PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

CARRERA DE SISTEMAS DE INFORMACIÓN

Trabajo de Titulación

# Tema: Desarrollo de un Sistema de Gestión de Citas y Seguimiento de Pacientes para la Clínica Dental ARPUL

AUTOR:

ERICK ALEXANDER CASA ORTIZ

QUITO DM, MARZO DE 2024

# INDICE GENERAL

INDICE GENERAL ..... 2
INDICE DE TABLAS ..... 3
INDICE DE FIGURAS ..... 4
INDICE DE ANEXOS ..... 6
CAPITULO I – INTRODUCIÓN ..... 8
**1.1 Tema** ..... 8
**1.2 Justificación** ..... 8
**1.3 Planteamiento del Problema** ..... 8
**1.4 Objetivos** ..... 9
**1.4.1 General** ..... 9
**1.4.2 Específico** ..... 9
**1.5 Alcance** ..... 10
**1.6 Esquema de Contenidos Básicos** ..... 11
CAPITULO II – MARCO TEORICO ..... 13
**2.1 Metodología Ágil de Desarrollo Software** ..... 13
**2.1.1 Introducción al Modelo Ágil** ..... 13
**2.1.2 Scrum** ..... 14
**2.1.3 Roles** ..... 16
**2.1.4 Backlog** ..... 17
**2.1.5 Sprint** ..... 17
**2.2 Herramientas** ..... 18
**2.2.1 Bases de Datos** ..... 20
**2.2.2 Backend** ..... 21
**2.2.3 Frontend** ..... 22
CAPITULO III – PLANIFICACIÓN ..... 24
**3.1 Product Backlog** ..... 24
**3.2 Estrategia o Planificación** ..... 27
**3.2.1 Iteraciones** ..... 27
CAPITULO IV – DESARROLLO ..... 29
**4.1 Desarrollo Del Sprint 1** ..... 29
**4.1.1 Sprint Backlog 1** ..... 29
**4.1.2 Diseño 1** ..... 53
**4.1.3 Codificación 1** ..... 62
**4.1.4 Pruebas 1** ..... 65

**4.2 Desarrollo Del Sprint 2** ..... 66
**4.2.1 Sprint Backlog 2** ..... 66
**4.2.2 Diseño 2** ..... 82
**4.2.2 Codificación 2** ..... 85
**4.2.4 Pruebas 2** ..... 87
**4.3 Desarrollo Del Sprint 3** ..... 88
**4.3.1 Sprint Backlog 3** ..... 88
**4.3.2 Diseño 3** ..... 91
**4.3.3 Codificación 3** ..... 95
**4.3.4 Pruebas 3** ..... 99
CAPITULO V – CONCLUSIONES Y RECOMENDACIONES ..... 100
**5.1 Conclusiones** ..... 100
**5.2 Recomendaciones** ..... 101
**5.3 Anexos** ..... 102
BIBLIOGRAFÍA ..... 150

# INDICE DE TABLAS

**Tabla 1** Asignación de Roles en el Marco Scrum para el proyecto de la Clínica Dental ARPUL .16
**Tabla 2** Herramientas de Desarrollo Utilizadas en el Proyecto ..... 19
**Tabla 3** Product Backlog del Sprint 1 ..... 31
**Tabla 4** Excepciones en el proceso de Crear Roles y Permisos ..... 35
**Tabla 5** Excepciones en el proceso de Modificar Roles y Permisos ..... 36
**Tabla 6** Excepciones en el proceso de Eliminar Roles y Permisos ..... 38
**Tabla 7** Excepciones en el proceso de Consultar Roles y Permisos ..... 39
**Tabla 8** Excepciones en el proceso de Crear Usuarios de Doctores ..... 41
**Tabla 9** Excepciones en el proceso de Crear Usuarios de Personal ..... 42
**Tabla 10** Excepciones en el proceso de Modificar Usuarios ..... 44
**Tabla 11** Excepciones en el proceso de Eliminar Usuarios ..... 45
**Tabla 12** Excepciones en el proceso de Consultar Usuarios ..... 46
**Tabla 13** Excepciones en el proceso de Crear Especialidades ..... 48
**Tabla 14** Excepciones en el proceso de Modificar Especialidades ..... 50
**Tabla 15** Excepciones en el proceso de Eliminar Especialidades ..... 51
**Tabla 16** Excepciones en el proceso de Consultar Especialidades ..... 53
**Tabla 17** Product Backlog del Sprint 2 ..... 68

**Tabla 18** Excepciones en el Proceso de Registrar Pacientes ..... 70
**Tabla 19** Excepciones en el Proceso de Modificar Pacientes ..... 72
**Tabla 20** Excepciones en el Proceso de Eliminar Pacientes ..... 73
**Tabla 21** Excepciones en el Proceso de Consultar Pacientes ..... 75
**Tabla 22** Excepciones en el Proceso de Programar Citas Médicas ..... 77
**Tabla 23** Excepciones en el Proceso de Modificar Citas Médicas ..... 78
**Tabla 24** Excepciones en el Proceso de Cancelar Citas Médicas ..... 80
**Tabla 25** Excepciones en el Proceso de Consultar Citas Médicas ..... 81
**Tabla 26** Product Backlog del Sprint 3 ..... 90

# INDICE DE FIGURAS

**Figura 1** Diagrama de Ciclo de Vida SCRUM ..... 15
**Figura 2** Diagrama General - Casos de Uso ..... 32
**Figura 3** Diagrama de Roles y Permisos ..... 33
**Figura 4** Diagrama a Detalle de Crear Roles y Permisos ..... 34
**Figura 5** Diagrama a Detalle de Modificar Roles y Permisos ..... 35
**Figura 6** Diagrama a Detalle de Eliminar Roles y Permisos ..... 37
**Figura 7** Diagrama a Detalle de Consultar Roles y Permisos ..... 38
**Figura 8** Diagrama de Usuarios ..... 39
**Figura 9** Diagrama a Detalle de Crear Usuarios de Doctores ..... 40
**Figura 10** Diagrama a Detalle de Crear Usuarios de Personal ..... 41
**Figura 11** Diagrama a Detalle de Modificar Usuarios ..... 43
**Figura 12** Diagrama a Detalle de Eliminar Usuarios ..... 44
**Figura 13** Diagrama a Detalle de Consultar Usuarios ..... 45
**Figura 14** Diagrama de Especialidades ..... 47
**Figura 15** Diagrama a Detalle de Crear Especialidades ..... 47
**Figura 16** Diagrama a Detalle de Modificar Especialidades ..... 49
**Figura 17** Diagrama a Detalle de Eliminar Especialidades ..... 50
**Figura 18** Diagrama a Detalle de Consultar Especialidades ..... 52
**Figura 19** Diagrama Entidad - Relación de la BBDD ..... 54
**Figura 20** Pestaña de Login para los Usuarios de la Clínica ..... 56
**Figura 21** Pestaña de Registro de un Nuevo Rol ..... 57

**Figura 22** Pestaña de Listado de Roles con sus Permisos ..... 58
**Figura 23** Pestaña del Listado de Usuarios del Personal ..... 59
**Figura 24** Pestaña para Añadir un Usuario de Personal ..... 59
**Figura 25** Pestaña con el Listado de Usuarios de Doctores ..... 60
**Figura 26** Pestaña para Añadir un Nuevo Usuario de Doctor ..... 60
**Figura 27** Pestaña del Listado de Especialidades ..... 61
**Figura 28** Pestaña para Añadir una Nueva Especialidad ..... 61
**Figura 29** Autenticación JWT ..... 62
**Figura 30** Gestión de Roles y Permisos ..... 63
**Figura 31** Manejo de Tokens con JWT ..... 64
**Figura 32** Registro de Usuarios ..... 64
**Figura 33** Pruebas del Sprint 1 ..... 66
**Figura 34** Diagrama de Gestión de Pacientes ..... 69
**Figura 35** Diagrama a Detalle de Crear Pacientes ..... 69
**Figura 36** Diagrama a Detalle de Modificar Pacientes ..... 71
**Figura 37** Diagrama a Detalle de Eliminar Pacientes ..... 72
**Figura 38** Diagrama a Detalle de Consultar Pacientes ..... 74
**Figura 39** Diagrama de Gestión de Citas Médicas ..... 75
**Figura 40** Diagrama a Detalle de Crear Citas Médicas ..... 76
**Figura 41** Diagrama a Detalle de Modificar Citas Médicas ..... 77
**Figura 42** Diagrama a Detalle de Eliminar Citas Médicas ..... 79
**Figura 43** Diagrama a Detalle de Consultar Citas Médicas ..... 80
**Figura 44** Pestaña de Registro de Pacientes ..... 82
**Figura 45** Pestaña del Listado de Pacientes ..... 83
**Figura 46** Pestaña de Programación de Citas ..... 84
**Figura 47** Pestaña del Listado de Citas de los Doctores ..... 84
**Figura 48** Funcionalidad para el Registro ..... 85
**Figura 49** API RESTful para Gestión de Pacientes y Citas ..... 86
**Figura 50** Validaciones en Laravel ..... 87
**Figura 51** Pruebas del Sprint 2 ..... 88
**Figura 52** Pestaña del Calendario de Citas Médicas ..... 91
**Figura 53** Pestaña de Atención de la Consulta Médica ..... 92
**Figura 54** Pestaña de Perfil del Doctor ..... 93
**Figura 55** Pestaña de Perfil del Paciente ..... 93

**Figura 56** Dashboard de la Clínica para el Administrador....94
**Figura 57** Dashboard de la Clínica para el Doctor....95
**Figura 58** Configuración de Redis....96
**Figura 59** Configuración de la Función FullCalendar....97
**Figura 60** Métodos HTTP para Consultas Médicas....97
**Figura 61** Configuración de la funcionalidad del Dashboard de Admin....98
**Figura 62** Pruebas del Sprint 3....99

# INDICE DE ANEXOS

**Anexo A** Cronograma de Trabajo....102
**Anexo B** Captura de Código subido a GitHub....102
**Anexo C** Documentación Entrevistas....103
**Anexo D** Funcionalidad Login....105
**Anexo E** Funcionalidad del Header....106
**Anexo F** Funcionalidad del SideBar....107
**Anexo G** Funcionalidad de Añadir una Cita....108
**Anexo H** Funcionalidad de Atención Médica....110
**Anexo I** Funcionalidad de Editar una Cita....111
**Anexo J** Funcionalidad de Listado de Citas....113
**Anexo K** Funcionalidad del Calendario de Citas....115
**Anexo L** Funcionalidad de Añadir un Doctor....116
**Anexo M** Funcionalidad del Perfil de Doctores....119
**Anexo N** Funcionalidad de Editar un Doctor....120
**Anexo O** Funcionalidad de Listado de Doctores....123
**Anexo P** Funcionalidad de Añadir Paciente....125
**Anexo Q** Funcionalidad de Editar Paciente....127
**Anexo R** Funcionalidad de Listado de Pacientes....129
**Anexo S** Funcionalidad del Perfil del Paciente....131
**Anexo T** Funcionalidad de Añadir un Rol....132
**Anexo U** Funcionalidad de Editar un Rol....133
**Anexo V** Funcionalidad de Añadir Especialidad....134
**Anexo W** Funcionalidad de Editar Especialidad....134
**Anexo X** Funcionalidad de Listado de Especialidades....135

**Anexo Y** Funcionalidad de Añadir Personal ..... 137
**Anexo Z** Funcionalidad de Editar Personal ..... 138
**Anexo AA** Funcionalidad de Listado de Personal ..... 139
**Anexo BB** Gestión de Autenticación con Angular: auth.service.ts ..... 141
**Anexo CC** Servicio de Gestión de Datos en Angular: data.service.ts ..... 142
**Anexo DD** Punto de Entrada para API en Laravel: api.php ..... 147
**Anexo EE** Controlador de Autenticación en Laravel: AuthController.php ..... 148

# CAPITULO I – INTRODUCIÓN

## 1.1 Tema

Desarrollo de un Sistema de Gestión de Citas y Seguimiento de Pacientes como Aplicación Web para Clínica Dental ARPUL.

## 1.2 Justificación

La gestión de citas, es esencial para clínicas dentales. Un sistema de gestión eficiente en forma de aplicación web puede mejorar significativamente la organización y atención al paciente, elevando la calidad del servicio y la satisfacción del cliente. Este proyecto propone desarrollar un sistema web que gestione citas, adaptándose a las necesidades específicas de médicos y pacientes en la Clínica Dental ARPUL. Actualmente, el manejo manual de citas en nuestra clínica presenta desafíos significativos, incluyendo la pérdida de pacientes y la carencia de un repositorio de información unificado. Estos problemas comprometen directamente la calidad del servicio. Un sistema web personalizado mejoraría estos aspectos críticos. En la Clínica Dental ARPUL, la gestión de citas actualmente varía según el médico, con cada uno organizando sus citas de manera individual y según sus preferencias personales. Esta diversidad de métodos conlleva a una serie de desafíos operativos, como por ejemplo:

- La dificultad para gestionar las citas de forma eficiente, lo que puede generar retrasos, confusiones y errores.
- La falta de información centralizada sobre los pacientes, lo que dificulta el acceso a los datos clínicos y la atención personalizada.
- La dificultad para realizar seguimientos de los pacientes, lo que puede afectar a la calidad del servicio.

El desarrollo de un sistema de gestión de citas para la Clínica Dental ARPUL permitiría abordar estos inconvenientes y mejorar la eficiencia y la calidad del servicio.

## 1.3 Planteamiento del Problema

La gestión eficiente de las clínicas dentales implica abordar múltiples aspectos críticos. En este proyecto, se plantea abordar dos de ellos, centrándose en las necesidades del doctor encargado. El problema principal que se plantea es la falta de uniformidad en el proceso de registro y seguimiento de pacientes, dificultando la coordinación y la eficiencia en la prestación de servicios. Esto facilitará una gestión más coherente y eficiente, mejorando

la experiencia tanto para los profesionales como para los pacientes. Al integrar estas funciones en una sola plataforma, la Clínica Dental ARPUL podrá optimizar sus operaciones, reducir el riesgo de errores y omisiones, y ofrecer un servicio más personalizado y de mayor calidad.

Los problemas secundarios que se derivan de este problema principal son:

- Dificultad para gestionar las citas de forma eficiente.
- Falta de información centralizada sobre los pacientes.
- Dificultad para realizar seguimientos de los pacientes.

En base a estos problemas, se plantean las siguientes preguntas de investigación:

- ¿Cómo se puede mejorar la gestión de citas en Clínica Dental ARPUL?
- ¿Cómo se puede mejorar la información centralizada sobre los pacientes?
- ¿Cómo se puede facilitar el seguimiento de los pacientes?

Las respuestas a estas preguntas de investigación permitirán desarrollar un sistema de gestión de citas para la Clínica Dental ARPUL que sea eficiente, eficaz y satisfaga las necesidades de la clínica.

Este planteamiento del problema es claro, preciso y concreto. Se identifica el problema principal y los problemas secundarios, y se plantean las preguntas de investigación que permitirán abordarlos.

## 1.4 Objetivos

### 1.4.1 General

Desarrollar una Aplicación Web para la Gestión de Citas en la Clínica Dental ARPUL, mejorando la eficiencia y la atención al paciente.

### 1.4.2 Específico

- Revisar los procesos de la Clínica Dental ARPUL para determinar la metodología de desarrollo web más adecuada.
- Detallar requisitos para la gestión de citas.
- Especificar los requerimientos de interfaz de usuario, acceso seguro, y funcionalidades

de recordatorio y seguimiento para la aplicación web.

- Diseñar una interfaz de usuario web amigable.
- Planificar la arquitectura del sistema para una implementación eficiente y segura.
- Implementar, probar y asegurar una aplicación web con sistema de gestión de citas para la Clínica Dental ARPUL, enfocándose en la centralización y seguridad de la información para satisfacer sus necesidades.

## 1.5 Alcance

El alcance del proyecto comprende el diseño y desarrollo de una aplicación que, y se implementará en un entorno real durante el desarrollo tesis, será sometida a pruebas para asegurar su funcionamiento. El desarrollo, con una duración prevista de tres meses, estará a cargo de un único desarrollador bajo la supervisión de un tutor.

- **Análisis de Metodología:** Se investigarán metodologías de desarrollo que mejor se adecuen a las necesidades y contexto de la Clínica Dental ARPUL. Se seleccionará una basada en la eficiencia y pertinencia para el tipo de sistema a desarrollar.
- **Definición de Requerimientos:** Se determinarán los requerimientos funcionales y no funcionales del sistema basados en las necesidades expresadas por la Clínica Dental ARPUL, en especial las relativas a gestión de citas.
- **Diseño de la Aplicación**: El diseño se enfocará en una interfaz amigable e intuitiva que permita gestionar citas y acceder a ellas de manera eficiente. También se tomarán en cuenta aspectos de seguridad de datos.
- **Desarrollo del Sistema:** Se llevará a cabo la programación y desarrollo del sistema, implementar las funciones definidas previamente. Dada la limitación de tiempo de tres meses, se priorizarán las funcionalidades más críticas y se buscará la optimización del proceso de desarrollo para maximizar la eficacia del sistema en ese periodo.
- **Pruebas Respectivas:** Se realizarán pruebas para asegurar el correcto funcionamiento del sistema y la adecuada satisfacción de los requerimientos definidos.

## 1.6 Esquema de Contenidos Básicos

**CAPÍTULO I - INTRODUCCIÓN**

1.1 Titulo

1.2 Justificación

1.3 Planteamiento del Problema

1.4 Objetivos

1.5 Alcance

1.6 Esquema de Contenidos Básicos

**CAPÍTULO II - MARCO TEÓRICO**

2.1 Metodología Ágil de Desarrollo Software

- 2.1.1 Introducción al modelo ágil
- 2.1.2 SCRUM
- 2.1.3 Roles
- 2.1.4 Backlog
- 2.1.5 Sprint

2.2 Herramientas

- 2.2.2 Base de Datos
- 2.2.3 Backend
- 2.2.4 FrontEnd

**CAPÍTULO III – PLANIFICACIÓN**

3.1 Product Backlog

3.2 Estrategia o Planificación

- 3.2.1 Iteraciones

**CAPÍTULO IV - DESARROLLO**

4.1 Desarrollo del Sprint 0

- 4.1.1 Sprint Backlog 0
- 4.1.2 Diseño 0
- 4.1.3 Codificación 0
- 4.1.4 Pruebas 0

4.2 Desarrollo del Sprint 1

- 4.2.1 Sprint Backlog 1
- 4.2.2 Diseño 1

4.2.3 Codificación 1

4.2.4 Pruebas 1

4.3 Desarrollo del Sprint 2

4.3.1 Sprint Backlog 2

4.3.2 Diseño 2

4.3.3 Codificación 2

4.3.4 Pruebas 2

**CAPÍTULO V - CONCLUSIONES Y RECOMENDACIONES**

5.1 Conclusiones

5.2 Recomendaciones

5.3 Anexos

# CAPITULO II – MARCO TEORICO

En este capítulo se desarrollará todo el marco teórico del proyecto del Sistema de Gestión de Citas y Seguimiento de Pacientes para la Clínica Dental ARPUL. Se tomarán temas como metodologías, herramientas y tecnologías seleccionadas para la realización del proyecto.

Al comienzo estarán las metodologías ágiles de desarrollo de software, mostrando un enfoque iterativo y colaborativo, que ofrece ventajas significativas sobre los métodos tradicionales de gestión de proyectos. Dentro de este contexto, Scrum se destaca como un marco de trabajo ágil que promueve la flexibilidad, la transparencia y la adaptación continua a las cambiantes necesidades de la clínica y sus pacientes.

Además, se mencionarán los roles esenciales dentro de la metodología Scrum y cómo, en un proyecto individual, se fusionan en una única persona que maneja la totalidad del desarrollo del software. Esto incluye la gestión del Product Backlog y la ejecución de Sprints, que son críticos para la planificación y ejecución eficaz del proyecto.

El siguiente apartado detalla las herramientas que apoyan el desarrollo del backend y el frontend del sistema. Se examinará la utilidad de PHP, la selección de MySQL como sistema de gestión de bases de datos, y el papel de XAMPP como plataforma de desarrollo. Además, se profundizará en cómo Composer facilita la gestión de dependencias en PHP, mientras que Laravel y Node.js son las herramientas necesarias en la construcción de la lógica de negocio y la interfaz de usuario. Concluyendo con Angular CLI, necesaria para el desarrollo de la interfaz de usuario, buscando que la aplicación web pueda ser eficiente como accesible y agradable para el usuario final.

A través de este capítulo, se presentará una justificación integral y metódica de cada elección tecnológica, ilustrando cómo estas decisiones se integran para formar un enfoque coherente y orientado al éxito para el desarrollo del sistema propuesto.

## 2.1 Metodología Ágil de Desarrollo Software

### 2.1.1 Introducción al Modelo Ágil

El desarrollo de software ha cambiado mucho con el tiempo, impulsado por avances rápidos en tecnología y necesidades de los usuarios cada vez más complejas. Estos cambios han puesto en duda la efectividad de métodos tradicionales de gestión de proyectos como PMP, IPMA, PRINCE2 y ISO 21500. Estos métodos son muy estructurados, pero a menudo

no son lo suficientemente flexibles para adaptarse a cambios rápidos y demandas de entregas rápidas (López Gil, 2018).

En respuesta a estas limitaciones, las metodologías ágiles han surgido como una alternativa, destacando por su adaptabilidad y la participación constante del usuario o cliente final. Estas metodologías promueven la entrega de resultados de manera iterativa y la colaboración continua, lo que permite ajustes y adaptaciones rápidas a medida que aparecen nuevos requerimientos durante el proceso de desarrollo (Molina Montero et al., 2018).

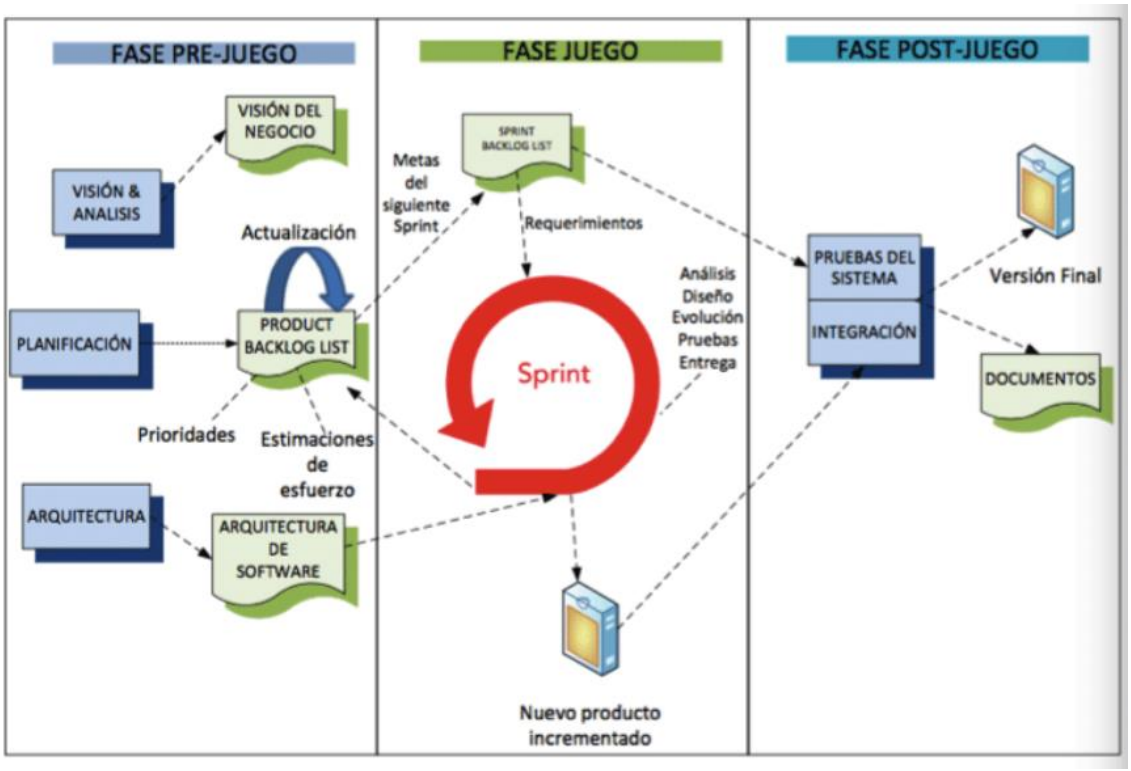
A diferencia de los métodos tradicionales, las metodologías ágiles dividen los proyectos en partes más pequeñas, lo que permite entregas más frecuentes y revisiones constantes. Esto no solo mejora la satisfacción del cliente mostrando progreso de forma continua, sino que también facilita la gestión de cambios. Esto es especialmente importante en proyectos que necesitan adaptarse rápidamente, como el Sistema de Gestión de Citas y Seguimiento de Pacientes para la Clínica Dental ARPUL, donde adaptarse a las necesidades cambiantes de los pacientes y del personal es esencial.

