario de MySQL con el que se conecta la aplicación.

• “DB\_PASSWORD”: Aquí se coloca la contraseña del usuario.

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

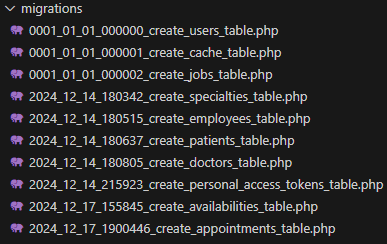
php artisan make:model medical\_office -mcr

php artisan make:model appointment -mcr

php artisan make:model availability -mcr

php artisan make:controller AuthController

Mediante las cuales se crea el modelo, el controlador y la migración para crear las tablas entidad:

Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

Una vez creada las migraciones, y terminada de llenar los campos que debe llevar tabla. Se ejecuta el siguiente comando para crearla en la base de datos

**php artisan migrate –seed**

Con este comando ya tenemos creadas nuestras tablas

6.3.2 Rutas

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.

En nuestro sistema cada una de las rutas de nuestras entidades, están representada de la siguiente forma:

Texto

El contenido generado por IA puede ser incorrecto.

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.

6.4 Implementacion Front end

Para crear el Proyecto en angular se utiliza la siguiente instrucccion:

**ng new softsalud**

* creación de componentes de paginas

ng generate component pages/login

ng generate component pages/dashboard

ng generate component pages/employees

ng generate component pages/patients

ng generate component pages/doctors

ng generate component pages/specialties

ng generate component pages/appointment

Estos comandos crean **componentes visuales** dentro del directorio pages/. Cada uno representa una vista o sección de tu aplicación, por ejemplo:

* login: pantalla de inicio de sesión.
* dashboard: panel principal tras iniciar sesión.
* employees, patients, doctors, specialties, medical\_office, appointment: páginas para gestionar cada entidad del sistema.

Estos componentes se usan para estructurar la interfaz del usuario.

* Generación de servicios

ng generate service services/auth

ng generate service services/employee

ng generate service services/doctor

ng generate service services/specialty

ng generate service services/patient

ng generate service services/appointment

ng generate service services/availability

Estos comandos crean **servicios Angular**, que son clases para manejar la lógica de negocio y las peticiones HTTP al backend (por ejemplo, obtener pacientes, crear doctores, etc.). Permiten mantener la lógica separada de la interfaz

* Generación de interceptor y guardia

ng generate interceptor interceptors/auth

Un **interceptor** modifica todas las peticiones HTTP salientes, por ejemplo, agregando el token a los encabezados de autenticación

ng generate guard guards/auth

Un guard (guardián) protege rutas. Por ejemplo, impide que usuarios no autenticados accedan al dashboard o a las páginas de gestión.

* Explicación de las rutas en angular

{ path: "", redirectTo: '/login', pathMatch: 'full' }

Redirecciona automáticamente la raíz del sitio (/) a la ruta /login.

{ path: 'login', component: LoginComponent }

Muestra el componente de inicio de sesión cuando el usuario visita /login.

Las siguientes rutas están **protegidas** con canActivate: [AuthGuard], lo que significa que sólo usuarios autenticados pueden acceder:

{ path: “dashboard”, component: DashboardComponent, canActivate: [AuthGuard] }

{ path: "patients", component: PatientsComponent, canActivate: [AuthGuard] }

{ path: "employees", component: EmployeesComponent, canActivate: [AuthGuard] }

{ path: "appointments", component: AppointmentComponent, canActivate: [AuthGuard] }

{ path: "appointmentsbydoctor", component: AppointmentByDoctorComponent, canActivate: [AuthGuard] }

{ path: "specialty", component: SpecialtiesComponent, canActivate: [AuthGuard] }

{ path: "doctors", component: DoctorsComponent, canActivate: [AuthGuard] }

{ path: "availability", component: AvailabilityComponent, canActivate: [AuthGuard] }

Cada ruta representa una sección de administración de la aplicación: pacientes, empleados, citas, especialidades, médicos y disponibilidad

{ path: 'availability\_details/:id', component: AvailabilityDetailsComponent, canActivate: [AuthGuard] }

Esta ruta usa un **parámetro dinámico** (:id) para mostrar detalles de disponibilidad según el médico seleccionado

7.Conclusiones,recomendaciones y trabajos a future

8.Bibliografia

Diseño web illusion Studio, ( 2024, marzo 21), ¿Qué es un framework? de https://www.illusionstudio.es/que-es-un-framework-web

Alberto Casero, (2024,octubre 21), ¿Qué es laravel? de

https://keepcoding.io/blog/que-es-laravel-y-como-dominarlo/

Manuel José Gonçalves,(2021,octubre 13), ¿Qué es angular?

https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/

Jeffrey Erickson, (2024, agosto 29) ¿Qué es Mysql? de

https://www.oracle.com/ar/mysql/what-is-mysql/

Fernando Ramirez, (2023, mayo 08) ¿Qué es git, para que se usa en el desarrollo de software? de

https://itsoftware.com.co/content/git-que-es-para-que-se-usa-en-el-desarrollo-de-software/

Gustavo B, (2025, marzo 13), ¿Qué es GitHub? de

https://www.hostinger.com/es/tutoriales/que-es-github

Fernán García de Zúñiga, (2024, octubre 30), ¿Qué es Visual Studio Code (VS Code)? de

https://www.arsys.es/blog/que-es-visual-studio-code-y-cuales-son-sus-ventajas

Nacho Morato, (2025, abril 30), ¿Qué es exactamente Composer? de

https://www.drupalweb.com/que-es-composer

Pablo Fernandez, (2019, noviembre 06), ¿Qué es NPM? de

https://openwebinars.net/blog/que-es-npm/

Iván Vázquez, (2023, diciembre 25), XAMPP: Una Solución Integral para Desarrolladores de https://draftdesignweb.com/2023/12/25/xampp-una-solucion-integral-para-desarrolladores/

linuxitos (2025, Marzo 08), API REST Laravel 12 Sanctum Autenticación Autorización Bearer de https://blog.linuxitos.com/post/api-rest-laravel-12-sanctum-autenticacion-autorizacion-bearer

Leandro Matilla, (15 julio 2021), ¿Por qué deberías usar Angular Material? de

https://leandromatilla1.medium.com/por-qu%C3%A9-deber%C3%ADas-usar-angular-material-e37f333c26f

domini code, (2023, febrero 12), Instalar NGX-TOASTR de https://dominicode.com/instalar-ngx-toastr/

9.Anexos y Apendices