### 2.1.2 Scrum

Scrum es un marco de trabajo que se utiliza para manejar proyectos complejos de forma productiva y creativa, siendo ampliamente adoptado en el desarrollo de software y otros campos que requieren una gestión compleja y colaborativa (Schwaber & Sutherland, 2020). Este marco promueve un ambiente de trabajo colaborativo y eficiente a través de principios como transparencia, inspección y adaptación, y valores como compromiso, coraje, enfoque, apertura y respeto.

**Figura 1**

*Diagrama de Ciclo de Vida SCRUM*




*Nota*. Representación del ciclo de vida Scrum, utilizado como marco metodológico en el proyecto para la Clínica Dental ARPUL. Tomado de Ciclo de Vida Scrum [Figura] ResearchGate, s. f, https://www.researchgate.net/figure/Figura-27-Ciclo-de-Vida-Scrum_fig3_318280962

La **Figura 1** muestra el ciclo de vida de la metodología Scrum, utilizada como marco metodológico en el proyecto para la Clínica Dental ARPUL. En este proyecto, Scrum se estructura en fases de Pre-Juego, Juego y Post-Juego, ofreciendo un enfoque iterativo adecuado para ambientes clínicos dinámicos. La fase de Pre-Juego define la arquitectura y prioridades, los Sprints en la fase de Juego desarrollan y prueban incrementos del producto, y la fase de Post-Juego asegura la integración y efectividad del sistema. Este enfoque permite una retroalimentación continua y se adapta a las necesidades cambiantes, lo cual es esencial para la clínica.

A pesar de los desafíos que presenta Scrum, como la resistencia al cambio y la necesidad de compromiso intenso en un entorno donde la colaboración constante puede ser difícil, las ventajas de una mayor satisfacción del cliente y respuesta ágil a las necesidades

cambiantes justifican su implementación (Rodríguez & Dorado, n.d.; Schwaber & Sutherland, 2020).

### 2.1.3 Roles

Se entiende que para la metodología SCRUM, todo el proyecto se divide en diferentes Sprints y en diferentes grupos de trabajo con sus roles respectivos por lo que facilita el avance el proyecto.

En la ***Tabla 1*** se presenta la asignación de roles dentro del marco Scrum para el proyecto de la Clínica Dental ARPUL. A continuación, se desglosa la responsabilidad de cada rol, enfatizando la importancia de cada uno en el desarrollo del proyecto.

**Tabla 1**

*Asignación de Roles en el Marco Scrum para el proyecto de la Clínica Dental ARPUL*

| Rol | Persona Asignada |
|---|---|
| Product Owner | Estudiante (Erick Casa) |
| Scrum Master | Tutor (Fabián de la Cruz) |
| Developer | Estudiante (Erick Casa) |

*Nota*. Tabla para resumir los roles en dicho proyecto

- **Product Owner**: El estudiante asume este rol, con la responsabilidad de definir y priorizar el backlog del producto para asegurar que refleje los objetivos y necesidades del sistema de manera precisa (Schwaber & Sutherland, 2020).
- **Scrum Master**: Esta función es desempeñada por el tutor, quien se encarga de mantener los ritmos de trabajo, resolver obstáculos como limitaciones de tiempo y recursos, y garantizar la disciplina y auto-organización dentro del equipo (Rodríguez & Dorado, n.d.).
- **Developer**: También ocupado por el estudiante, este rol involucra ejecutar las tareas de desarrollo, incluida la codificación y el diseño, comprometiéndose a satisfacer las necesidades del cliente y garantizar una entrega continua de valor.

La integración de estos roles en una sola persona se entiende que va a presentar desafíos únicos, como la necesidad de una auto-gestión efectiva y un enfoque equilibrado entre la planificación y la ejecución. Sin embargo, la metodología Scrum promueve una

estructura de trabajo que facilita la adaptabilidad y la mejora continua, crucial para proyectos individuales donde la agilidad y la capacidad de respuesta son claves para el cumplimiento del trabajo (Schwaber & Sutherland, 2020).

### 2.1.4 Backlog

En el marco de Scrum, el Backlog del Producto es crucial para gestionar proyectos eficazmente, especialmente en entornos de desarrollo individual. Según Schwaber y Sutherland (2020), este Backlog es una lista prioritaria de todo lo necesario para mejorar el producto, siendo la única fuente de trabajo para el equipo Scrum, lo que resalta su importancia en la planificación y ejecución de proyectos.

Para proyectos como el Sistema de Gestión de Citas y Seguimiento de Pacientes, el Backlog del Producto es central. Actúa como el núcleo de planificación donde se registran todas las funcionalidades, requisitos y cambios necesarios. En un entorno unipersonal, donde una sola persona asume múltiples roles, el Backlog facilita la autogestión y organización del trabajo (Schwaber & Sutherland, 2020).

El Backlog permite priorizar tareas basadas en su importancia y urgencia, enfocándose en los aspectos críticos del proyecto. Su flexibilidad es vital en proyectos individuales, permitiendo adaptaciones rápidas a nuevos requerimientos que surgen durante el desarrollo (Rodríguez & Dorado, 2015).

En resumen, el Backlog del Producto es fundamental en Scrum por su rol en la organización, planificación y ejecución del desarrollo del sistema, demostrando la eficacia de Scrum incluso cuando se cuenta con un solo desarrollador.

### 2.1.5 Sprint

Dentro del marco de Scrum, los Sprints son esenciales para estructurar el flujo de trabajo en ciclos iterativos y manejables, promoviendo la entrega continua y la adaptabilidad. Este concepto, aplicado a un proyecto unipersonal que es el desarrollo del Sistema de Gestión de Citas y Seguimiento de Pacientes, facilita una organización eficaz del trabajo y proponiendo un progreso constante hacia los objetivos del proyecto.

Los Sprints son periodos de tiempo fijo durante los cuales se crea un incremento de producto terminado, utilizable y potencialmente entregable. La duración de un Sprint es típicamente de uno a cuatro semanas, manteniéndose constante a lo largo del proyecto para asegurar regularidad y previsibilidad en el trabajo (Takeuchi & Nonaka, 1986; Schwaber & Sutherland, 2020).

En el contexto de un proyecto gestionado por una sola persona, la implementación de Sprints ayuda a estructurar el trabajo de forma que se maximice la eficiencia, manteniendo un equilibrio entre la planificación, el desarrollo, la revisión y la adaptación.

## 2.2 Herramientas

Antes de examinar cada herramienta específica, se quiere dar una visión general que subraya la importancia de las tecnologías seleccionadas para el desarrollo del Sistema de Gestión de Citas y Seguimiento de Pacientes. Cada herramienta desempeña un papel esencial en diversas funciones del proyecto, desde la gestión de datos hasta la interacción del usuario, asegurando que el sistema sea robusto, eficiente y adaptable.

En la **Tabla 2** se presenta un resumen de las principales herramientas utilizadas en el proyecto, proporcionando un rápido entendimiento de su papel y contribución al desarrollo del sistema.

**Tabla 2**

*Herramientas de Desarrollo Utilizadas en el Proyecto*

| Herramienta | Descripción | Uso en el Proyecto |
|---|---|---|
| PHP | Lenguaje de programación del lado del servidor. | Utilizado para el backend del sistema, gestionando la lógica y las interacciones con la base de datos. |
| MySQL | Sistema de gestión de bases de datos. | Almacena y gestiona los datos de pacientes y citas. |
| XAMPP | Paquete de software que instala Apache, MySQL, PHP. | Proporciona un entorno de desarrollo local para pruebas y desarrollo. |
| Laravel | Framework de PHP basado en el patrón MVC. | Facilita la estructura del desarrollo y mantiene el código organizado. |
| Node.js | Entorno de ejecución para JavaScript del lado del servidor. | Maneja solicitudes asíncronas y mejora la escalabilidad del sistema. |
| Angular CLI | Herramienta para automatizar el desarrollo con Angular. | Utilizado para desarrollar la interfaz de usuario, haciendo que sea interactiva y responsiva. |

*Nota*. Resumen en tabla de las herramientas con el uso en el proyecto.

Tras esta visión general, ahora detallaremos cada herramienta individualmente para entender mejor cómo contribuyen específicamente al funcionamiento y éxito del sistema. Este análisis proporcionará una comprensión más profunda de la arquitectura del sistema y de cómo la elección de cada tecnología se alinea con los objetivos del proyecto.

### 2.2.1 Bases de Datos

- MySQL

MySQL es ampliamente valorado por su eficiencia y rapidez, cualidades esenciales para aplicaciones que gestionan grandes volúmenes de datos y requieren operaciones de lectura y escritura rápidas. Estas características lo hacen ideal para aplicaciones web dinámicas y para sistemas complejos como el Sistema de Gestión de Citas y Seguimiento de Pacientes. Al emplear el modelo relacional, MySQL facilita la organización y manipulación de los datos, garantizando una integración efectiva y segura en entornos que demandan un manejo intensivo de información (ProximaHost, 2023; Arsys, 2023).

Además, la integración de MySQL con PHP a través de APIs simplifica el desarrollo de aplicaciones web interactivas, permitiendo una interacción fluida entre la lógica de la aplicación y la base de datos. Esto es especialmente crucial para sistemas como el de la clínica dental, donde la adaptabilidad y eficiencia en la gestión de datos son fundamentales. La popularidad de MySQL también se debe a su modelo de código abierto, lo que proporciona flexibilidad y un amplio soporte comunitario, facilitando su uso y adaptación por una gran comunidad de usuarios y desarrolladores (ProximaHost, 2023).

- XAMPP

Para el desarrollo del Sistema de Gestión de Citas y Seguimiento de Pacientes en la Clínica Dental ARPUL, se eligió XAMPP como plataforma de desarrollo por su versatilidad y facilidad de uso. XAMPP, que integra Apache, MySQL y PHP, ofrece un entorno de desarrollo unificado que es fundamental para crear aplicaciones web estables y seguras (EcuRed, n.d.). XAMPP facilita una configuración rápida y eficiente del entorno de desarrollo, incorporando todos los componentes esenciales en un solo paquete, lo que acelera el proceso de desarrollo.

La funcionalidad de su servidor MySQL es crucial para gestionar datos de manera eficiente y segura. Además, PHP en XAMPP asegura operaciones del lado del servidor rápidas y efectivas, mejorando la respuesta y funcionalidad del sistema (EcuRed, n.d.). La elección de XAMPP subraya la necesidad de una plataforma integrada que respalde el desarrollo ágil de aplicaciones web, proporcionando una solución práctica y eficiente para las necesidades técnicas del proyecto.

### 2.2.2 Backend

- PHP

PHP fue seleccionado para el desarrollo del Sistema de Gestión de Citas y Seguimiento de Pacientes debido a su flexibilidad y eficiencia en el desarrollo de aplicaciones web. Este lenguaje facilita la integración con diversas tecnologías y bases de datos, siendo ideal para metodologías ágiles como Scrum, lo cual es crucial para proyectos dinámicos y adaptables.

Según Bautista-Villegas (2022), PHP es ampliamente valorado por su sintaxis accesible y compatibilidad con prácticas avanzadas de ingeniería de software, apoyando la eficiencia, confiabilidad y escalabilidad necesarias en este sistema. Su integración con el framework Laravel, utilizando el patrón Modelo-Vista-Controlador (MVC), optimiza la separación entre la lógica de negocio y la interfaz de usuario, facilitando el desarrollo y mantenimiento del sistema. Esta capacidad hace de PHP una opción robusta para adaptarse a los requerimientos cambiantes y proporcionar una interfaz intuitiva para los usuarios (Bautista-Villegas, 2022).

- Node.js

Node.js es una plataforma de ejecución orientada a eventos asíncronos que sobresale en el desarrollo de aplicaciones web intensivas en datos, siendo más eficiente y ligero que tecnologías como PHP, especialmente para sitios web en tiempo real. Según Lei et al. (2014), Node.js maneja más solicitudes en comparación con PHP y Python para aplicaciones de gran escala y alta concurrencia. Su capacidad para gestionar múltiples conexiones simultáneas lo hace ideal para aplicaciones que procesan grandes volúmenes de datos en tiempo real. Esta eficiencia y escalabilidad hacen de Node.js una opción clave para el desarrollo del Sistema de Gestión de Citas y Seguimiento de Pacientes, maximizando la eficiencia y capacidad de respuesta del sistema.

- Laravel

Laravel, un destacado framework de PHP, facilita el desarrollo web eficiente mediante su adhesión al patrón Modelo-Vista-Controlador (MVC), que separa la lógica de negocio de la interfaz de usuario y optimiza la adaptabilidad y respuesta de las aplicaciones en diversos dispositivos. Su interoperabilidad con la arquitectura API REST, crucial para aplicaciones modernas como móviles o SPAs, permite una comunicación fluida entre servidor y clientes,

esencial en la gestión de datos sensibles como los de pacientes y citas, según Bautista-Villegas (2022).

La arquitectura flexible de Laravel, junto con su micro-framework Lumen, soporta la creación de servicios escalables que se integran bien con la arquitectura REST, facilitando la expansión o modificación de aplicaciones sin interrupciones. En el desarrollo de la aplicación web para la Clínica Dental ARPUL, la elección de Laravel se ve reforzada por su escalabilidad, una comunidad activa y acceso a herramientas de desarrollo actualizadas, garantizando un sistema de gestión de citas robusto, seguro y adaptable a las necesidades cambiantes de la clínica (Bautista-Villegas, 2022; Avilés Matute et al., 2020).

- Composer

La elección de Composer como herramienta de gestión de dependencias es crucial en el desarrollo del proyecto, ya que facilita la declaración y gestión eficiente de las bibliotecas necesarias, manteniendo el sistema actualizado y coherente. Este enfoque optimiza la integración y actualización de Laravel y otras dependencias, mejorando la compatibilidad y eficiencia durante el ciclo de desarrollo (Sunardi & Suharjito, 2019).

Composer permite la instalación y actualización simplificadas de Laravel y sus dependencias, ahorrando tiempo en la configuración y simplificando el mantenimiento a largo plazo. Al emplear Composer, los desarrolladores pueden utilizar las versiones más recientes y seguras de los paquetes, reforzando así la seguridad y estabilidad del sistema de gestión de citas (Bean, 2015).

Este método de gestión de dependencias subraya el compromiso con la calidad, seguridad y eficiencia, siendo esencial para el desarrollo de aplicaciones web con Laravel. Además, Composer promueve las mejores prácticas dentro de la comunidad PHP, facilitando el desarrollo de aplicaciones mantenibles y resultando una opción ideal para el proyecto.

### 2.2.3 Frontend

- Angular CLI

Angular CLI es una interfaz de línea de comandos que facilita la creación, desarrollo y prueba de aplicaciones Angular mediante un conjunto de herramientas que mejoran la eficiencia del proceso de desarrollo. Es crucial para gestionar el entorno del proyecto y

automatizar tareas como la generación de componentes, el manejo de pruebas y la optimización del despliegue.

Según Jukka Korva (2016), Angular CLI es fundamental para mantener la consistencia y seguir las mejores prácticas en el desarrollo de aplicaciones web dinámicas, asegurando así la calidad y mantenibilidad del código. Angular CLI automatiza el flujo de trabajo de desarrollo, lo que permite a los desarrolladores concentrarse más en la lógica de la aplicación y menos en la configuración del proyecto, según la documentación oficial de Angular (Angular Team, n.d.).

Esta automatización es especialmente valiosa en metodologías ágiles, facilitando iteraciones rápidas y una integración continua, esenciales para la entrega iterativa de valor en Scrum. Angular CLI, por lo tanto, juega un papel clave en la construcción de la interfaz de usuario del sistema, optimizando tanto el desarrollo como el despliegue en el entorno del proyecto.

# CAPITULO III – PLANIFICACIÓN

Este capítulo detalla la estrategia de planificación del proyecto para el Sistema de Gestión de Citas y Seguimiento de Pacientes utilizando una metodología ágil. Se inicia con la configuración del Product Backlog, que es una lista que recopila todas las necesidades del sistema. Este listado se basa en entrevistas exhaustivas con el dueño de la clínica, quien aportó su conocimiento y experiencia para definir claramente lo que se necesita en el sistema. El Product Backlog se actualiza continuamente para adaptarse a los cambios y retroalimentación durante el proyecto.

La planificación descrita en este capítulo facilita un desarrollo iterativo y enfocado en valor, organizado en cuatro Sprints específicos. Cada Sprint tiene metas definidas y se centra en desarrollar incrementos funcionales del sistema, desde la configuración inicial y la gestión de usuarios hasta la implementación de mejoras y correcciones basadas en las sugerencias y necesidades del cliente.

Cada Sprint se inicia con una sesión de planificación que determina los elementos del Product Backlog a desarrollar, asegurando que el proyecto se mantenga alineado con las necesidades operativas y estratégicas de la clínica. La estructura de cada Sprint se enfoca en maximizar la eficacia del proceso de desarrollo, permitiendo entregas oportunas y relevantes, a la vez que se mantiene la calidad y se optimiza el rendimiento del sistema.

Este capítulo ilustra cómo la estrategia de planificación y las iteraciones propuestas son fundamentales para el éxito del proyecto, garantizando que el sistema final sea robusto, eficiente y altamente funcional.

## 3.1 Product Backlog

El Product Backlog para el proyecto se desarrolla con el objetivo de abarcar todos los aspectos necesarios para el funcionamiento del sistema, con las base de datos y buscando desarrollar una interfaz amigable. La estructuración y manejo del Product Backlog es vital, busca garantizar que el desarrollo se alinee con las necesidades del proyecto y los intereses de los usuarios. La implementación de Scrum y el uso de un Product Backlog bien mantenido contribuyen significativamente a la agilidad y flexibilidad requeridas para adaptar el sistema a medida que evoluciona el proyecto (Mariño & Alfonzo, 2014).

El Product Backlog para este proyecto fue elaborado tras una serie de entrevistas detalladas con el dueño de la clínica, quien aportó su visión y experiencia para definir con precisión los requerimientos funcionales y no funcionales del sistema. Estas entrevistas fueron fundamentales para asegurar que el desarrollo del sistema se alinee con las necesidades reales de la clínica y sus expectativas operativas.

**Requerimientos Funcionales y No Funcionales:**

- **Requerimientos Funcionales**: Incluyen todas las acciones específicas que el sistema debe ser capaz de realizar.
- **Requerimientos No Funcionales**: Incluyen aspectos como seguridad, rendimiento, y compatibilidad, que describen cómo el sistema debe operar.

A continuación, se detallan los elementos del Product Backlog, organizados por categorías como solicitado por el cliente, reflejando lo que el sistema debe ser capaz de hacer:

**Requerimientos Funcionales**

Los requerimientos funcionales definen las capacidades esenciales que el sistema debe tener para facilitar las operaciones diarias de la clínica:

1. **Autenticación y Seguridad**
    - Debe existir un mecanismo que permita a los usuarios validar su identidad con el login del sistema, para acceder a las funcionalidades del sistema que correspondan a su rol.
2. **Gestión de Roles y Permisos**
    - El sistema debe ofrecer la capacidad de asignar roles específicos a los usuarios, como doctores, enfermeros, recepcionistas y administradores, con funcionalidades accesibles según el rol asignado.
3. **Perfiles de Usuarios**
    - Los usuarios deben poder modificar su información personal y ajustes de cuenta de manera sencilla dentro del sistema.
4. **Sistema de Reserva de Citas**

- Se requiere un sistema que permita a los pacientes y al personal reservar citas, visualizando claramente las opciones de tiempo y disponibilidad de los doctores.

5. **Calendario de Citas Médicas**
    - Un calendario visual que integre todas las citas médicas programadas, proporcionando una visión global de las actividades diarias en la clínica.
6. **Registro y Seguimiento de Pacientes**
    - El sistema debe facilitar el registro de nuevos pacientes y el seguimiento continuo de su historial médico y tratamientos.
7. **Reportes y Análisis de Datos**
    - Herramientas que permitan visualizar reportes sobre la operación de la clínica.

**Requerimientos No Funcionales**

Los requerimientos no funcionales describen las características que el sistema debe cumplir para soportar adecuadamente las operaciones de la clínica:

- **Consistencia**
    - El sistema debe ofrecer una experiencia de usuario consistente en términos de diseño, comportamiento de la interfaz y mensajes de error a través de todas sus funciones.
- **Compatibilidad**
    - El sistema debe ser compatible con los principales navegadores web utilizados en la clínica (por ejemplo, Chrome, Firefox, Safari).
- **Intuitividad**
    - El sistema pueda ser intuitivo para personas que tienen cierta habilidad tecnológica, permitiendo a los usuarios realizar tareas comunes sin necesidad de formación o soporte técnico extensivo.
- **Usabilidad**
    - La interfaz de usuario del sistema debe ser clara y fácil de navegar para personas con variados niveles de habilidad tecnológica.

Cada elemento del Product Backlog está diseñado para ser completado en iteraciones sucesivas, agregando valor incremental al sistema y acercándonos a la visión completa del producto. Esta lista se revisará y adaptará regularmente para reflejar el feedback del usuario y los cambios en los requisitos clínicos y tecnológicos.

## 3.2 Estrategia o Planificación

La estrategia de planificación para el proyecto se fundamenta en principios de desarrollo ágil. Este enfoque asegura que la ejecución del proyecto sea adaptable, iterativa y centrada en el valor continuo del producto. Se adoptará el marco de trabajo Scrum, el cual apoya la planificación flexible y responde eficazmente a la naturaleza impredecible del desarrollo de software (Schwaber & Sutherland, 2020).

Dentro de Scrum, la planificación se divide en ciclos cortos y repetitivos denominados Sprints. Estos Sprints facilitan un enfoque sistemático para el desarrollo del proyecto, asegurando que cada función sea diseñada, desarrollada y probada dentro de un marco temporal establecido. Al final de cada Sprint, se revisa el trabajo realizado y se ajusta el plan de acuerdo con la retroalimentación y los resultados obtenidos.

### 3.2.1 Iteraciones

Las iteraciones son fundamentales en la metodología ágil. A través de los Sprints, el proyecto evoluciona en etapas, con cada Sprint enfocado en la entrega de incrementos de producto. La estructura de los Sprints para el proyecto será la siguiente:

**Sprint 1: Autenticación de Usuarios, Roles y Gestión del Personal**

- **Duración:** 3 semanas
- **Objetivos:**
    - **Autenticación y Login:** Implementar un sistema seguro de autenticación y login para todos los usuarios.
    - **Roles y Permisos:** Establecer y configurar los roles y permisos para diferentes usuarios (doctores, asistentes, recepcionistas, administradores).

- **Módulo de Usuario:** Implementar funcionalidades para manejar la información profesional de los doctores y pacientes.
- **Módulo de Especialidades:** Crear un sistema para gestionar especialidades médicas dentro de la clínica.

**Sprint 2: Gestión de Doctores, Pacientes y Citas Médicas**

- **Duración:** 2.5 semanas
- **Objetivos:**
  - **Módulo de Pacientes:** Desarrollar un sistema para registrar y seguir la información de los pacientes.
  - **Citas Médicas:** Crear un sistema integral para la programación y gestión de citas médicas.

**Sprint 3: Calendario, Consultas Médicas y Dashboards**

- **Duración:** 2.5 semanas
- **Objetivos:**
  - **Calendario de Citas Médicas:** Implementar un calendario para visualizar todas las citas médicas programadas.
  - **Atención de la Consulta Médica:** Desarrollar funcionalidades para gestionar la atención durante las consultas médicas, incluyendo el registro de diagnósticos y tratamientos.
  - **Perfiles de Doctor y Pacientes:** Implementar interfaces para visualizar los perfiles detallados de doctores y pacientes.
  - **Dashboards:** Crear dashboards para visualizar KPIs y otros datos relevantes para la administración de la clínica.

Cada Sprint comienza con una planificación detallada donde se seleccionan los elementos del Product Backlog que serán desarrollados, se definen los objetivos del Sprint y se elabora el Sprint Backlog. Esta estructura promueve la transparencia y la revisión constante, elementos que son vitales para mantener el proyecto alineado con los objetivos estratégicos de la clínica dental (Schwaber & Sutherland, 2020).

# CAPITULO IV – DESARROLLO

Este capítulo aborda el desarrollo práctico del Sistema de Gestión de Citas y Seguimiento de Pacientes para la Clínica Dental ARPUL, estructurado en tres sprints consecutivos, cada uno con objetivos específicos y un enfoque iterativo que subraya la metodología ágil de Scrum.

En el **Sprint 1**, con una duración de tres semanas, se centró en la autenticación de usuarios, roles y gestión del personal. Se implementaron sistemas seguros de autenticación y login, configuración de roles y permisos para diferentes usuarios, y módulos para gestionar información de doctores y especialidades médicas.

El **Sprint 2**, que duró 2.5 semanas, se dedicó a la gestión de doctores, pacientes y citas médicas. Durante este sprint, se desarrollaron sistemas para registrar y seguir la información de los pacientes, así como para la programación y gestión de citas médicas.

Finalmente, el **Sprint 3**, también de 2.5 semanas, se enfocó en la implementación de un calendario de citas médicas, atención durante las consultas médicas, y creación de dashboards para visualizar KPIs y otros datos relevantes. También se desarrollaron interfaces para los perfiles detallados de doctores y pacientes.

Cada sprint abordó un conjunto clave de funcionalidades, asegurando que el sistema evolucionará constantemente para cumplir con las necesidades emergentes de la clínica, al tiempo que se mantenía la consistencia y calidad del desarrollo a través de pruebas y revisiones continuas. Este capítulo no solo detalla el progreso y los resultados de cada sprint, sino que también refleja cómo la adaptabilidad y el enfoque colaborativo de Scrum son esenciales para el éxito del proyecto.

## 4.1 Desarrollo Del Sprint 1

### 4.1.1 Sprint Backlog 1

El Sprint Backlog 0 se centra en la implementación de funcionalidades esenciales para la autenticación de usuarios, la gestión de roles y permisos, la gestión de usuarios y la gestión de especialidades. Estas tareas son cruciales para establecer las bases del sistema y asegurar que las operaciones administrativas de la clínica dental se lleven a cabo de manera eficiente y segura.

Para comenzar, se ha implementado un sistema seguro de autenticación y login para todos los usuarios del sistema. Este módulo de autenticación permite a los usuarios acceder al sistema utilizando sus credenciales únicas. La creación de usuarios (doctores, asistentes, recepcionistas y administradores) y la asignación de sus contraseñas son manejadas exclusivamente por el administrador del sistema. Esto garantiza que solo personal autorizado pueda acceder a las funcionalidades críticas, asegurando la integridad y seguridad del sistema.

El enfoque en la gestión de usuarios permite una administración centralizada de todos los perfiles dentro del sistema. Los casos de uso se dividen en la gestión de doctores y pacientes, pero en esencia, se refieren a la administración de todos los usuarios que interactúan con el sistema. Este nivel de control es esencial para cumplir con los requisitos operativos y de seguridad especificados por la Clínica Dental ARPUL durante las entrevistas iniciales.

**Objetivos del Sprint 1**

**Objetivos Específicos**:

- **Autenticación y Login**: Implementar un sistema seguro de autenticación y login para todos los usuarios.
- **Roles y Permisos**: Establecer y configurar los roles y permisos para diferentes usuarios (doctores, asistentes, recepcionistas, administradores).
- **Módulo de Usuarios**: Implementar funcionalidades para manejar la información profesional de los doctores y pacientes.
- **Módulo de Especialidades**: Crear un sistema para gestionar especialidades médicas dentro de la clínica.

Estas funcionalidades son esenciales para garantizar que cada usuario tenga acceso solo a las áreas del sistema que son relevantes para su rol, mejorando la seguridad y eficiencia operativa. Las entrevistas con el personal de la clínica revelaron la necesidad de un sistema robusto que permita una administración eficiente y segura de las operaciones diarias.

En la **Tabla 3** se presenta el Product Backlog del Sprint 1, destacando las historias de usuario, criterios de aceptación y prioridades. Estas funcionalidades son esenciales para garantizar que cada usuario tenga acceso solo a las áreas del sistema que son relevantes para su rol, mejorando la seguridad y eficiencia operativa. Las entrevistas con el personal de la clínica revelaron la necesidad de un sistema robusto que permita una administración eficiente y segura de las operaciones diarias.

**Tabla 3**

*Product Backlog del Sprint 1*

| ID | Historia de Usuario | Criterios de Aceptación | Prioridad |
|---|---|---|---|
| 1 | Como administrador, quiero poder crear roles para diferenciar los accesos de los usuarios, para asegurar que cada usuario solo acceda a las funcionalidades que le corresponden. | Se debe poder crear roles desde una interfaz administrativa y asignar permisos específicos a cada rol. | Alta |
| 2 | Como usuario, quiero poder registrarme e iniciar sesión utilizando mi correo electrónico y contraseña para acceder al sistema. | El sistema debe validar las credenciales y permitir el acceso solo si son correctas. Debe existir la opción de recuperar contraseña. | Alta |
| 3 | Como administrador, necesito gestionar las especialidades médicas para poder asignar doctores a sus respectivas áreas de especialización. | Debe existir una interfaz donde se puedan añadir, modificar y eliminar especialidades médicas. | Media |
| 4 | Como administrador, quiero gestionar la información de los usuarios para mantener actualizados los datos de doctores y personal. | Debe existir una interfaz para crear, modificar y eliminar usuarios, incluyendo detalles como especialidad y horarios. | Alta |
| 5 | Como administrador, quiero configurar los permisos de acceso de los usuarios para | El sistema debe permitir definir y modificar los permisos asociados a | Alta |

| | | |
|---|---|---|
| | controlar el acceso a diferentes partes del sistema. | cada rol, con cambios reflejados en tiempo real. |

*Nota*. Tabla sobre el Product Backlog del primer sprint con las prioridades

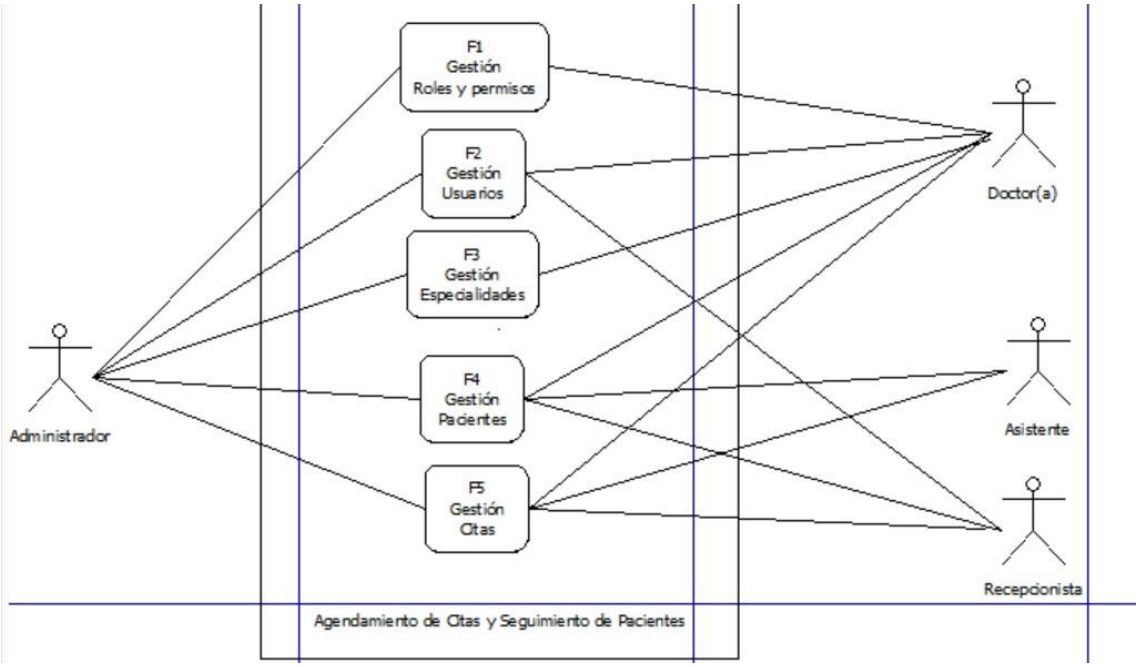
Tras revisar el Product Backlog, procederemos a explorar los casos de uso que describen la implementación práctica de estas funcionalidades, asegurando que el sistema cumpla con las necesidades operativas y de seguridad de la clínica.

**Diagrama General de Casos de Uso**

A continuación se presenta el diagrama general de casos de uso que muestra la relación entre el administrador y las diferentes funcionalidades del sistema:

**Figura 2**

*Diagrama General - Casos de Uso*




*Nota*. Diagrama de los Casos de Usos con las interacciones que tiene cada actor

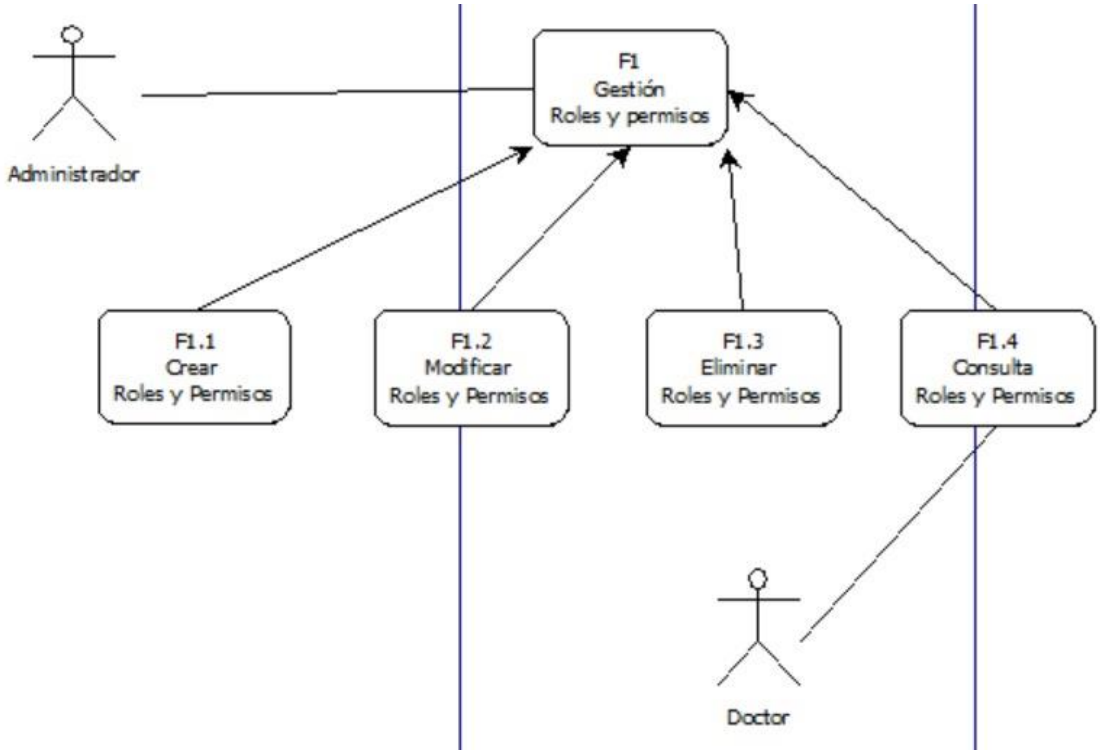
Con la **Figura 2** proporciona una visión general de las interacciones entre los actores y las funcionalidades del sistema, facilitando la comprensión del flujo de operaciones y asegurando que todas las necesidades del negocio estén cubiertas de manera eficiente.

**Detalle de Casos de Uso**

- **Caso de Uso: Gestión de Roles y Permisos**

***Figura 3***

*Diagrama de Roles y Permisos*




*Nota*. Diagrama detallado sobre la primera función de Gestión de Roles y Permisos

**Descripción**: Establecer y configurar roles y permisos para diferentes tipos de usuarios.

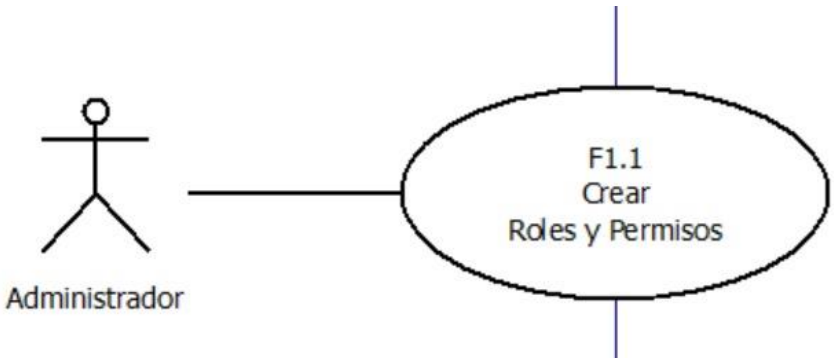
**Actores**: Administrador y Doctor

El diagrama en la **Figura 3** ilustra el caso de uso para la gestión de roles y permisos. Este diagrama detalla cómo se establecen y configuran los roles y permisos para diferentes tipos de usuarios, con los actores principales siendo el Administrador y el Doctor.

**F1.1: Crear Roles y Permisos**

**Figura 4**

*Diagrama a Detalle de Crear Roles y Permisos*




*Nota*. Diagrama a Detalle de Crear un Rol y Permisos que solo puede el Administrador

**Descripción**: El administrador puede crear nuevos roles y asignarles permisos específicos.

**Actor**: Administrador

El diagrama en la **Figura 4** muestra el proceso detallado para crear roles y permisos. El administrador puede crear nuevos roles y asignarles permisos específicos. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Roles" del menú principal.
2. El sistema presenta la ventana de roles.
3. El actor ingresa el nombre del rol.
4. El actor selecciona los permisos asociados al rol.
5. El actor presiona "Guardar".
6. El sistema verifica si el nombre del rol ya existe (E1).
7. El sistema almacena los datos del nuevo rol (E2).

En la **Tabla 4** se muestran las excepciones en el proceso de crear roles y permisos, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 4**

*Excepciones en el proceso de Crear Roles y Permisos*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El nombre del rol ya existe | Ir al caso de uso F1.2 o F1.3. |
| E2 | No se almacenan los datos | Consultar con el administrador. |

*Nota*. Tabla sobre las excepciones de crear un rol y permiso en el flujo principal con su causa y solución.

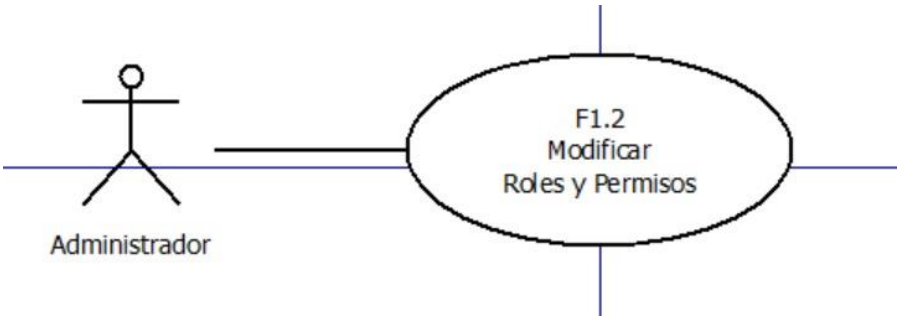
**Flujo Alterno**:

6. Existe el nombre del rol → Ver Caso de Uso F1.2 o F1.3 para modificar o eliminar.

**F1.2: Modificar Roles y Permisos**

**Figura 5**

*Diagrama a Detalle de Modificar Roles y Permisos*




*Nota*. Diagrama a Detalle de Modificar un Rol y Permiso que solo puede el Administrador

**Descripción**: El administrador puede modificar los permisos asignados a un rol existente.

**Actor**: Administrador

El diagrama en la **Figura 5** ilustra el proceso detallado para modificar roles y permisos. El administrador puede modificar los permisos asignados a un rol existente. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Roles" del menú principal.
2. El sistema presenta la ventana de roles.
3. El actor busca el rol que desea modificar (E1).
4. El sistema carga los datos del rol.
5. El actor modifica los permisos del rol.
6. El actor presiona "Guardar".
7. El sistema almacena los datos modificados (E2).

En la **Tabla 5** se muestran las excepciones en el proceso de modificar roles y permisos, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 5**

*Excepciones en el proceso de Modificar Roles y Permisos*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El rol no existe | Ir al caso de uso F1.1 |
| E2 | No se almacenan los datos | Consultar con el administrador. |

*Nota*. Tabla de excepciones de Modificar un Rol y Permiso con su causa y solución

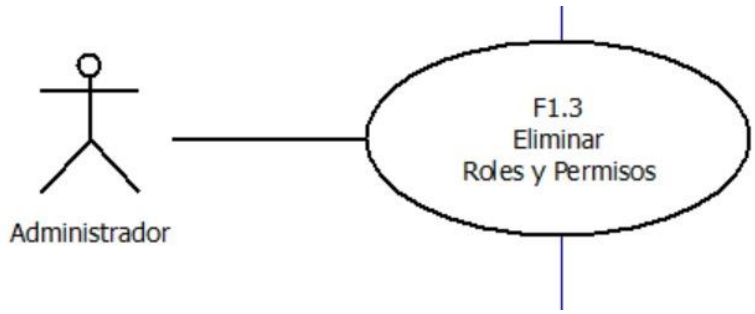
**Flujo Alterno**:

3. No existe el rol → Ver Caso de Uso F1.1 para crear el rol.

**F1.3: Eliminar Roles y Permisos**

**Figura 6**

*Diagrama a Detalle de Eliminar Roles y Permisos*




*Nota*. Diagrama a Detalle de Eliminar un Rol y Permiso que solo puede el Adminstrador

**Descripción**: El administrador puede eliminar roles que ya no sean necesarios.

**Actor**: Administrador

El diagrama en la **Figura 6** muestra el proceso detallado para eliminar roles y permisos. El administrador puede eliminar roles que ya no sean necesarios. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Roles" del menú principal.
2. El sistema presenta la ventana de roles.
3. El actor busca el rol que desea eliminar (E1).
4. El actor presiona la opción de "Eliminar".
5. El sistema solicita confirmación.
6. El actor confirma la eliminación.
7. El sistema elimina el rol (E2).

En la **Tabla 6** se muestran las excepciones en el proceso de eliminar roles y permisos, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 6**

*Excepciones en el proceso de Eliminar Roles y Permisos*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El rol no existe | Ir al caso de uso F1.1 |
| E2 | No se eliminan los datos | Consultar con el administrador. |

**Nota**. Tabla de excepciones al Eliminar un Rol o Permiso con sus causas y soluciones

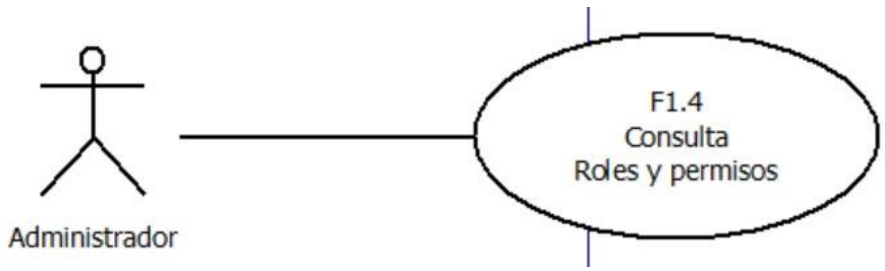
**Flujo Alterno**:

3. No existe el rol → Ver Caso de Uso F1.1 para crear el rol.

**F1.4: Consultar Roles y Permisos**

**Figura 7**

*Diagrama a Detalle de Consultar Roles y Permisos*




*Nota*. Diagrama a Detalle de Consultar un Rol y Permiso que solo puede el Administrador

**Descripción**: El administrador puede consultar los roles y permisos existentes.

**Actor**: Administrador

El diagrama en la **Figura 7** muestra el proceso detallado para consultar roles y permisos. El administrador puede consultar los roles y permisos existentes. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Roles" del menú principal.

2. El sistema presenta la ventana de roles.
3. El actor busca el rol que desea consultar.
4. El sistema muestra los permisos asociados al rol (E1).

En la **Tabla 7** se muestran las excepciones en el proceso de consultar roles y permisos, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 7**

*Excepciones en el proceso de Consultar Roles y Permisos*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El rol no existe | Ir al caso de uso F1.1 |

*Nota***.** Tabla de excepciones de Consultar un Rol y Permiso con su causa y solución.

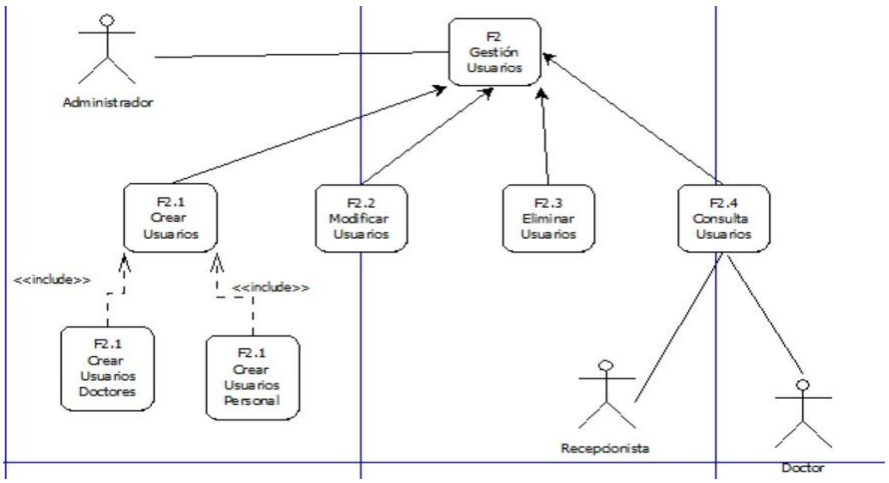
**Flujo Alterno**:

5. No existe el rol → Ver Caso de Uso F1.1 para crear el rol.

**Caso de Uso: Gestión de Usuarios**

***Figura 8***

*Diagrama de Usuarios*




*Nota*. Diagrama detallado sobre la segunda función de Gestión de Usuarios con los Doctores y Personal

**Descripción**: Implementar funcionalidades para manejar la información profesional de los doctores y pacientes.

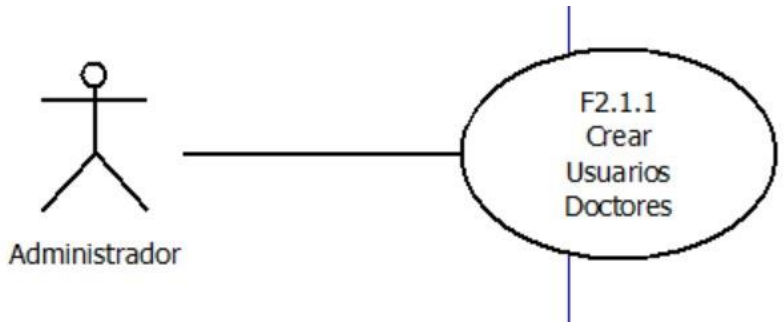
**Actores**: Administrador, Recepcionista y Doctor

El diagrama en la **Figura 8** ilustra el caso de uso para la gestión de usuarios, destacando las interacciones con los doctores y el personal. A continuación, se detalla el primer subproceso:

**F2.1.1: Crear Usuarios Doctores**

**Figura 9**

*Diagrama a Detalle de Crear Usuarios de Doctores*




*Nota*. Diagrama a Detalle del Crear un Usuario de Doctores que solo puede el Administrador.

**Descripción**: El administrador puede crear usuarios específicos para los doctores.

**Actor**: Administrador

El diagrama en la **Figura 9** muestra el proceso detallado para crear usuarios específicos para los doctores. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Usuarios" del menú principal.
2. El sistema presenta la ventana de usuarios.
3. El actor selecciona "Crear Usuario Doctor".
4. El actor ingresa los datos del doctor.
5. El actor ingresa el calendario que tendría disponible.

6. El actor presiona "Guardar".

7. El sistema revisa que el usuario no existe (E1).

8. El sistema almacena los datos del nuevo usuario doctor (E2).

En la **Tabla 8** se muestran las excepciones en el proceso de crear usuarios de doctores, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 8**

*Excepciones en el proceso de Crear Usuarios de Doctores*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El usuario ya existe | Ir al caso de uso F2.2 o F2.3 |
| E2 | No se almacenan los datos | Consultar con el administrador. |

*Nota*. Tabla de excepciones al Crear un Usuario de Doctores con sus causas y soluciones.

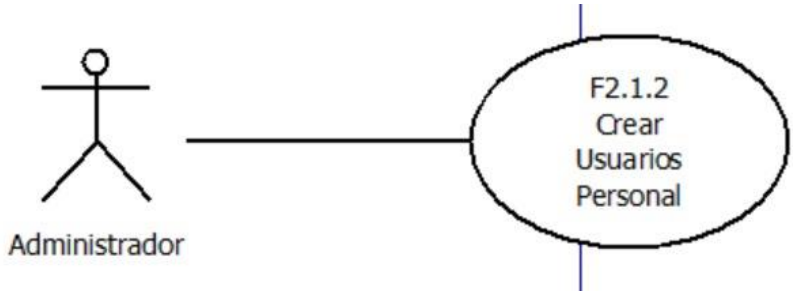
**Flujo Alterno**:

7. El usuario ya existe → Ver Caso de Uso F2.2 o F2.3 para modificar o eliminar.

**F2.1.2: Crear Usuarios Personal**

**Figura 10**

*Diagrama a Detalle de Crear Usuarios de Personal*




*Nota*. Diagrama a Detalle de Crear un Usuario de Personal que solo puede hacer el Administrador.

**Descripción**: El administrador puede crear usuarios específicos para el personal.

**Actor**: Administrador

El diagrama en la **Figura 10** muestra el proceso detallado para crear usuarios específicos para el personal. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Usuarios" del menú principal.
2. El sistema presenta la ventana de usuarios.
3. El actor selecciona "Crear Usuario Personal".
4. El actor ingresa los datos del personal.
5. El actor presiona "Guardar".
6. El sistema verifica si el usuario existe (E1).
7. El sistema almacena los datos del nuevo usuario personal (E2).

En la **Tabla 9** se muestran las excepciones en el proceso de crear usuarios de personal, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 9**

*Excepciones en el proceso de Crear Usuarios de Personal*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El usuario ya existe | Ir al caso de uso F2.2 o F2.3 |
| E2 | No se almacenan los datos | Consultar con el administrador. |

*Nota*. Tabla de excepciones al Crear un Usuario de Personal con sus causas y soluciones

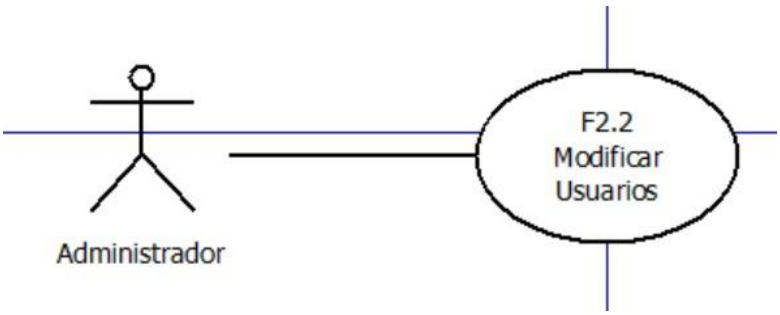
**Flujo Alterno**:

6. El usuario ya existe → Ver Caso de Uso F2.2 o F2.3 para modificar o eliminar.

**F2.2: Modificar Usuarios**

**Figura 11**

*Diagrama a Detalle de Modificar Usuarios*




*Nota*. Diagrama a Detalle de Modificar un Usuario que solo puede hacer el Administrador

**Descripción**: El administrador puede modificar la información de los usuarios existentes.

**Actor**: Administrador

El diagrama en la **Figura 11** ilustra el proceso detallado para modificar usuarios. El administrador puede modificar la información de los usuarios existentes. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Usuarios" del menú principal.
2. El sistema presenta la ventana de usuarios.
3. El actor busca un usuario que quiera modificar (E1).
4. El sistema carga los datos del usuario.
5. El actor modifica los datos.
6. El actor presiona "Guardar".
7. El sistema almacena los datos modificados (E2).

En la **Tabla 10** se muestran las excepciones en el proceso de modificar usuarios, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 10**

*Excepciones en el proceso de Modificar Usuarios*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El usuario no existe | Mostrar mensaje de error. |
| E2 | No se almacenan los datos | Consultar con el administrador. |

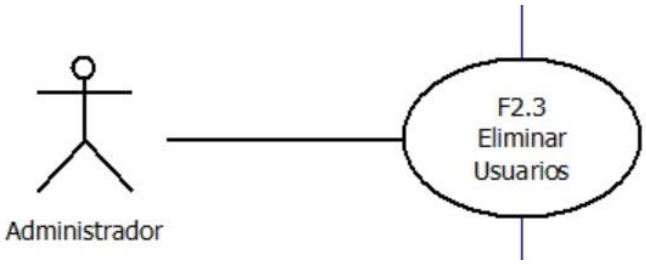
*Nota*. Tabla de excepciones al Modificar un Usuario con sus causas y soluciones

**Flujo Alterno**: 4. Si el usuario no existe, se muestra un mensaje de error (E1).

**F2.3: Eliminar Usuarios**

**Figura 12**

*Diagrama a Detalle de Eliminar Usuarios*




*Nota*. Diagrama a Detalle de Eliminar un Usuario que solo puede hacer el Administrador

**Descripción**: El administrador puede eliminar usuarios que ya no sean necesarios.

**Actor**: Administrador

El diagrama en la **Figura 12** muestra el proceso detallado para eliminar usuarios. El administrador puede eliminar usuarios que ya no sean necesarios. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Usuarios" del menú principal.
2. El sistema presenta la ventana de usuarios.

3. El actor busca un usuario que quiera eliminar (E1).
4. El actor presiona "Eliminar".
5. El sistema solicita confirmación.
6. El actor confirma la eliminación.
7. El sistema elimina los datos del usuario (E2).

En la **Tabla 11** se muestran las excepciones en el proceso de eliminar usuarios, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 11**

*Excepciones en el proceso de Eliminar Usuarios*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El usuario no existe | Ir al caso de uso F2.1.1 o F2.1.2. |
| E2 | No se eliminan los datos | Consultar con el administrador. |

*Nota*. Tabla de excepciones de Eliminar un Usuarios con sus causas y soluciones.

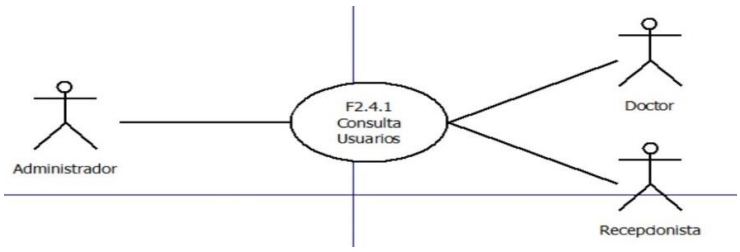
**Flujo Alterno**:

3. No existe el usuario → Ver Caso de Uso F2.1.1 o F2.1.2 para crear el usuario.

**F2.4: Consultar Usuarios**

**Figura 13**

*Diagrama a Detalle de Consultar Usuarios*




*Nota*. Diagrama a Detalle de Consultar Usuarios que pueden el Administrador, Recepcionista y Doctor.

**Descripción**: El administrador puede consultar la información de los usuarios registrados.

**Actores**: Administrador, Doctor y Recepcionista

El diagrama en la **Figura 13** muestra el proceso detallado para consultar usuarios. El administrador, doctor y recepcionista pueden consultar la información de los usuarios registrados. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Usuarios" del menú principal.
2. El sistema presenta la ventana de usuarios.
3. El actor busca un usuario existente.
4. El sistema muestra los datos del usuario (E1).

En la **Tabla 12** se muestran las excepciones en el proceso de consultar usuarios, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 12**

*Excepciones en el proceso de Consultar Usuarios*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El usuario no existe | Ir al caso de uso F2.1.1 o F2.1.2 |

*Nota*. Tabla de excepciones al Consultar un Usuario con su causa y solución.

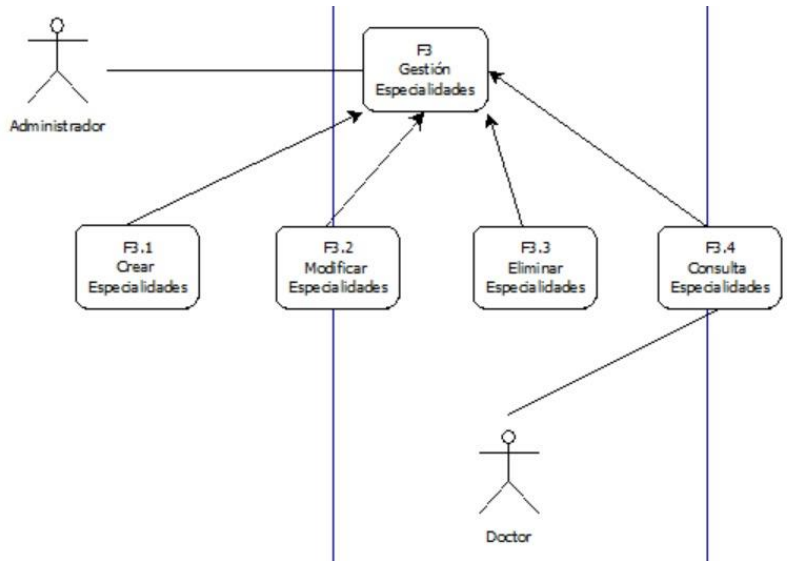
**Flujo Alterno**:

4. No existe el usuario → Ver Caso de Uso F2.1.1 o F2.1.2 para crear el usuario.

**Caso de Uso: Gestión de Especialidades**

**Figura 14**

*Diagrama de Especialidades*




Nota. Diagrama detallado sobre la tercera función de Gestión de Especialidades

El diagrama en la **Figura 14** ilustra el caso de uso para la gestión de especialidades, destacando las interacciones con el administrador y el doctor.

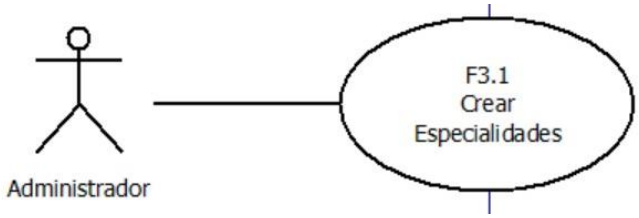
**Descripción**: Crear un sistema para gestionar las especialidades médicas dentro de la clínica.

**Actores:** Administrador y Doctor

**F3.1: Registrar Especialidades**

**Figura 15**

*Diagrama a Detalle de Crear Especialidades*




*Nota*. Diagrama a Detalle de Crear una Especialidad que solo puede hacer el Administrador.

**Descripción**: El administrador puede registrar nuevas especialidades médicas.

**Actor**: Administrador

El diagrama en la **Figura 15** muestra el proceso detallado para registrar nuevas especialidades médicas. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Especialidades" del menú principal.
2. El sistema presenta la ventana de especialidades.
3. El actor ingresa los datos de la especialidad.
4. El actor presiona "Guardar".
5. El sistema verifica si la especialidad ya existe (E1)
6. El sistema almacena los datos de la nueva especialidad (E2).

En la **Tabla 13** se muestran las excepciones en el proceso de crear especialidades, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 13**

*Excepciones en el proceso de Crear Especialidades*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | La especialidad ya existe | Ir al caso de uso F3.2 o F3.3 |
| E2 | No se almacenan los datos | Consultar con el administrador. |

*Nota*. Tabla de excepciones de Crear una Especialidad con sus causas y soluciones.

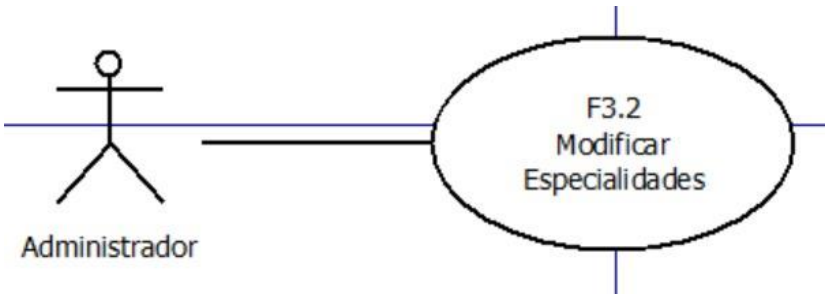
**Flujo Alterno**:

5. La especialidad ya existe → Ver Caso de Uso F3.2 o F3.3 para modificar o eliminar.

**F3.2: Modificar Especialidades**

**Figura 16**

*Diagrama a Detalle de Modificar Especialidades*




*Nota*. Diagrama a Detalle de Modificar una Especialidad que solo pude hacer el Administrador.

**Descripción**: El administrador puede actualizar la información de especialidades existentes.

**Actor**: Administrador

El diagrama en la **Figura 16** ilustra el proceso detallado para modificar especialidades. El administrador puede actualizar la información de especialidades existentes. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Especialidades" del menú principal.
2. El sistema presenta la ventana de especialidades.
3. El actor busca una especialidad existente (E1).
4. El sistema carga los datos de la especialidad.
5. El actor modifica los datos.
6. El actor presiona "Guardar".
7. El sistema almacena los datos modificados (E2).

En la **Tabla 14** se muestran las excepciones en el proceso de modificar especialidades, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 14**

*Excepciones en el proceso de Modificar Especialidades*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | La especialidad no existe | Ir al caso de uso F3.1 |
| E2 | No se almacenan los datos | Consultar con el administrador. |

*Nota*. Tabla de excepciones de Modificar una Especialidad con sus causas y soluciones.

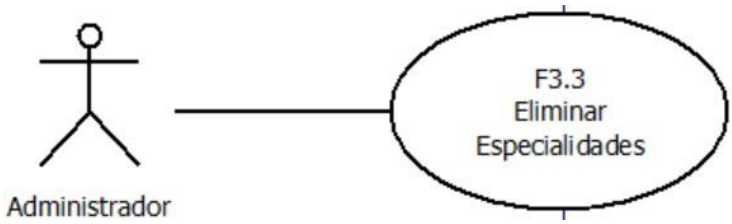
**Flujo Alterno**:

3. No existe la especialidad → Ver Caso de Uso F3.1 para crear la especialidad.

**F3.3: Eliminar Especialidades**

**Figura 17**

*Diagrama a Detalle de Eliminar Especialidades*




Nota. Diagrama a Detalle de Eliminar una Especialidad que solo puede hacer el Administrador.

**Descripción**: El administrador puede eliminar especialidades que ya no sean necesarias.

**Actor**: Administrador

El diagrama en la **Figura 17** muestra el proceso detallado para eliminar especialidades. El administrador puede eliminar especialidades que ya no sean necesarias. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Especialidades" del menú principal.
2. El sistema presenta la ventana de especialidades.
3. El actor busca una especialidad existente (E1).
4. El sistema carga los datos de la especialidad.
5. El actor presiona "Eliminar".
6. El sistema solicita confirmación.
7. El actor confirma la eliminación.
8. El sistema elimina la especialidad (E2).

En la **Tabla 15** se muestran las excepciones en el proceso de eliminar especialidades, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 15**

*Excepciones en el proceso de Eliminar Especialidades*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | La especialidad no existe | Ir al caso de uso F3.1 |
| E2 | No se eliminan los datos | Consultar con el administrador. |

*Nota*. Tabla de excepciones de Eliminar una Especialidad con sus causas y soluciones.

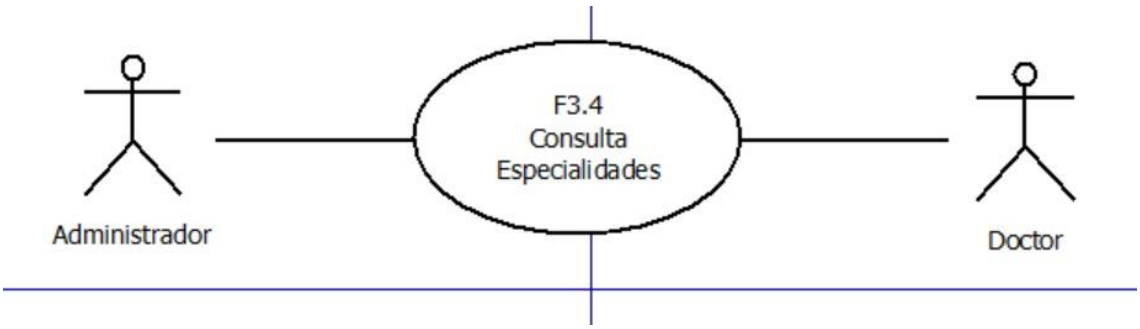
**Flujo Alterno**:

3. No existe la especialidad → Ver Caso de Uso F3.1 para crear la especialidad.

**F3.4: Consultar Especialidades**

**Figura 18**

*Diagrama a Detalle de Consultar Especialidades*




*Nota*. Diagrama a Detalle de Consultar una Especialidad que puede el Administrador y el Doctor.

**Descripción**: El administrador puede consultar las especialidades registradas.

**Actores**: Administrador y Doctor

El diagrama en la **Figura 18** muestra el proceso detallado para consultar especialidades. El administrador y el doctor pueden consultar las especialidades registradas. A continuación, se describe el flujo principal:

**Flujo Principal**:

1. El actor selecciona la opción "Gestión de Especialidades" del menú principal.
2. El sistema presenta la ventana de especialidades.
3. El actor busca una especialidad existente.
4. El sistema muestra los datos de la especialidad (E1).

En la **Tabla 16** se muestran las excepciones en el proceso de consultar especialidades, incluyendo sus causas y soluciones.

**Excepciones**:

**Tabla 16**

*Excepciones en el proceso de Consultar Especialidades*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | La especialidad no existe | Ir al caso de uso F3.1 |

*Nota*. Tabla de excepciones de Consultar una Especialidad con su causa y solución

**Flujo Alterno**:

4. No existe la especialidad → Ver Caso de Uso F3.1 para crear la especialidad.

El Sprint Backlog 1 establece una base sólida para el desarrollo del sistema, enfocándose en la gestión eficiente de roles y permisos, la administración del personal y la gestión de especialidades. Estas funcionalidades son esenciales para garantizar que el sistema cumpla con los requisitos operativos y de seguridad de la Clínica Dental ARPUL.

### 4.1.2 Diseño 1

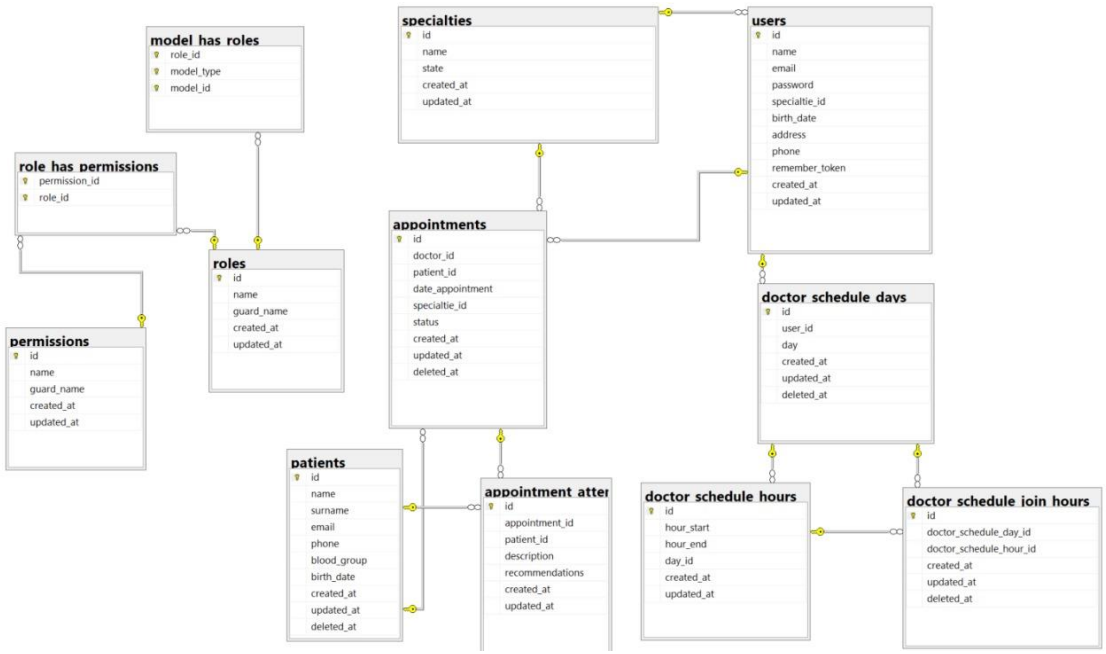
El diseño en el Sprint 1 establece la arquitectura de la base de datos y las interfaces de usuario necesarias para soportar la autenticación de usuarios, gestión de roles y permisos, manejo de la información del personal médico y administrativo, y la organización de especialidades médicas. Este diseño proporciona la fundación para todas las operaciones clínicas y administrativas que se llevarán a cabo en el sistema.

**Base de Datos**

El diseño de la base de datos se enfoca en establecer relaciones claras y eficientes entre entidades como usuarios, roles, permisos, especialidades, pacientes, citas médicas y horarios de los doctores. El Diagrama Entidad-Relación muestra cómo cada tabla está interconectada, facilitando la comprensión del flujo de datos y la integridad del sistema.

**Figura 19**

*Diagrama Entidad - Relación de la BBDD*




*Nota*. Modelo de la BD de las necesidad primordiales del sistema

La **Figura 19** presenta el Diagrama Entidad-Relación que detalla cómo cada tabla está interconectada, facilitando la comprensión del flujo de datos y la integridad del sistema.

- **Tabla de Especialidades Médicas (specialties)**
    - **Propósito**: Almacenar las diferentes especialidades médicas disponibles en la clínica. Esto es crucial para asociar doctores con sus respectivas áreas de especialización y para gestionar las citas por especialidad.
    - **Relaciones**: Se relaciona con la tabla de usuarios (users) para indicar la especialidad de cada doctor.
- **Tablas de Roles y Permisos (roles, permissions, role_has_permissions, model_has_roles)**
    - **Propósito**: Gestionar los diferentes roles dentro del sistema (como admin, doctor, recepcionista) y definir los permisos específicos que cada rol puede tener, asegurando un control de acceso adecuado y seguridad en la gestión de datos.

  - **Relaciones**: role_has_permissions vincula roles con permisos en una relación muchos a muchos. model_has_roles conecta los usuarios con roles, permitiendo que varios usuarios tengan múltiples roles.

- **Tabla de Usuarios (users)**
  - **Propósito**: Registrar todos los usuarios del sistema, incluyendo personal médico y administrativo. Contiene información personal y profesional, además de credenciales de acceso.
  - **Relaciones**: Se vincula con citas médicas (appointments) para identificar a los doctores y con especialidades médicas (specialties) para indicar su especialidad.

- **Tabla de Pacientes (patients)**
  - **Propósito**: Contener la información de los pacientes, como nombre, contactos, y datos médicos relevantes.
  - **Relaciones**: Fundamental para la tabla de citas médicas (appointments), donde se registran las citas médicas de los pacientes.

- **Tabla de Citas Médicas (appointments)**
  - **Propósito**: Registrar cada cita médica, incluyendo el doctor asignado, el paciente, la fecha, la hora, y el estado de la cita.
  - **Relaciones**: Conecta pacientes con doctores y utiliza la información de la tabla de especialidades (specialties) para la especialidad requerida.

- **Tablas de Horarios (doctor_schedule_days, doctor_schedule_hours, doctor_schedule_join_hours)**
  - **Propósito**: Estas tablas gestionan los horarios de los doctores, detallando los días y horas específicas en que están disponibles para consultas.
  - **Relaciones**: doctor_schedule_days y doctor_schedule_hours están interrelacionadas para ofrecer un horario detallado. La tabla doctor_schedule_join_hours se usa para relacionar los días con las horas específicas.

- **Tabla de Atención Médica (appointment_attentions)**
    - **Propósito**: Capturar detalles específicos de cada atención médica durante las citas, como diagnósticos y recomendaciones.
    - **Relaciones**: Enlaza directamente con la tabla de citas médicas (appointments) y la tabla de pacientes (patients), permitiendo un seguimiento detallado de cada consulta médica.

Este diseño robusto y detallado es esencial para asegurar que el sistema pueda manejar eficientemente la información médica y administrativa, proporcionando una plataforma sólida para la expansión futura y la incorporación de nuevas funcionalidades en sprints posteriores.

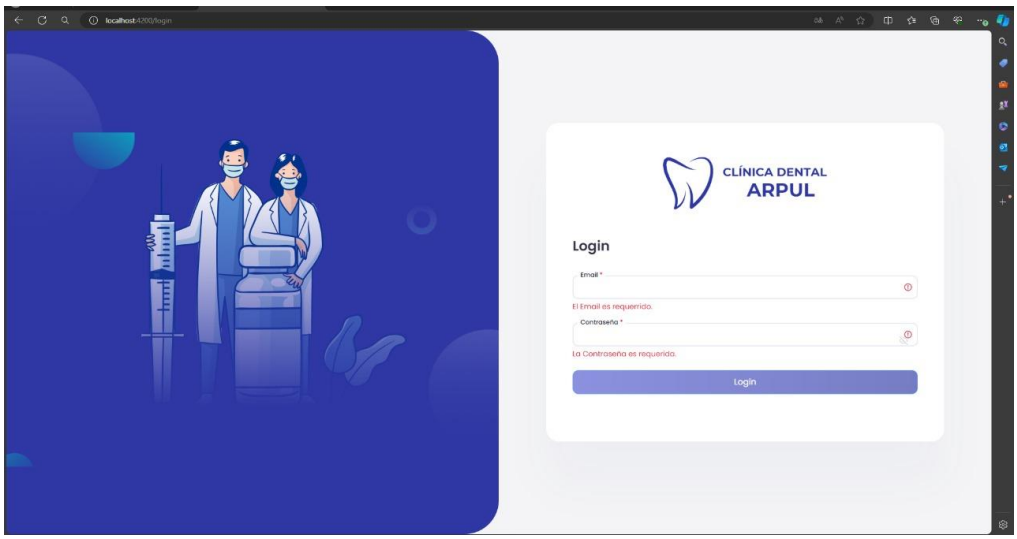
**Interfaces de Usuario**

Las interfaces de usuario están diseñadas para facilitar la interacción eficiente con el sistema, proporcionando accesibilidad y seguridad en la gestión de datos y operaciones del sistema. Se describen a continuación las interfaces específicas para cada funcionalidad relevante del Sprint 1:

1. **Autenticación y Login**
    - **Propósito:** Garantizar que solo los usuarios autorizados puedan acceder al sistema mediante credenciales seguras.

**Figura 20**

*Pestaña de Login para los Usuarios de la Clínica*




*Nota*. Pestaña en un navegador de como se ve el Login del sistema

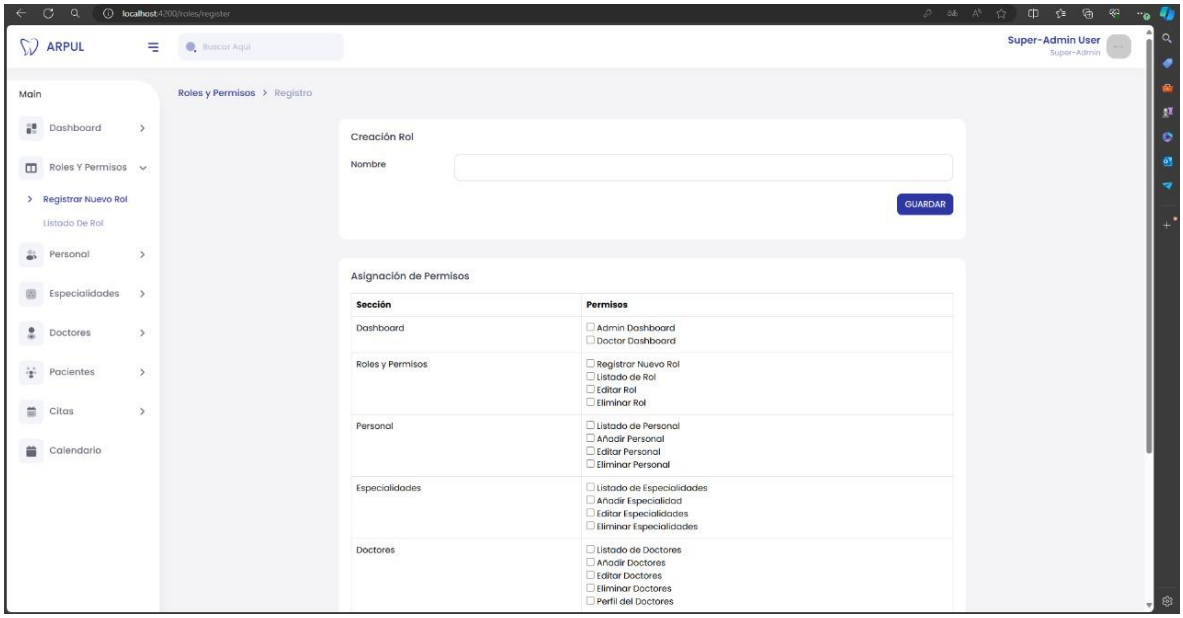
La **Figura 20** muestra la pestaña desarrollada para la presentación del login de los usuarios de la clínica.

2. **Gestión de Roles y Permisos**

   - **Propósito:** Configurar los roles dentro del sistema (como administrador, doctor, recepcionista) y definir los permisos específicos que cada rol puede ejercer, para garantizar un adecuado control de acceso y seguridad de los datos.

**Figura 21**

*Pestaña de Registro de un Nuevo Rol*

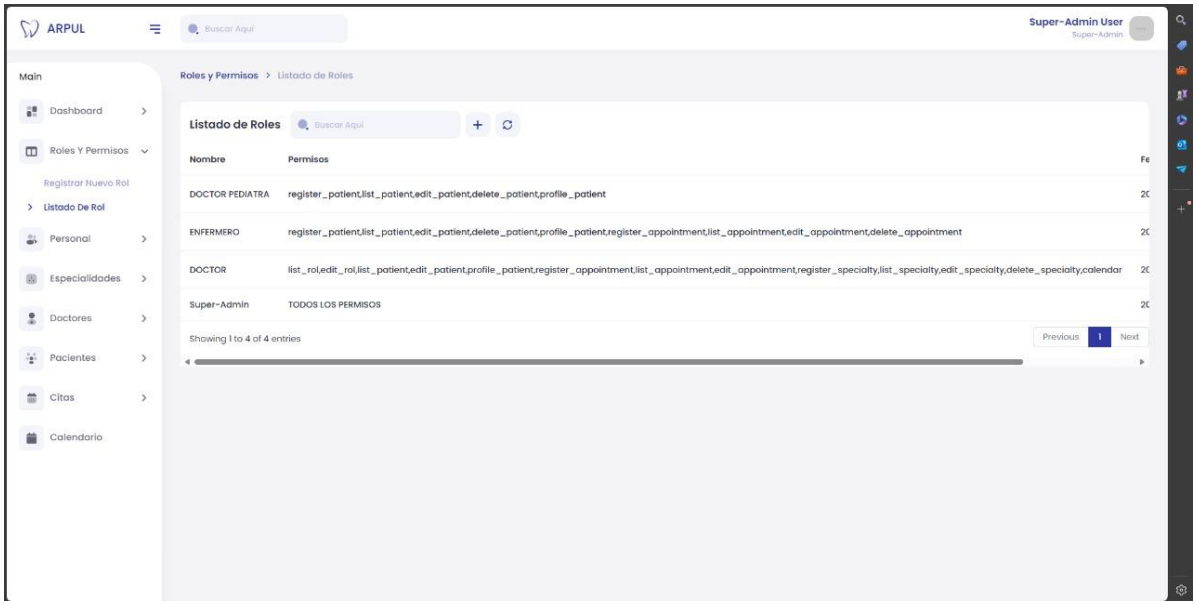



*Nota*. Pestaña que muestra cómo se Registra un Rol y se asignan los Permisos que quiera tener ese Rol o las cosas que pueda gestionar

La **Figura 21** muestra la pestaña de registro de un nuevo rol, donde se pueden asignar permisos específicos.

**Figura 22**

*Pestaña de Listado de Roles con sus Permisos*




*Nota*. Pestaña que muestra el listado de los Roles existentes y los Permisos que tiene ese Rol.

La **Figura 22** presenta la pestaña de listado de roles con sus permisos correspondientes.

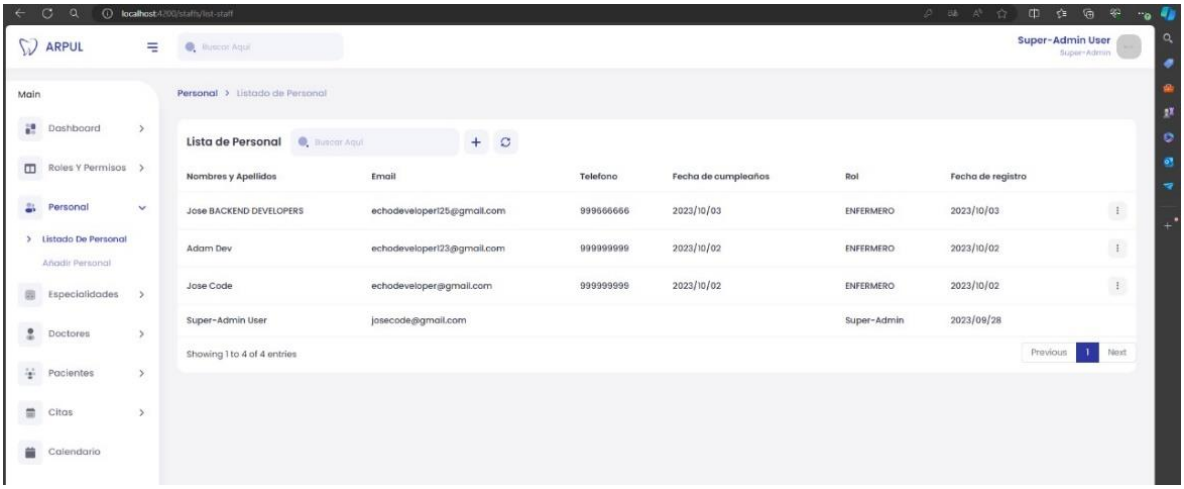
3. **Gestión de Usuarios**

    - **Doctores y Personal Administrativo:** Implementación de interfaces para manejar la información profesional y personal, facilitando la creación, modificación y visualización de perfiles de usuario.

    - Algo importante a mencionar en este campo es que al indicar los resultados y tener una reunión con el dueño de la clínica, quiso que se añada el campo de avatar, que va a ser la imagen del doctor y del personal.

**Figura 23**

*Pestaña del Listado de Usuarios del Personal*

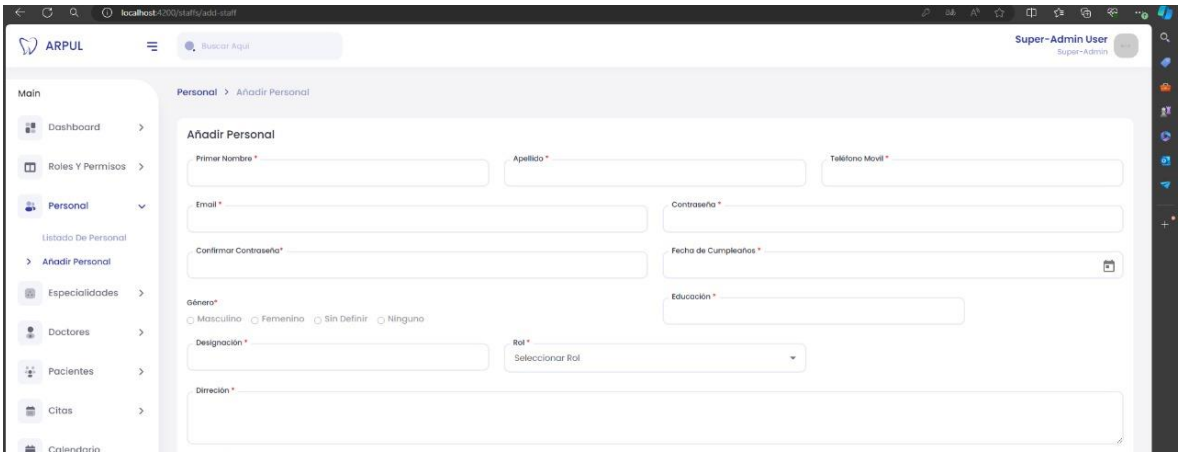



*Nota*. Pestaña que muestra los Usuarios del Personal.

La **Figura 23** muestra la pestaña desarrollada para la presentación de los usuarios del personal.

**Figura 24**

*Pestaña para Añadir un Usuario de Personal*

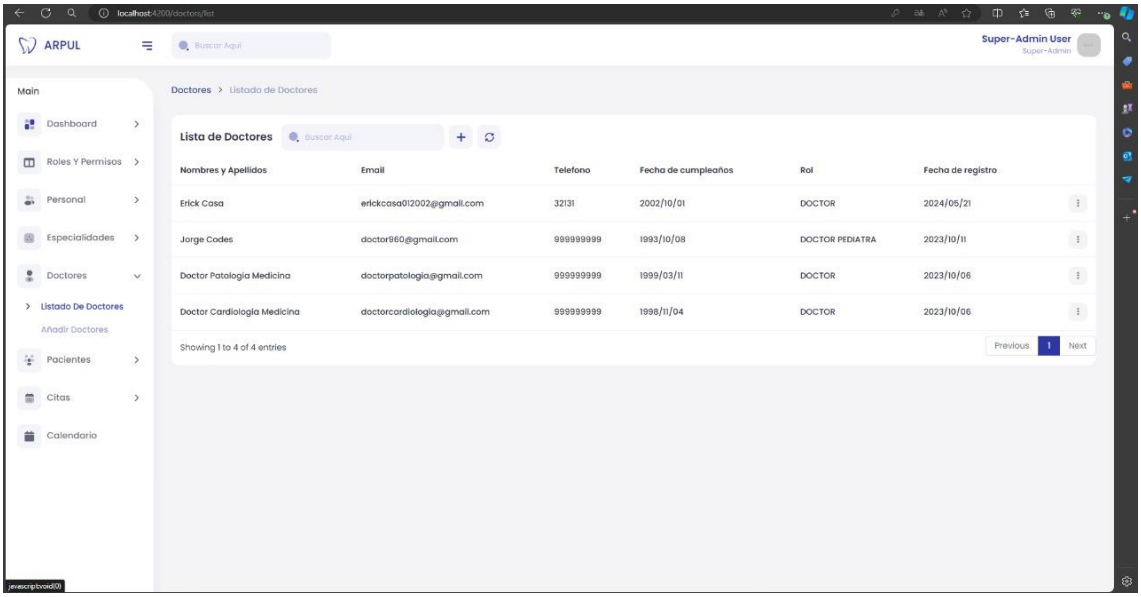



*Nota*. Pestaña que muestra cómo se Registra un Usuario de Personal.

La **Figura 24** muestra la pestaña utilizada para registrar un nuevo usuario de personal.

**Figura 25**

*Pestaña con el Listado de Usuarios de Doctores*

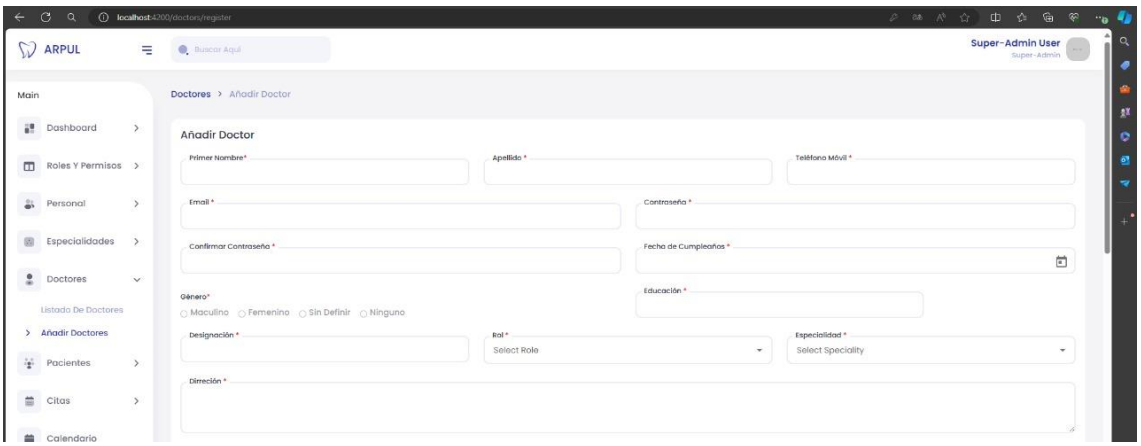



*Nota*. Pestaña que muestra los Usuarios de Doctores.

La **Figura 25** presenta la pestaña con el listado de usuarios de doctores.

**Figura 26**

*Pestaña para Añadir un Nuevo Usuario de Doctor*




*Nota*. Pestaña que muestra cómo se Registra un Usuario de Doctores.

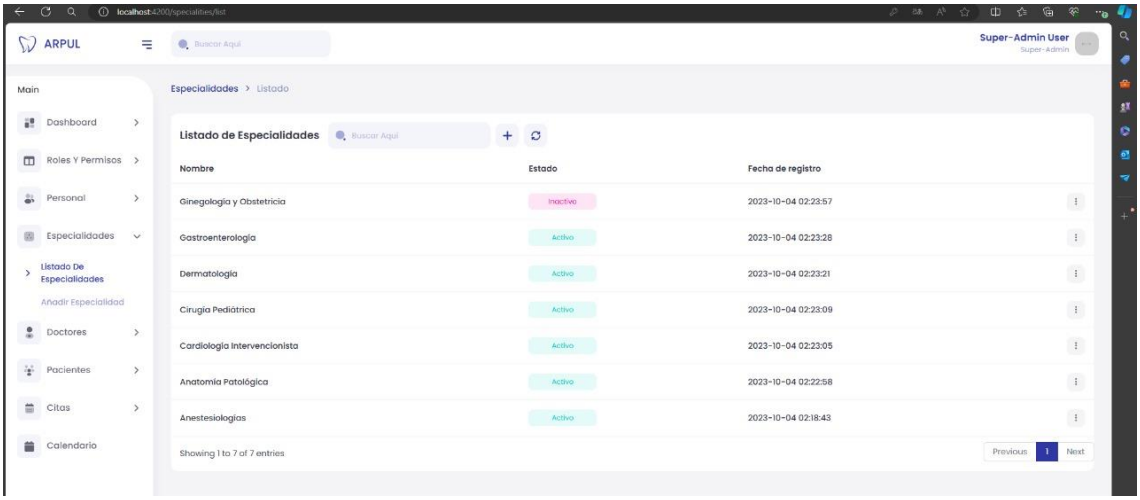
La **Figura 26** muestra la pestaña utilizada para añadir un nuevo usuario de doctor.

4. **Módulo de Especialidades Médicas**

    - **Propósito:** Administrar las especialidades médicas ofrecidas en la clínica, crucial para la organización de los servicios y la asignación de doctores.

**Figura 27**

*Pestaña del Listado de Especialidades*

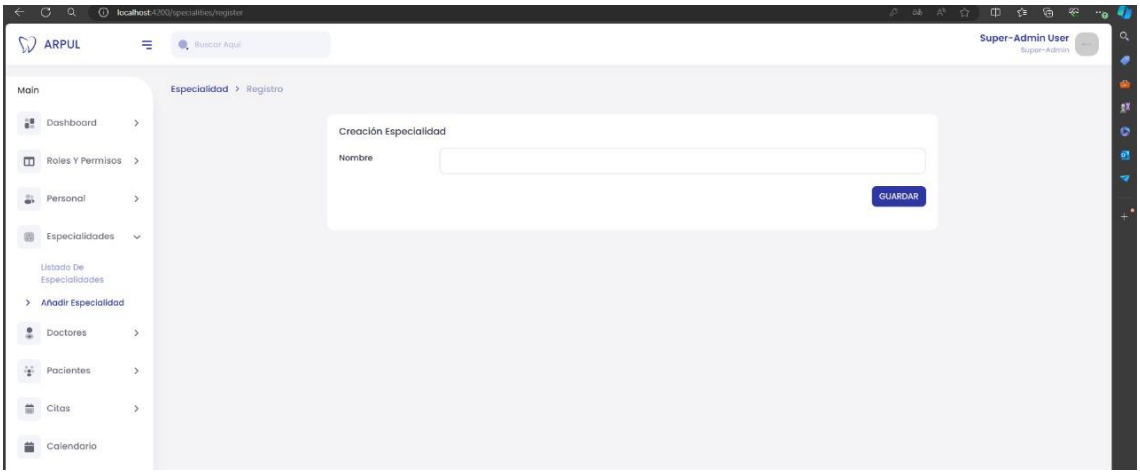



*Nota*. Pestaña que muestran las Especialidades y si se encuentran activas o no.

La **Figura 27** muestra la pestaña desarrollada para la presentación del listado de especialidades médicas.

**Figura 28**

*Pestaña para Añadir una Nueva Especialidad*




*Nota*. Pestaña que muestra como añadir una Especialidad.

La **Figura 28** muestra la pestaña utilizada para añadir una nueva especialidad médica.

Este diseño enfoca claramente en los requerimientos iniciales del sistema, asegurando que todas las funcionalidades esenciales para el funcionamiento inicial de la clínica estén correctamente implementadas y probadas. Este enfoque estratégico no solo facilita una

implementación efectiva sino que también establece una base sólida para futuras expansiones y mejoras en los sprints subsiguientes.

### 4.1.3 Codificación 1

Durante el Sprint 1, establecimos las bases cruciales de nuestro sistema, centrados en la seguridad y gestión de usuarios. Implementamos autenticación robusta y sistemas de gestión de roles y permisos, esenciales para la seguridad y funcionalidad eficaz del sistema. Con el tema de la codificación, se puede destacar los siguientes temas:

1. **JWT (JSON Web Tokens)**
    - **Propósito:** Utilizado para autenticar y mantener sesiones de usuarios en el sistema.
    - **Implementación:** Se integra en Laravel mediante la librería Tymon's JWT-auth para manejar la autenticación a través de tokens. Estos tokens son esenciales para la seguridad de las sesiones, ya que permiten verificar la identidad de los usuarios sin almacenar su estado de sesión en el servidor.

**Figura 29**

*Autenticación JWT*

```
1 reference | 0 overrides
public function login()
{
    $credentials = request(['email', 'password']);

    if (! $token = auth('api')->attempt($credentials)) {
        return response()->json(['error' => 'Unauthorized'], 401);
    }

    return $this->respondWithToken($token);
}
```

*Nota*. Implementación de JWT para la autenticación de usuarios utilizando credenciales de correo electrónico y contraseña en un entorno de API.

La ***Figura 29*** muestra la implementación de JWT para la autenticación de usuarios utilizando credenciales de correo electrónico y contraseña en un entorno de API.

2. **Laravel Permission (spatie/laravel-permission)**

- **Propósito:** Gestionar permisos y roles dentro de la aplicación.
- **Implementación:** Esta biblioteca se usa para definir roles como administrador, doctor, etc., y asignar permisos específicos a estos roles. Permite una gestión flexible de accesos según el rol de cada usuario, haciendo uso de middlewares para controlar el acceso a diferentes partes de la aplicación.

**Figura 30**

*Gestión de Roles y Permisos*

```
0 references | 0 overrides
public function run()
{
    // Reset cached roles and permissions
    app()[PermissionRegistrar::class]->forgetCachedPermissions();

    // create permissions
    Permission::create(['guard_name' => 'api','name' => 'register_rol']);
    Permission::create(['guard_name' => 'api','name' => 'list_rol']);
    Permission::create(['guard_name' => 'api','name' => 'edit_rol']);
    Permission::create(['guard_name' => 'api','name' => 'delete_rol']);

    Permission::create(['guard_name' => 'api','name' => 'register_doctor']);
    Permission::create(['guard_name' => 'api','name' => 'list_doctor']);
    Permission::create(['guard_name' => 'api','name' => 'edit_doctor']);
    Permission::create(['guard_name' => 'api','name' => 'delete_doctor']);
    Permission::create(['guard_name' => 'api','name' => 'profile_doctor']);

    Permission::create(['guard_name' => 'api','name' => 'register_patient']);
    Permission::create(['guard_name' => 'api','name' => 'list_patient']);
    Permission::create(['guard_name' => 'api','name' => 'edit_patient']);
    Permission::create(['guard_name' => 'api','name' => 'delete_patient']);
    Permission::create(['guard_name' => 'api','name' => 'profile_patient']);
```

*Nota*. Código para definir y gestionar roles y permisos dentro del sistema, utilizando la biblioteca Laravel Permission.

La **Figura 30** presenta el código utilizado para definir y gestionar roles y permisos dentro del sistema, utilizando la biblioteca Laravel Permission.

3. **Middleware API**
   - **Propósito:** Regular el acceso a las rutas de la API basándose en roles y permisos.
   - **Implementación:** En Laravel, se configura un middleware que intercepta las solicitudes HTTP para verificar si un usuario tiene los permisos necesarios para acceder a ciertas rutas. Este mecanismo es clave para la seguridad del sistema y asegura que solo los usuarios autorizados puedan realizar operaciones sensibles.

**Figura 31**

*Manejo de Tokens con JWT*

```
        if (! $token = auth('api')->attempt($credentials)) {
            return response()->json(['error' => 'Unauthorized'], 401);
        }

        return $this->respondWithToken($token);
    }
```

*Nota*. Código que maneja la verificación de credenciales y la emisión de tokens JWT para usuarios autenticados, proporcionando acceso seguro a la API.

La **Figura 31** muestra el código que maneja la verificación de credenciales y la emisión de tokens JWT para usuarios autenticados, proporcionando acceso seguro a la API.

4. **Laravel Eloquent**

   - **Propósito:** ORM utilizado para la interacción con la base de datos.
   - **Implementación:** Eloquent permite una fácil manipulación de la base de datos mediante objetos y métodos en lugar de SQL crudo. Esto se usa ampliamente para todas las operaciones de base de datos como la creación de usuarios, manejo de roles, registros de citas, etc.

**Figura 32**

*Registro de Usuarios*

```
1 reference | 0 overrides
public function register() {

    $validator = Validator::make(request()->all(), [
        'name' => 'required',
        'email' => 'required|email|unique:users',
        'password' => 'required|min:8',
    ]);

    if($validator->fails()){
        return response()->json($validator->errors()->toJson(), 400);
    }

    $user = new User;
    $user->name = request()->name;
    $user->email = request()->email;
    $user->password = bcrypt(request()->password);
    $user->save();

    return response()->json($user, 201);
}
```

*Nota*. Proceso de registro de nuevos usuarios en el sistema, validando los requisitos de los datos y almacenando información segura.

La **Figura 32** muestra el proceso de registro de nuevos usuarios en el sistema, validando los requisitos de los datos y almacenando información segura.

Estas herramientas y métodos son fundamentales para el Sprint 0, enfocado en establecer una base sólida para la autenticación y gestión de usuarios y roles. Cada componente ha sido elegido por su robustez y compatibilidad con Laravel, asegurando un desarrollo eficiente y seguro del sistema.

### 4.1.4 Pruebas 1

Al realizar las pruebas intensivas para validar tanto la funcionalidad como la usabilidad de las nuevas características implementadas en el sistema. Estas pruebas son esenciales para asegurar que el sistema no solo cumpla con las especificaciones técnicas y de negocio, sino que también brinde una experiencia de usuario óptima. La realización de pruebas de usabilidad y funcionalidad ayuda a detectar problemas tempranos en el desarrollo y a asegurar que las soluciones sean efectivas y accesibles para los usuarios finales.

En la **Figura 33** se muestran todas las pruebas realizadas en el 1er Sprint, abarcando el login del sistema, la funcionalidad de roles, usuarios y de especialidades; estas de igual forma con la respectiva firma de revisión por parte del propietario e indiciando cómo va el desarrollo de la aplicación con respecto a sus especificaciones.

**Figura 33**

*Pruebas del Sprint 1*

**Pruebas del Sprint 1**

| ID | Prueba | Descripción | Tipo de Prueba | Resultado Esperado | Status |
|---|---|---|---|---|---|
| P1 | Inicio de sesión válido | Probar el acceso con credenciales válidas. | Funcionalidad | El usuario accede correctamente al sistema. | ✓ |
| P2 | Inicio de sesión inválido | Intentar acceso con credenciales incorrectas. | Funcionalidad | Mensaje de error apropiado y acceso denegado. | ✓ |
| P3 | Respuesta a expiración del token | Acceso con un token expirado. | Usabilidad | Mensaje claro de sesión expirada y solicitud de relogin. | ✓ |
| P4 | Creación de roles | Crear un nuevo rol dentro del sistema. | Funcionalidad | El rol se crea correctamente y es asignable. | ✓ |
| P5 | Asignación de permisos a roles | Asignar permisos específicos a un rol creado. | Funcionalidad | Los permisos se reflejan correctamente en el rol. | ✓ |
| P6 | Restricción de acceso sin permisos | Acceso a función sin el permiso necesario. | Usabilidad | Acceso denegado con mensaje explicativo. | ✓ |
| P7 | Registro de nuevo doctor o personal | Crear un perfil de doctor o personal con datos completos. | Funcionalidad | El doctor o personal se registra sin errores. | ✓ |
| P8 | Edición de perfil de usuario | Modificar datos de un usuario existente. | Usabilidad | Los cambios se guardan correctamente. | ✓ |
| P9 | Eliminación de usuario | Eliminar un usuario inactivo. | Funcionalidad | El usuario se elimina correctamente del sistema. | ✓ |
| P10 | Añadir nueva especialidad | Registrar una nueva especialidad médica. | Funcionalidad | La especialidad se añade correctamente. | ✓ |
| P11 | Modificar especialidad | Cambiar detalles de una especialidad existente. | Usabilidad | Los cambios se reflejan correctamente. | ✓ |
| P12 | Eliminación de especialidad obsoleta | Eliminar una especialidad que ya no se ofrece. | Funcionalidad | La especialidad se elimina sin afectar otras entidades. | ✓ |

Erick Alexander Casa Ortiz
Desarrollador
CI: 1727389627

Dr. Paúl Esteban Casa Ortiz
Propietario Clínica Dental ARPUL
CI: 1721161089

*Nota*. Tabla resumiendo las pruebas realizadas con la respectiva firma de los involucrados.

Estas pruebas validan que el sistema pueda gestionar adecuadamente las especialidades médicas, una parte crucial para la organización de servicios en la clínica. Es importante que las especialidades se puedan añadir, modificar y eliminar sin complicaciones para adaptarse a las necesidades cambiantes del entorno médico.

## 4.2 Desarrollo Del Sprint 2

### 4.2.1 Sprint Backlog 2

El Sprint Backlog 2 se centra en la implementación de funcionalidades esenciales para la gestión de doctores, pacientes y citas médicas. Estas tareas son cruciales para establecer un sistema integral que permita manejar de manera eficiente la información médica y la programación de citas, asegurando una administración efectiva y segura en la clínica dental.

Para comenzar, se ha desarrollado un sistema robusto para registrar y seguir la información de los pacientes, además de implementar un sistema integral para la

programación y gestión de citas médicas. Estas funcionalidades aseguran que la clínica pueda ofrecer un servicio eficiente y organizado, mejorando la experiencia tanto para el personal como para los pacientes.

**Objetivos del Sprint 2**

**Objetivos Específicos:**

- **Módulo de Pacientes:** Desarrollar un sistema para registrar y seguir la información de los pacientes.
- **Citas Médicas:** Crear un sistema integral para la programación y gestión de citas médicas.

Estas funcionalidades son esenciales para garantizar que cada paciente reciba la atención adecuada y que las citas médicas se manejen de manera eficiente. Las entrevistas con el personal de la clínica revelaron la necesidad de un sistema que permita una administración fluida y organizada de las citas y la información de los pacientes.

**Product Backlog del Sprint 2**

En la **Tabla 17** se presenta el Product Backlog del Sprint 2, destacando las historias de usuario, criterios de aceptación y prioridades. Estas funcionalidades son esenciales para garantizar que cada paciente reciba la atención adecuada y que las citas médicas se manejen de manera eficiente.

**Tabla 17**

*Product Backlog del Sprint 2*

| ID | Historia de Usuario | Criterios de Aceptación | Prioridad |
|---|---|---|---|
| 1 | Como administrador, quiero poder registrar la información de los pacientes para llevar un control detallado. | Se debe poder registrar información personal y médica de los pacientes desde una interfaz administrativa. | Alta |
| 2 | Como recepcionista, quiero poder programar citas médicas para los pacientes para organizar el calendario. | El sistema debe permitir crear, modificar y cancelar citas médicas, y asignarlas a los doctores correspondientes. | Alta |
| 3 | Como doctor, quiero poder ver mi calendario de citas para planificar mis consultas. | El sistema debe mostrar el calendario de citas de cada doctor, incluyendo detalles de las citas y la información de los pacientes. | Media |
| 4 | Como administrador, quiero poder gestionar la información de los doctores para mantener actualizados sus datos. | Debe existir una interfaz para crear, modificar y eliminar doctores, incluyendo detalles como especialidad y horarios. | Alta |

*Nota*. Tabla sobre el Product Backlog del segundo sprint con las prioridades.

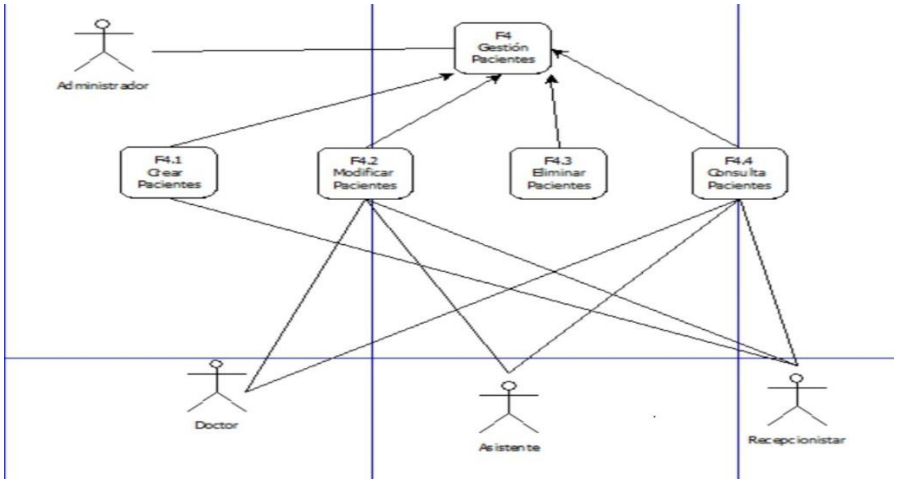
Tras revisar el Product Backlog, procederemos a explorar los casos de uso que describen la implementación práctica de estas funcionalidades, asegurando que el sistema cumpla con las necesidades operativas y de seguridad de la clínica.

**Detalle de Casos de Uso**

**Caso de Uso: Gestión de Pacientes**

**Figura 34**

*Diagrama de Gestión de Pacientes*



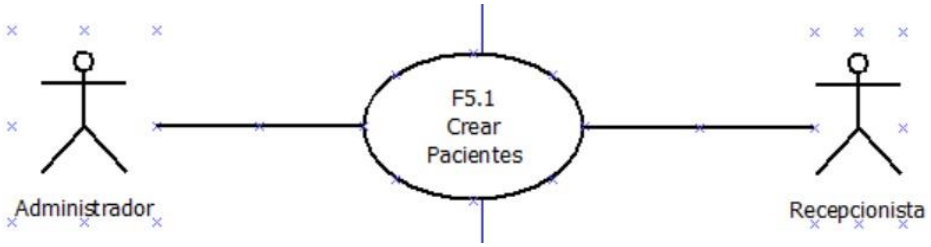

*Nota*. Diagrama detallado sobre la primera función de Gestión de Pacientes.

El diagrama en la **Figura 34** ilustra el caso de uso para la gestión de pacientes. Este diagrama detalla cómo se registran y gestionan los datos de los pacientes en el sistema, con los actores principales siendo el Administrador y la Recepcionista.

**F1.1: Crear Pacientes**

**Figura 35**

*Diagrama a Detalle de Crear Pacientes*




*Nota*. Diagrama a Detalle de Crear un Paciente que solo puede el Administrador o la Recepcionista.

**Descripción**: El administrador o la recepcionista pueden registrar nuevos pacientes en el sistema.

**Actores:** Administrador, Recepcionista

El diagrama en la **Figura 35** muestra el proceso detallado para registrar pacientes. A continuación, se describe el flujo principal:

**Flujo Principal:**

1. El actor selecciona la opción "Gestión de Pacientes" del menú principal.
2. El sistema presenta la ventana de pacientes.
3. El actor ingresa los datos del paciente.
4. El actor presiona "Guardar".
5. El sistema verifica si el paciente ya existe (E1).
6. El sistema almacena los datos del nuevo paciente (E2).

En la **Tabla 18** se muestran las excepciones en el proceso de registrar pacientes, incluyendo sus causas y soluciones.

**Excepciones:**

**Tabla 18**

*Excepciones en el Proceso de Registrar Pacientes*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El paciente ya existe | Ir al caso de uso F1.2 o F1.3 |
| E2 | No se almacenan los datos | Consultar con el administrador |

**Nota**. Tabla de excepciones al Registrar un Paciente con sus causas y soluciones.

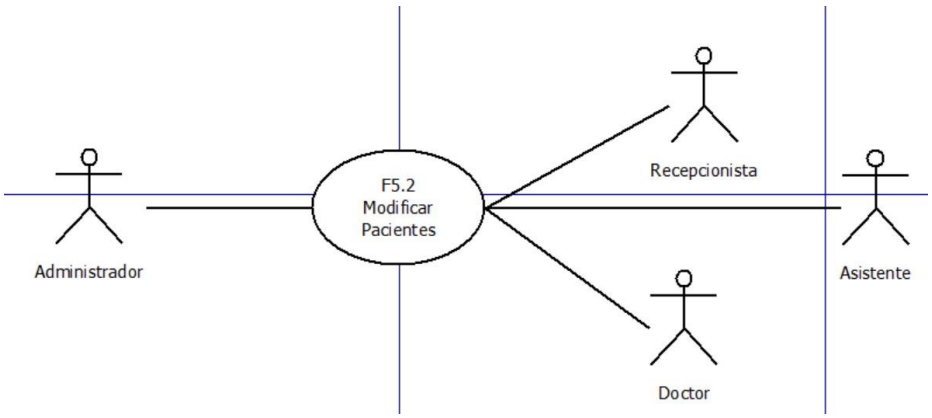
**Flujo Alterno:**

5. El paciente ya existe → Ver Caso de Uso F1.2 o F1.3 para modificar o eliminar.

**F1.2: Modificar Pacientes**

**Figura 36**

*Diagrama a Detalle de Modificar Pacientes*




*Nota*. Diagrama a Detalle de Modificar un Paciente que solo puede el Administrador o la Recepcionista.

**Descripción:** El administrador o la recepcionista pueden modificar la información de los pacientes existentes.

**Actores:** Administrador, Recepcionista

El diagrama en la **Figura 36** ilustra el proceso detallado para modificar pacientes. A continuación, se describe el flujo principal:

**Flujo Principal:**

1. El actor selecciona la opción "Gestión de Pacientes" del menú principal.
2. El sistema presenta la ventana de pacientes.
3. El actor busca el paciente que desea modificar (E1).
4. El sistema carga los datos del paciente.
5. El actor modifica los datos del paciente.
6. El actor presiona "Guardar".
7. El sistema almacena los datos modificados (E2).

En la **Tabla 19** se muestran las excepciones en el proceso de modificar pacientes, incluyendo sus causas y soluciones.

**Excepciones:**

**Tabla 19**

*Excepciones en el Proceso de Modificar Pacientes*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El paciente no existe | Ir al caso de uso F1.1 |
| E2 | No se almacenan los datos | Consultar con el administrador |

*Nota*. Tabla de excepciones al Modificar un Paciente con sus causas y soluciones.

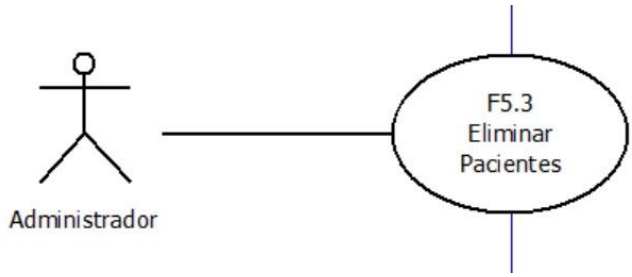
**Flujo Alterno:**

3. No existe el paciente → Ver Caso de Uso F1.1 para registrar el paciente.

**F1.3: Eliminar Pacientes**

**Figura 37**

*Diagrama a Detalle de Eliminar Pacientes*




*Nota*. Diagrama a Detalle de Eliminar un Paciente que solo puede el Administrador o la Recepcionista.

**Descripción:** El administrador o la recepcionista pueden eliminar pacientes que ya no sean necesarios.

**Actores:** Administrador, Recepcionista

El diagrama en la **Figura 37** muestra el proceso detallado para eliminar pacientes. A continuación, se describe el flujo principal:

**Flujo Principal:**

1. El actor selecciona la opción "Gestión de Pacientes" del menú principal.
2. El sistema presenta la ventana de pacientes.
3. El actor busca el paciente que desea eliminar (E1).
4. El actor presiona "Eliminar".
5. El sistema solicita confirmación.
6. El actor confirma la eliminación.
7. El sistema elimina los datos del paciente (E2).

En la **Tabla 20** se muestran las excepciones en el proceso de eliminar pacientes, incluyendo sus causas y soluciones.

**Excepciones:**

**Tabla 20**

*Excepciones en el Proceso de Eliminar Pacientes*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El paciente no existe | Ir al caso de uso F1.1 |
| E2 | No se eliminan los datos | Consultar con el administrador |

*Nota*. Tabla de excepciones al Eliminar un Paciente con sus causas y soluciones.

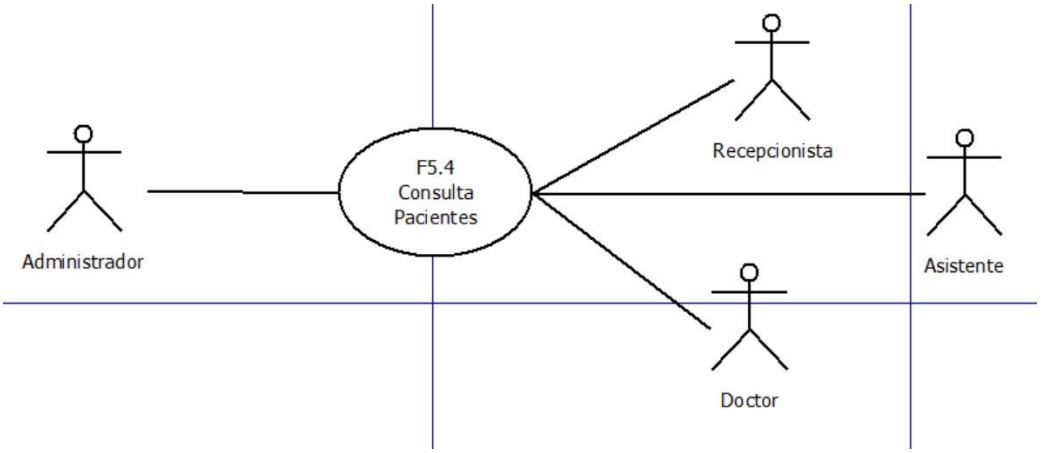
**Flujo Alterno:**

3. No existe el paciente → Ver Caso de Uso F1.1 para registrar el paciente.

**F1.4: Consultar Pacientes**

**Figura 38**

*Diagrama a Detalle de Consultar Pacientes*




*Nota*. Diagrama a Detalle de Consultar Pacientes que puede el Administrador, la Recepcionista y el Doctor.

**Descripción:** El administrador, la recepcionista y el doctor pueden consultar la información de los pacientes registrados.

**Actores:** Administrador, Recepcionista, Doctor

El diagrama en la **Figura 38** muestra el proceso detallado para consultar pacientes. A continuación, se describe el flujo principal:

**Flujo Principal:**

1. El actor selecciona la opción "Gestión de Pacientes" del menú principal.
2. El sistema presenta la ventana de pacientes.
3. El actor busca un paciente existente.
4. El sistema muestra los datos del paciente (E1).

En la **Tabla 21** se muestran las excepciones en el proceso de consultar pacientes, incluyendo sus causas y soluciones.

**Excepciones:**

**Tabla 21**

*Excepciones en el Proceso de Consultar Pacientes*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El paciente no existe | Ir al caso de uso F1.1 |

*Nota*. Tabla de excepciones al Consultar un Paciente con su causa y solución.

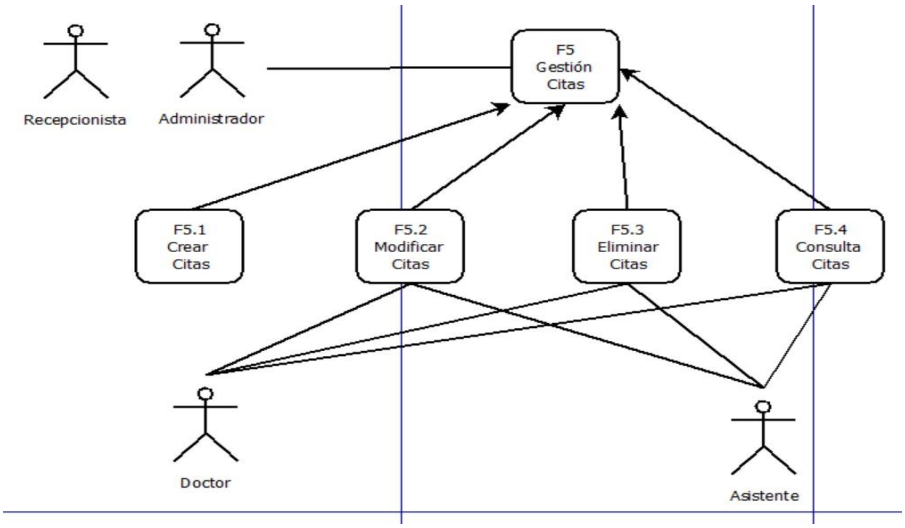
**Flujo Alterno:**

5. No existe el paciente → Ver Caso de Uso F1.1 para registrar el paciente.

**Caso de Uso: Gestión de Citas Médicas**

**Figura 39**

*Diagrama de Gestión de Citas Médicas*



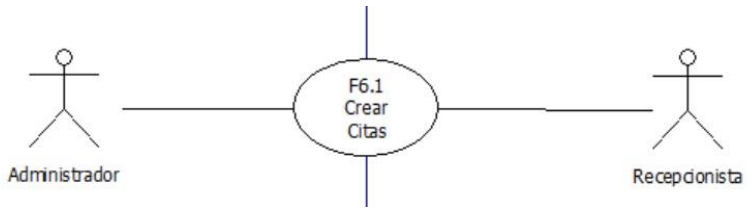

*Nota*. Diagrama detallado sobre la función de Gestión de Citas Médicas.

El diagrama en la **Figura 39** ilustra el caso de uso para la gestión de citas médicas. Este diagrama detalla cómo se programan, modifican y cancelan las citas médicas en el sistema, con los actores principales siendo el Administrador y la Recepcionista.

**F2.1: Crear Citas Médicas**

**Figura 40**

*Diagrama a Detalle de Crear Citas Médicas*




*Nota*. Diagrama a Detalle de Crear una Cita Médica que puede el Administrador o la Recepcionista.

**Descripción:** El administrador o la recepcionista pueden programar citas médicas para los pacientes.

**Actores:** Administrador, Recepcionista

El diagrama en la **Figura 40** muestra el proceso detallado para programar citas médicas. A continuación, se describe el flujo principal:

**Flujo Principal:**

1. El actor selecciona la opción "Gestión de Citas" del menú principal.
2. El sistema presenta la ventana de citas.
3. El actor ingresa los datos de la cita.
4. El sistema verifica la disponibilidad del doctor (E1).
5. El actor selecciona el horario disponible.
6. El actor presiona "Guardar".
7. El sistema almacena los datos de la nueva cita (E2).

En la **Tabla 22** se muestran las excepciones en el proceso de programar citas médicas, incluyendo sus causas y soluciones.

**Excepciones:**

**Tabla 22**

*Excepciones en el Proceso de Programar Citas Médicas*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | El doctor no está disponible | Seleccionar otro horario o doctor |
| E2 | No se almacenan los datos | Consultar con el administrador |

*Nota*. Tabla de excepciones al Programar una Cita Médica con sus causas y soluciones.

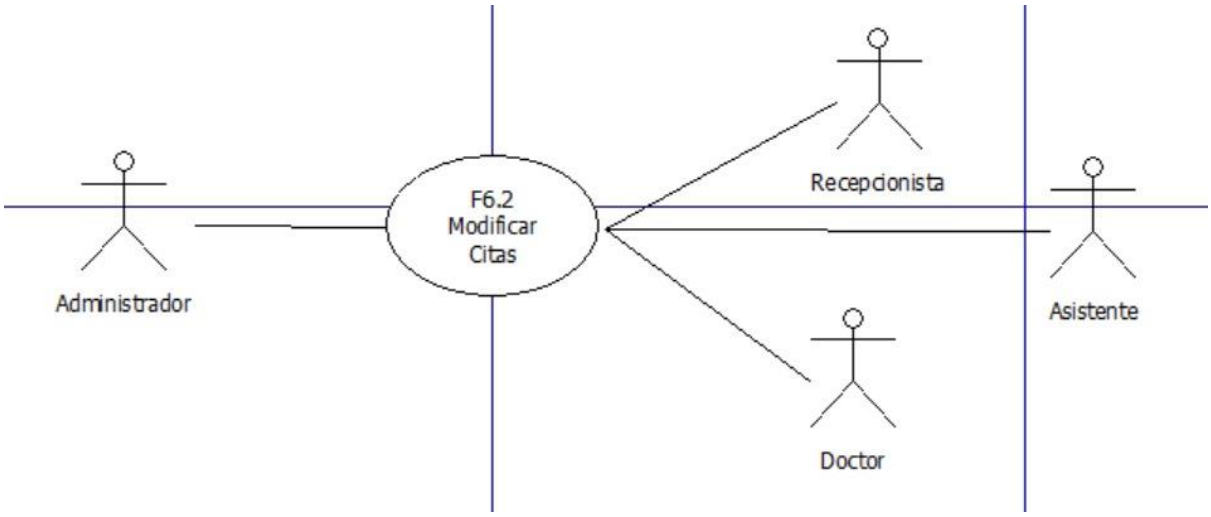
**Flujo Alterno:**

5. El doctor no está disponible → Seleccionar otro horario o doctor.

**F2.2: Modificar Citas Médicas**

**Figura 41**

*Diagrama a Detalle de Modificar Citas Médicas*




*Nota*. Diagrama a Detalle de Modificar una Cita Médica que puede el Administrador o la Recepcionista.

**Descripción:** El administrador o la recepcionista pueden modificar la información de las citas médicas existentes.

**Actores:** Administrador, Recepcionista

El diagrama en la **Figura 41** ilustra el proceso detallado para modificar citas médicas. A continuación, se describe el flujo principal:

**Flujo Principal:**

1. El actor selecciona la opción "Gestión de Citas" del menú principal.
2. El sistema presenta la ventana de citas.
3. El actor busca la cita que desea modificar (E1).
4. El sistema carga los datos de la cita.
5. El actor modifica los datos de la cita.
6. El actor presiona "Guardar".
7. El sistema almacena los datos modificados (E2).

En la **Tabla 23** se muestran las excepciones en el proceso de modificar citas médicas, incluyendo sus causas y soluciones.

**Excepciones:**

**Tabla 23**

*Excepciones en el Proceso de Modificar Citas Médicas*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | La cita no existe | Ir al caso de uso F2.1 |
| E2 | No se almacenan los datos | Consultar con el administrador |

*Nota*. Tabla de excepciones al Modificar una Cita Médica con sus causas y soluciones.

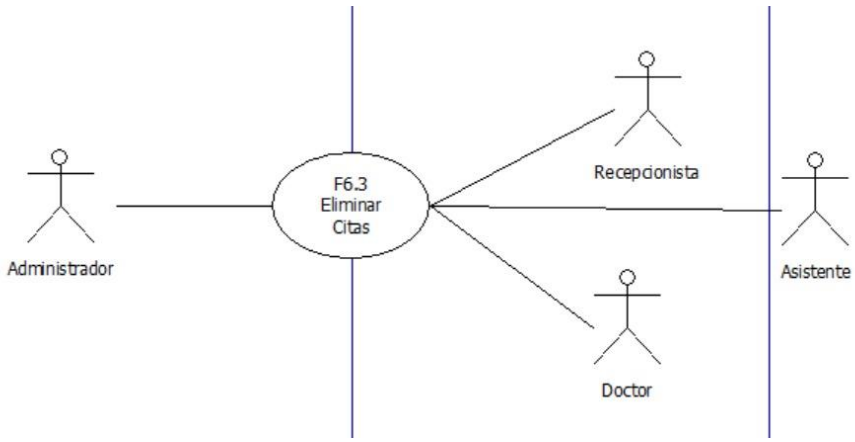
**Flujo Alterno:**

3. No existe la cita → Ver Caso de Uso F2.1 para programar la cita.

### F2.3: Eliminar Citas Médicas

**Figura 42**

*Diagrama a Detalle de Eliminar Citas Médicas*




*Nota*. Diagrama a Detalle de Eliminar una Cita Médica que puede el Administrador o la Recepcionista.

**Descripción:** El administrador o la recepcionista pueden cancelar citas médicas que ya no sean necesarias.

**Actores:** Administrador, Recepcionista

El diagrama en la **Figura 42** muestra el proceso detallado para cancelar citas médicas. A continuación, se describe el flujo principal:

**Flujo Principal:**

1. El actor selecciona la opción "Gestión de Citas" del menú principal.
2. El sistema presenta la ventana de citas.
3. El actor busca la cita que desea cancelar (E1).
4. El actor presiona "Eliminar".
5. El sistema solicita confirmación.
6. El actor confirma la cancelación.
7. El sistema cancela la cita (E2).

En la **Tabla 24** se muestran las excepciones en el proceso de cancelar citas médicas, incluyendo sus causas y soluciones.

**Excepciones:**

**Tabla 24**

*Excepciones en el Proceso de Cancelar Citas Médicas*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | La cita no existe | Ir al caso de uso F2.1 |
| E2 | No se cancelan los datos | Consultar con el administrador |

*Nota*. Tabla de excepciones al Cancelar una Cita Médica con sus causas y soluciones.

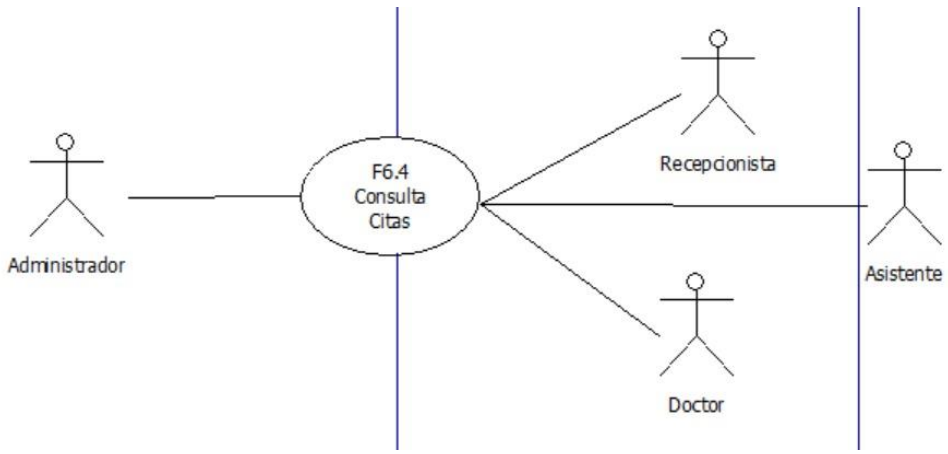
**Flujo Alterno:**

3. No existe la cita → Ver Caso de Uso F2.1 para programar la cita.

**F2.4: Consultar Citas Médicas**

**Figura 43**

*Diagrama a Detalle de Consultar Citas Médicas*




*Nota*. Diagrama a Detalle de Consultar Citas Médicas que puede el Administrador, la Recepcionista y el Doctor.

**Descripción:** El administrador, la recepcionista y el doctor pueden consultar la información de las citas médicas programadas.

**Actores:** Administrador, Recepcionista, Doctor

El diagrama en la **Figura 43** muestra el proceso detallado para consultar citas médicas. A continuación, se describe el flujo principal:

**Flujo Principal:**

1. El actor selecciona la opción "Gestión de Citas" del menú principal.
2. El sistema presenta la ventana de citas.
3. El actor busca una cita existente.
4. El sistema muestra los datos de la cita (E1).

En la **Tabla 25** se muestran las excepciones en el proceso de consultar citas médicas, incluyendo sus causas y soluciones.

**Excepciones:**

**Tabla 25**

*Excepciones en el Proceso de Consultar Citas Médicas*

| Excepciones | Causa | Solución |
|---|---|---|
| E1 | La cita no existe | Ir al caso de uso F2.1 |

*Nota*. Tabla de excepciones al Consultar una Cita Médica con su causa y solución.

**Flujo Alterno:**

4. No existe la cita → Ver Caso de Uso F2.1 para programar la cita.

El Sprint Backlog 2 establece una base sólida para la gestión de doctores, pacientes y citas médicas. Estas funcionalidades son esenciales para garantizar que el sistema cumpla con los requisitos operativos y de seguridad de la Clínica Dental ARPUL, proporcionando una administración eficiente y organizada de la información médica y la programación de citas.

## 4.2.2 Diseño 2

El diseño en el Sprint 1 se enfoca en el desarrollo del módulo de pacientes y la creación de un sistema integral para la programación y gestión de citas médicas. Este diseño es fundamental para asegurar que la clínica pueda manejar de manera eficiente y efectiva la información médica y la programación de citas, mejorando la experiencia tanto para el personal como para los pacientes.

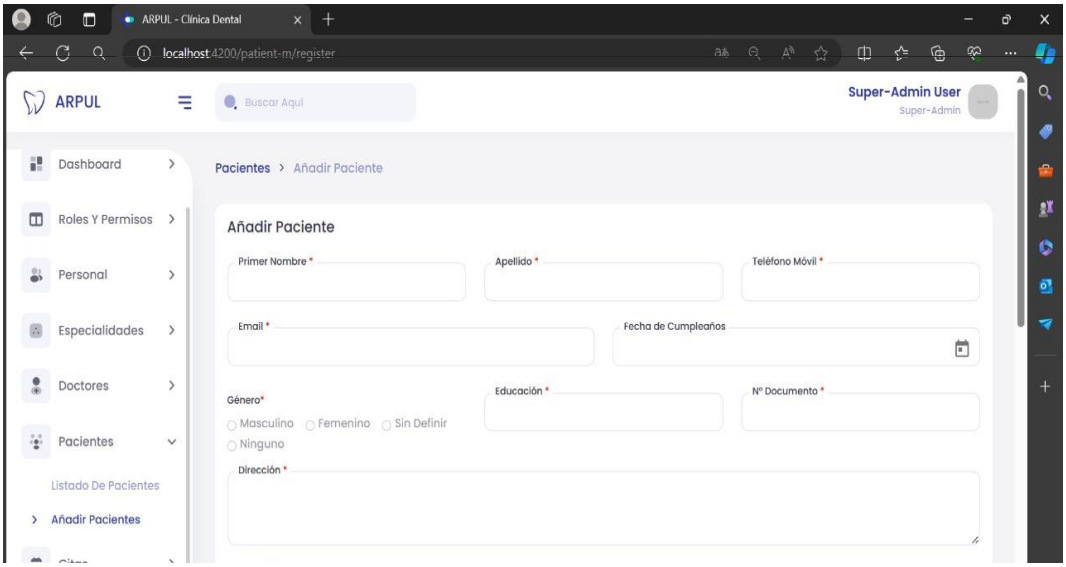
**Interfaces de Usuario**

Las interfaces de usuario están diseñadas para facilitar la interacción eficiente con el sistema, proporcionando accesibilidad y seguridad en la gestión de datos y operaciones del sistema. Se describen a continuación las interfaces específicas para cada funcionalidad relevante del Sprint 1:

1. **Gestión de Pacientes**
    - **Propósito:** Implementar un sistema para registrar y seguir la información de los pacientes, asegurando que todos los datos relevantes estén correctamente almacenados y accesibles para el personal autorizado.

**Figura 44**

*Pestaña de Registro de Pacientes*

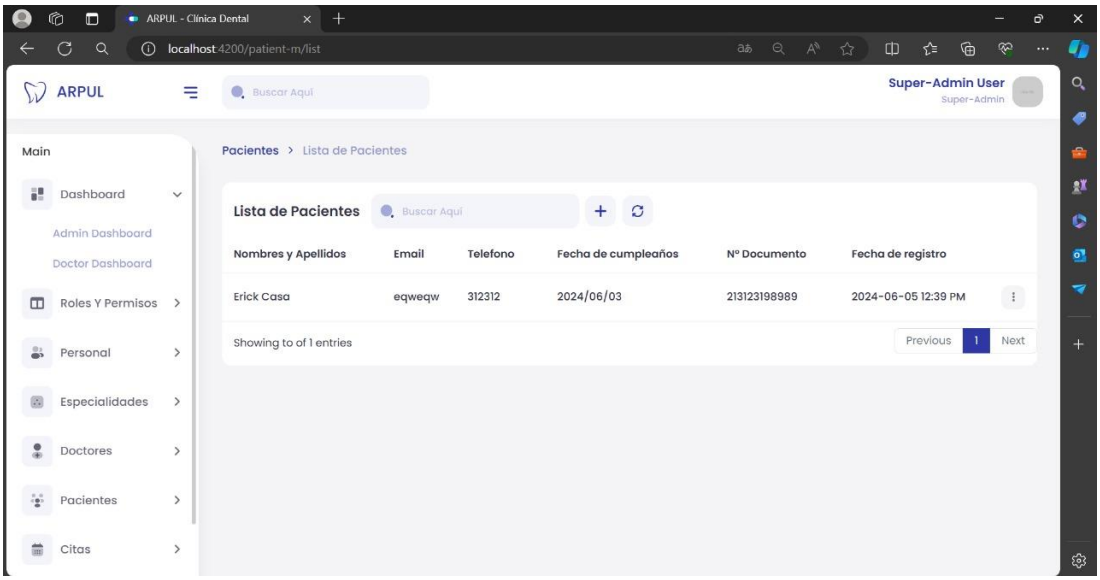



*Nota*. Pestaña que muestra cómo se registra la información de un paciente.

La **Figura 44** muestra la pestaña desarrollada para registrar la información de los pacientes. Esta pestaña permite al administrador o recepcionista ingresar datos personales y médicos relevantes de los pacientes.

**Figura 45**

*Pestaña del Listado de Pacientes*




*Nota*. Pestaña que muestra el listado de los pacientes registrados en el sistema.

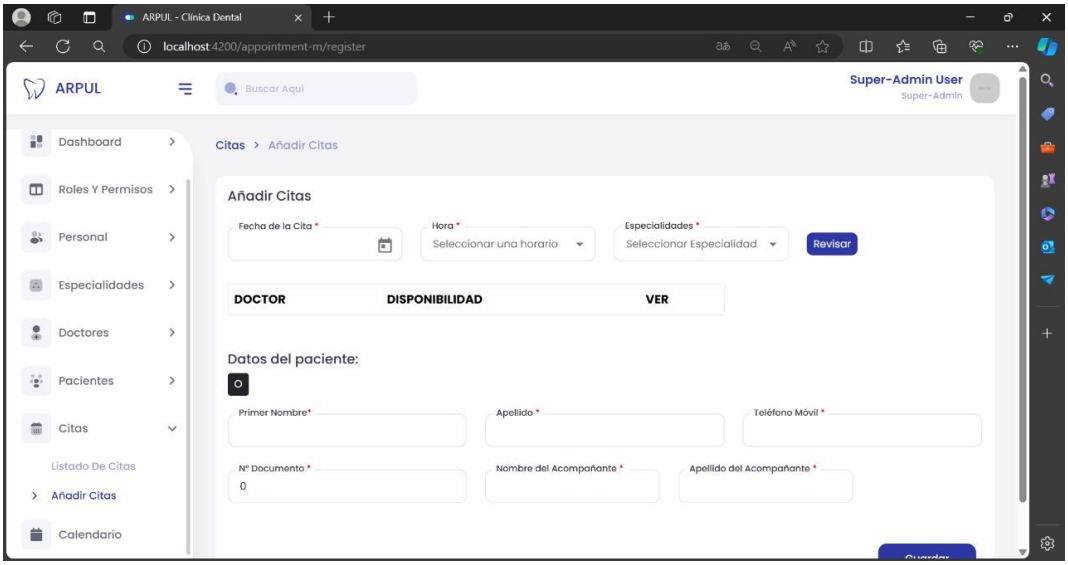
La **Figura 45** presenta la pestaña donde se muestra el listado de los pacientes registrados, permitiendo al administrador o recepcionista acceder y revisar la información de los pacientes.

2. **Gestión de Citas Médicas**

    - **Propósito:** Crear un sistema integral para la programación y gestión de citas médicas, facilitando la organización y planificación de las consultas médicas dentro de la clínica.

**Figura 46**

*Pestaña de Programación de Citas*

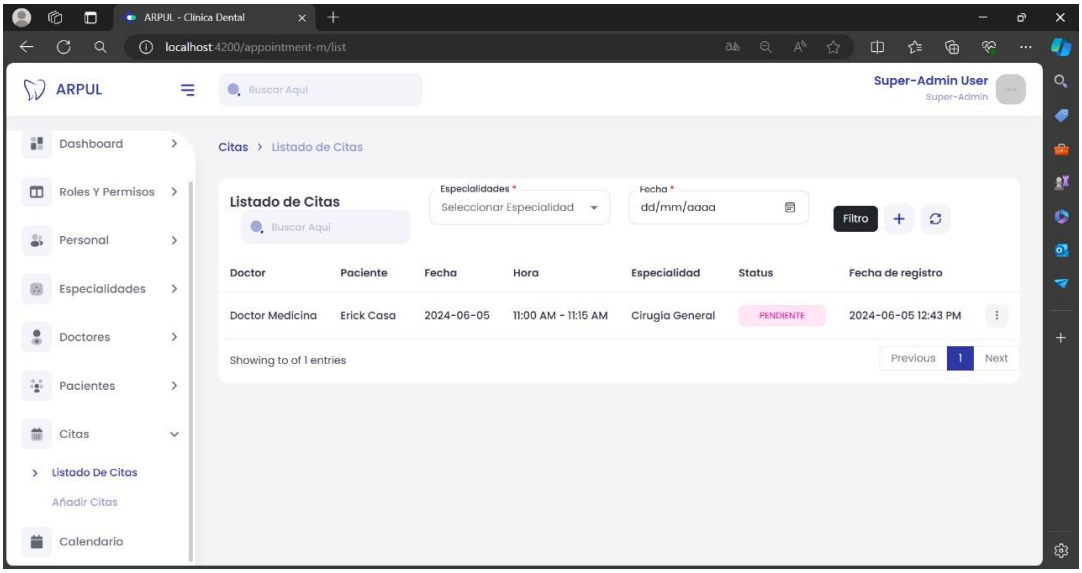



*Nota*. Pestaña que muestra cómo se programa una cita médica para un paciente.

La **Figura 46** muestra la pestaña desarrollada para la programación de citas médicas. Esta pestaña permite al administrador o recepcionista ingresar los detalles de la cita, seleccionar el doctor y confirmar la disponibilidad del horario.

**Figura 47**

*Pestaña del Listado de Citas de los Doctores*




*Nota*. Pestaña que muestra el Listado de citas de los doctores.

La **Figura 47** presenta la pestaña donde se muestra el listado de citas de los doctores, permitiendo al personal médico revisar y gestionar sus horarios de consulta.

Este diseño se centra en garantizar que el sistema sea intuitivo y fácil de usar, permitiendo al personal de la clínica manejar eficientemente la información de los pacientes y la programación de citas médicas. La implementación de estas interfaces asegura que todas las funcionalidades esenciales estén correctamente implementadas y probadas, facilitando una administración efectiva y segura dentro de la clínica dental.

## 4.2.2 Codificación 2

Durante el Sprint 2, nos centramos en la implementación de funcionalidades cruciales para la gestión de doctores, pacientes y citas médicas. A continuación, se destacan las principales herramientas, librerías y métodos utilizados en este sprint, junto con ejemplos de código que ilustran su implementación.

**1. Formularios Reactivos en Angular**

**Propósito:** Utilizados para crear y gestionar formularios dinámicos en el frontend, facilitando la entrada y validación de datos para pacientes y citas médicas.

**Implementación:** Los formularios reactivos permiten una gestión avanzada de la entrada de datos, incluyendo validaciones en tiempo real y estructuras de datos anidadas.

**Figura 48**

*Funcionalidad para el Registro*


*Nota*. La **Figura 48** muestra la implementación de formularios reactivos en Angular, utilizados para la entrada y validación de datos de pacientes.

**2. API RESTful en Laravel**

**Propósito:** Facilitar la comunicación entre el frontend y el backend, manejando operaciones CRUD para pacientes y citas médicas.

**Implementación:** Se crean endpoints RESTful que permiten crear, leer, actualizar y eliminar registros de pacientes y citas médicas.

**Figura 49**

*API RESTful para Gestión de Pacientes y Citas*

```
//
Route::get("patients/profile/{id}",[PatientController::class,"profile"]);
Route::post("patients/{id}",[PatientController::class,"update"]);
Route::resource("patients",PatientController::class);
//
Route::get("appointmet/config",[AppointmentController::class,"config"]);
Route::get("appointmet/patient",[AppointmentController::class,"query_patient"]);
Route::post("appointmet/filter",[AppointmentController::class,"filter"]);
Route::post("appointmet/calendar",[AppointmentController::class,"calendar"]);
Route::resource("appointmet",AppointmentController::class);
```

*Nota*. La **Figura 49** presenta el código para la API RESTful de gestión de pacientes y citas en Laravel, facilitando las operaciones CRUD.

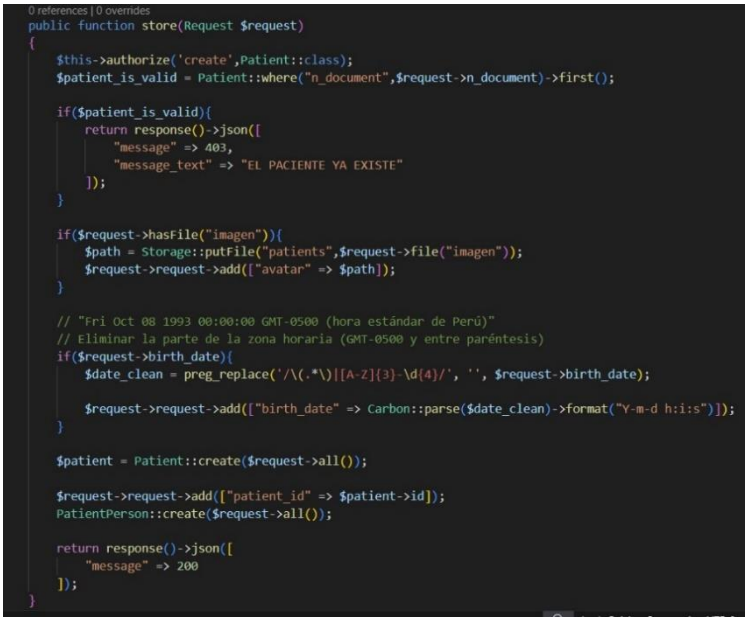
**3. Validaciones y Reglas de Negocio en Laravel**

**Propósito:** Garantizar que los datos ingresados en el sistema cumplan con los requisitos de formato y contenido, asegurando la integridad y consistencia de la información.

**Implementación:** Se definen reglas de validación en los controladores de Laravel para validar automáticamente los datos antes de guardarlos en la base de datos.

**Figura 50**

*Validaciones en Laravel*



```
0 references | 0 overrides
public function store(Request $request)
{
    $this->authorize('create',Patient::class);
    $patient_is_valid = Patient::where("n_document",$request->n_document)->first();

    if($patient_is_valid){
        return response()->json([
            "message" => 403,
            "message_text" => "EL PACIENTE YA EXISTE"
        ]);
    }

    if($request->hasFile("imagen")){
        $path = Storage::putFile("patients",$request->file("imagen"));
        $request->request->add(["avatar" => $path]);
    }

    // "Fri Oct 08 1993 00:00:00 GMT-0500 (hora estándar de Perú)"
    // Eliminar la parte de la zona horaria (GMT-0500 y entre paréntesis)
    if($request->birth_date){
        $date_clean = preg_replace('/\(.*\)|[A-Z]{3}-\d{4}/', '', $request->birth_date);

        $request->request->add(["birth_date" => Carbon::parse($date_clean)->format("Y-m-d h:i:s")]);
    }

    $patient = Patient::create($request->all());

    $request->request->add(["patient_id" => $patient->id]);
    PatientPerson::create($request->all());

    return response()->json([
        "message" => 200
    ]);
}
```

*Nota*. La **Figura 50** muestra las reglas de validación en Laravel, asegurando que los datos de los pacientes cumplan con los requisitos necesarios antes de ser almacenados.

Estas herramientas y métodos son fundamentales para el Sprint 2, enfocado en la gestión de pacientes y citas médicas. Cada componente ha sido elegido por su robustez y compatibilidad con el stack tecnológico utilizado, asegurando un desarrollo eficiente y seguro del sistema.

## 4.2.4 Pruebas 2

Al realizar las pruebas intensivas para validar tanto la funcionalidad como la usabilidad de las nuevas características implementadas en el sistema durante el Sprint 2, se asegura que el sistema no solo cumpla con las especificaciones técnicas y de negocio, sino que también brinde una experiencia de usuario óptima. La realización de pruebas de usabilidad y funcionalidad ayuda a detectar problemas tempranos en el desarrollo y a asegurar que las soluciones sean efectivas y accesibles para los usuarios finales.

En la **Figura 51** se puede obsevar las pruebas realizadas con la funcionalidad de los pacientes y de las citas médicas, las cuales de igual forma fueron revisadas y aprobadas por el propietario de la Clínica Dental.

**Figura 51**

*Pruebas del Sprint 2*

**Pruebas Sprint 2**

| ID | Prueba | Descripción | Tipo de Prueba | Resultado Esperado | Status |
|---|---|---|---|---|---|
| P1 | Registro de nuevo paciente | Crear un perfil de paciente con datos completos. | Funcionalidad | El paciente se registra sin errores. | ✓ |
| P2 | Edición de perfil de paciente | Modificar datos de un paciente existente. | Usabilidad | Los cambios se guardan correctamente. | ✓ |
| P3 | Eliminación de paciente | Eliminar un paciente inactivo. | Funcionalidad | El paciente se elimina correctamente del sistema. | ✓ |
| P4 | Programación de nueva cita médica | Crear una nueva cita médica con datos completos. | Funcionalidad | La cita se programa sin errores. | ✓ |
| P5 | Edición de cita médica | Modificar datos de una cita médica existente. | Usabilidad | Los cambios se guardan correctamente. | ✓ |
| P6 | Cancelación de cita médica | Cancelar una cita médica programada. | Funcionalidad | La cita se cancela correctamente en el sistema. | ✓ |

Erick Alexander Casa Ortiz
Desarrollador
CI: 1727389627

Dr. Paúl Esteban Casa Ortiz
Propietario Clínica Dental ARPUL
CI: 1721161089

*Nota*. Tabla de pruebas realizadas al sistema con la funcionalidad de pacientes y citas médicas con las respectivas firmas de los involucrados.

Estas pruebas validan que el sistema pueda gestionar adecuadamente las citas médicas, un aspecto crucial para la organización de los servicios médicos en la clínica. Es importante que las citas se puedan programar, modificar y cancelar sin complicaciones para adaptarse a las necesidades cambiantes del entorno médico.

## 4.3 Desarrollo Del Sprint 3

### 4.3.1 Sprint Backlog 3

El Sprint Backlog 3 se centra en la implementación de funcionalidades avanzadas para el calendario de citas médicas, la gestión de consultas médicas, la visualización de perfiles de doctores y pacientes, y la creación de dashboards para la administración de la clínica. Estas tareas son esenciales para mejorar la coordinación y eficiencia dentro de la clínica dental.

Para comenzar, se ha implementado un calendario interactivo que permite visualizar todas las citas médicas programadas, facilitando la organización de los horarios tanto para doctores como para pacientes. Además, se han desarrollado funcionalidades para gestionar la atención durante las consultas médicas, incluyendo el registro de diagnósticos y tratamientos.

Se han creado interfaces específicas para la visualización de perfiles detallados de doctores y pacientes, permitiendo un acceso rápido y fácil a la información relevante. También se han implementado dashboards que proporcionan una visualización clara de los KPIs y otros datos importantes para la administración de la clínica.

**Objetivos del Sprint 3**

**Objetivos Específicos:**

- **Calendario de Citas Médicas:** Implementar un calendario para visualizar todas las citas médicas programadas.
- **Atención de la Consulta Médica:** Desarrollar funcionalidades para gestionar la atención durante las consultas médicas, incluyendo el registro de diagnósticos y tratamientos.
- **Perfiles de Doctor y Pacientes:** Implementar interfaces para visualizar los perfiles detallados de doctores y pacientes.
- **Dashboards:** Crear dashboards para visualizar KPIs y otros datos relevantes para la administración de la clínica.

**Product Backlog del Sprint 3**

En la **Tabla 26** se presenta el Product Backlog del Sprint 3, destacando las historias de usuario, criterios de aceptación y prioridades.

**Tabla 26**

*Product Backlog del Sprint 3*

| ID | Historia de Usuario | Criterios de Aceptación | Prioridad |
|---|---|---|---|
| 1 | Como administrador, quiero visualizar todas las citas médicas programadas en un calendario interactivo. | Se debe implementar un calendario que muestre todas las citas médicas programadas de manera clara y accesible. | Alta |
| 2 | Como doctor, quiero registrar los diagnósticos y tratamientos durante las consultas médicas. | El sistema debe permitir a los doctores registrar diagnósticos y tratamientos en tiempo real durante las consultas. | Alta |
| 3 | Como administrador, quiero ver perfiles detallados de doctores y pacientes para gestionar mejor la clínica. | Las interfaces deben mostrar toda la información relevante de doctores y pacientes de manera organizada y accesible. | Alta |
| 4 | Como administrador, quiero tener dashboards que muestren los KPIs y datos relevantes de la clínica. | Los dashboards deben proporcionar visualizaciones claras de los KPIs y otros datos importantes para la administración de la clínica. | Alta |

*Nota*. Tabla sobre el Product Backlog del Sprint 3 con las prioridades.

Estas funcionalidades han sido priorizadas basándose en las necesidades operativas de la clínica, asegurando que las tareas más críticas se aborden primero. Tras revisar el Product Backlog, procederemos a explorar las funcionalidades y la implementación práctica de estas tareas, asegurando que el sistema cumpla con las necesidades operativas de la clínica.

### 4.3.2 Diseño 3

El diseño en el Sprint 3 se enfoca en la implementación de un calendario de citas médicas, la gestión de consultas médicas, la visualización de perfiles de doctores y pacientes, y la creación de dashboards. Este diseño es fundamental para mejorar la coordinación y eficiencia dentro de la clínica, proporcionando herramientas avanzadas para la gestión de la información y la toma de decisiones.

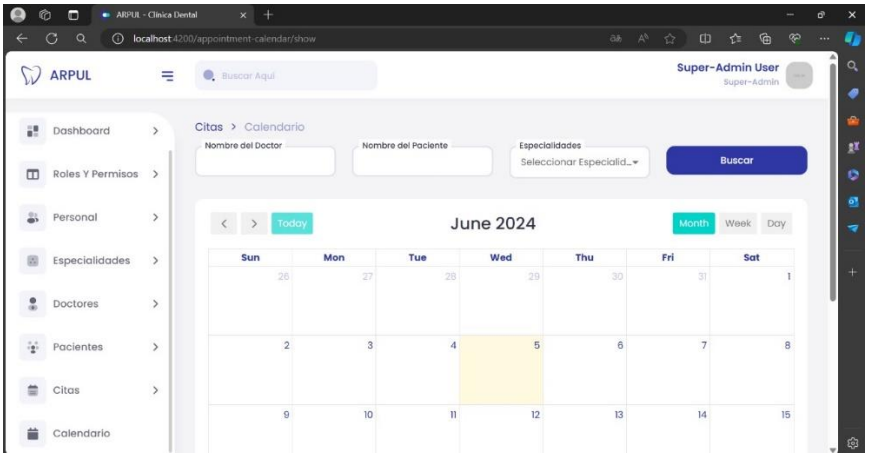
**Interfaces de Usuario**

Las interfaces de usuario están diseñadas para facilitar la interacción eficiente con el sistema, proporcionando accesibilidad y seguridad en la gestión de datos y operaciones del sistema. A continuación, se describen las interfaces específicas para cada funcionalidad relevante del Sprint 3:

1. **Calendario de Citas Médicas**
    - **Propósito:** Implementar un calendario para visualizar todas las citas médicas programadas, facilitando la organización y planificación de las consultas médicas dentro de la clínica.

**Figura 52**

*Pestaña del Calendario de Citas Médicas*




*Nota*. Pestaña que muestra el calendario interactivo con todas las citas médicas programadas.

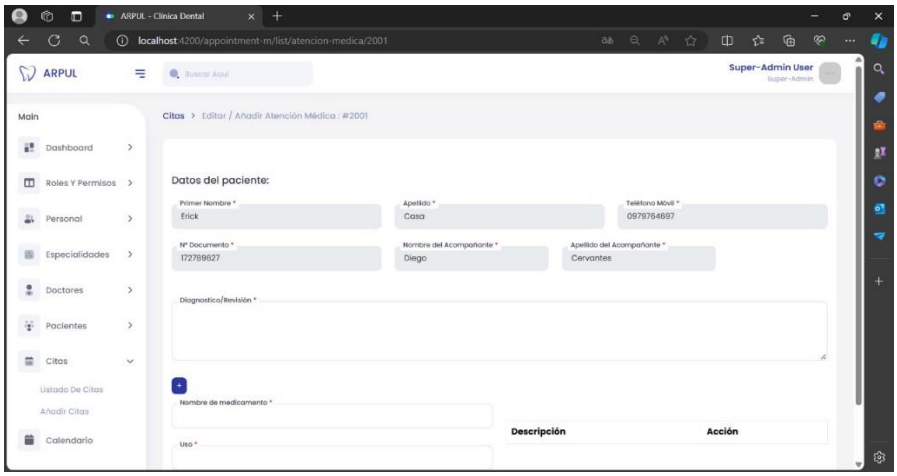
La **Figura 52** presenta la pestaña del calendario de citas médicas, que permite al personal de la clínica visualizar y gestionar todas las citas programadas de manera clara y accesible.

2. **Atención de la Consulta Médica**

    - **Propósito:** Desarrollar funcionalidades para gestionar la atención durante las consultas médicas, incluyendo el registro de diagnósticos y tratamientos.

**Figura 53**

*Pestaña de Atención de la Consulta Médica*




*Nota*. Pestaña que muestra cómo se registra la atención médica durante una consulta.

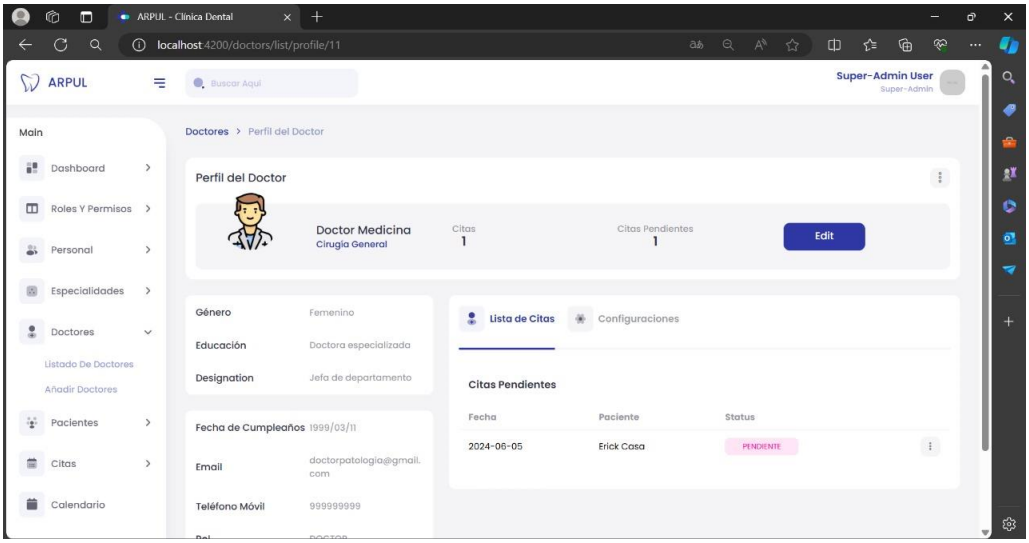
La **Figura 53** muestra la pestaña para la atención de la consulta médica, donde los doctores pueden registrar diagnósticos, tratamientos y otros detalles relevantes de cada consulta médica.

3. **Perfiles de Doctor y Pacientes**

    - **Propósito:** Implementar interfaces para visualizar los perfiles detallados de doctores y pacientes, facilitando el acceso a la información relevante.

**Figura 54**

*Pestaña de Perfil del Doctor*

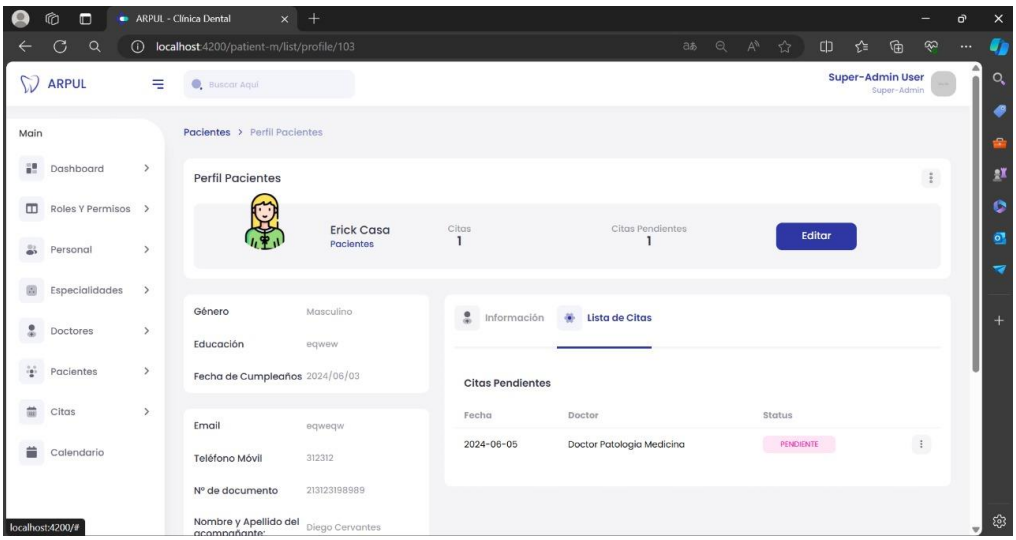



*Nota*. Pestaña que muestra el perfil detallado del doctor, incluyendo su especialidad y horarios.

La **Figura 54** presenta la pestaña del perfil del doctor, proporcionando una vista detallada de la información profesional y horarios del doctor.

**Figura 55**

*Pestaña de Perfil del Paciente*




*Nota*. Pestaña que muestra el perfil detallado del paciente, incluyendo su historial médico.

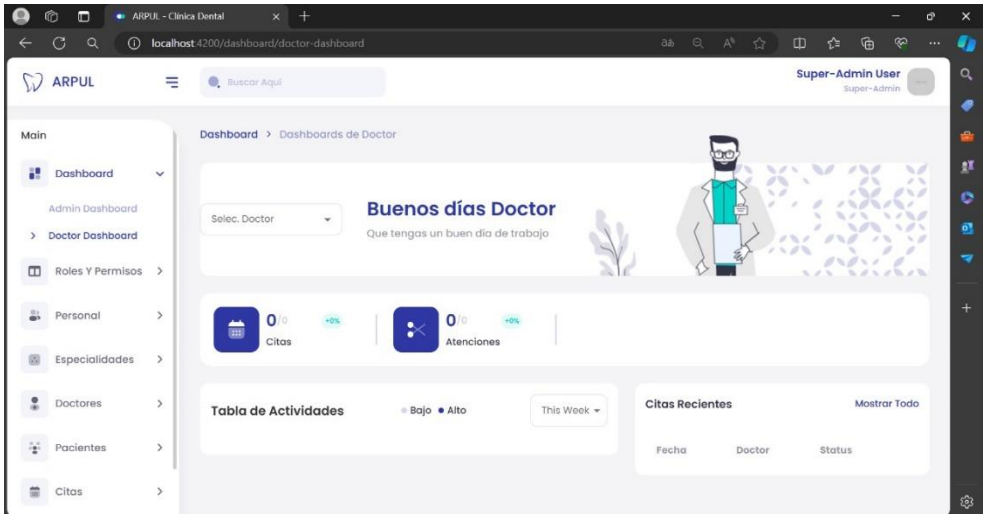
La **Figura 55** muestra la pestaña del perfil del paciente, donde se puede acceder a toda la información médica relevante y el historial de citas del paciente.

4. **Dashboards**

    - **Propósito:** Crear dashboards para visualizar KPIs y otros datos relevantes para la administración de la clínica, ayudando en la toma de decisiones y gestión operativa.

***Figura 56***

*Dashboard de la Clínica para el Administrador*

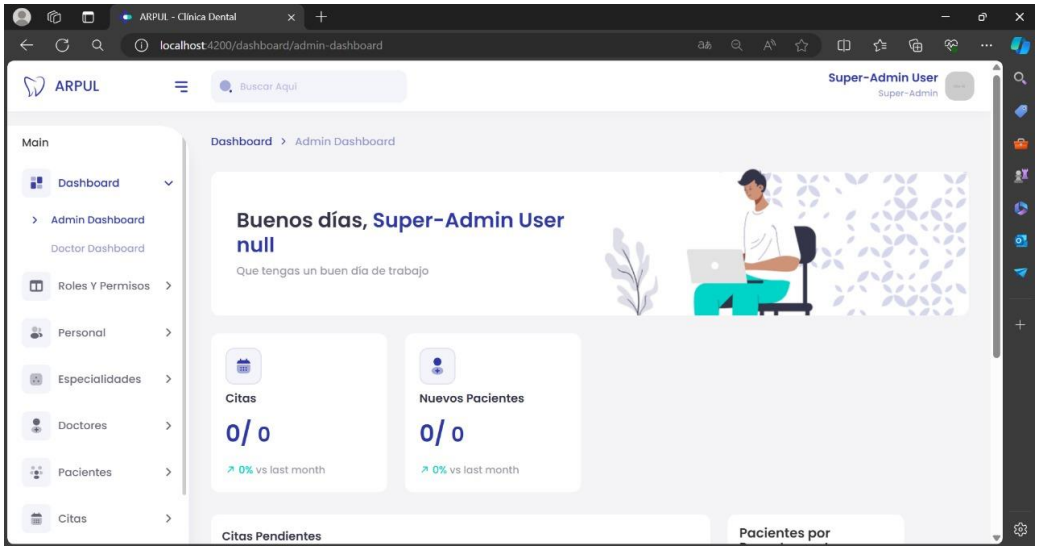



*Nota*. Dashboard que muestra los KPIs para el Administrador y datos relevantes para la administración de la clínica.

La **Figura 56** presenta el dashboard de la clínica, que proporciona una visualización clara de los KPIs, estadísticas y otros datos importantes para la administración y toma de decisiones de administradores.

**Figura 57**

*Dashboard de la Clínica para el Doctor*




*Nota*. Dashboard que muestra los KPIs para el Doctor y datos relevantes para la administración de la clínica.

La **Figura 57** presenta el dashboard de la clínica, que proporciona una visualización clara de los KPIs, estadísticas y otros datos importantes para la administración y toma de decisiones de doctores.

Este diseño se centra en que el sistema pueda ser intuitivo y fácil de usar, permitiendo al personal de la clínica manejar eficientemente la información de las citas médicas, consultas, y perfiles de doctores y pacientes. La implementación de estas interfaces asegura que todas las funcionalidades esenciales estén correctamente implementadas y probadas, facilitando una administración efectiva dentro de la clínica dental.

## 4.3.3 Codificación 3

Durante este Sprint, nos enfocamos en implementar el calendario de citas médicas, gestionar las consultas médicas y los perfiles de doctores y pacientes, además de desarrollar dashboards para visualizar KPIs importantes. A continuación se destacan algunas implementaciones clave:

1. **Redis Cache**

- **Propósito:** Utilizar Redis para mejorar la velocidad de carga y respuesta del sistema, especialmente para las funciones que requieren acceso frecuente a datos repetidos, como los perfiles de doctores y pacientes.
- **Implementación:** Se implementó Redis como una capa de caché en el backend, configurando la persistencia de datos necesaria para manejar los perfiles de usuarios. Los datos se almacenan temporalmente en Redis, lo que permite un acceso más rápido sin necesidad de consultar repetidamente la base de datos.

**Figura 58**

*Configuración de Redis*

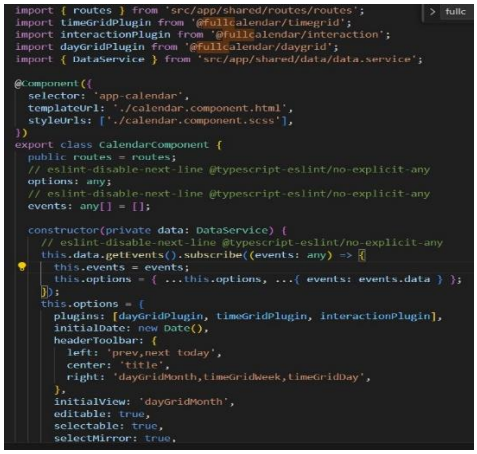

*Nota*. Configuración de Redis en el Perfil de Pacientes

La **Figura 58** muestra la configuración de Redis dentro del perfil de pacientes ya que se usan estos para poder mostrar la información y devolver los datos.

2. **Angular FullCalendar**
   - **Propósito:** Visualizar de manera eficiente todas las citas médicas programadas en un calendario interactivo.
   - **Implementación:** Se integró FullCalendar en Angular, aprovechando sus capacidades de personalización para mostrar eventos. Se configuraron vistas de mes, semana y día, y se habilitó la interacción para la edición y visualización detallada de cada cita.

**Figura 59**

*Configuración de la Función FullCalendar*



```
import { routes } from 'src/app/shared/routes/routes';
import timeGridPlugin from '@fullcalendar/timegrid';
import interactionPlugin from '@fullcalendar/interaction';
import dayGridPlugin from '@fullcalendar/daygrid';
import { DataService } from 'src/app/shared/data/data.service';

@Component({
  selector: 'app-calendar',
  templateUrl: './calendar.component.html',
  styleUrls: ['./calendar.component.scss'],
})
export class CalendarComponent {
  public routes = routes;
  // eslint-disable-next-line @typescript-eslint/no-explicit-any
  options: any;
  // eslint-disable-next-line @typescript-eslint/no-explicit-any
  events: any[] = [];

  constructor(private data: DataService) {
    // eslint-disable-next-line @typescript-eslint/no-explicit-any
    this.data.getEvents().subscribe((events: any) => {
      this.events = events;
      this.options = { ...this.options, ...{ events: events.data } };
    });
    this.options = {
      plugins: [dayGridPlugin, timeGridPlugin, interactionPlugin],
      initialDate: new Date(),
      headerToolbar: {
        left: 'prev,next today',
        center: 'title',
        right: 'dayGridMonth,timeGridWeek,timeGridDay',
      },
      initialView: 'dayGridMonth',
      editable: true,
      selectable: true,
      selectMirror: true,
```

*Nota*. Configuración de la funcionalidad de calendario con FullCalendar

La **Figura 59** muestra la configuración de la funcionalidad del calendario con la librería FullCalendar.

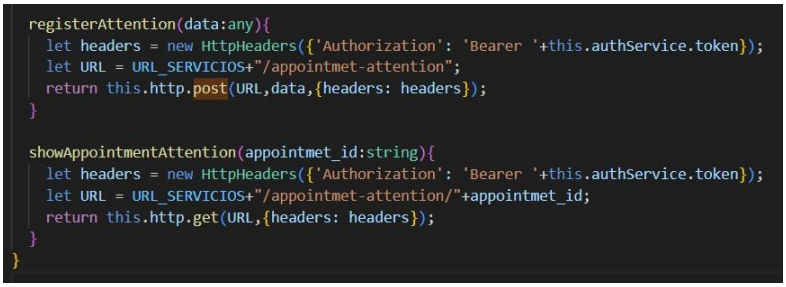
3. **APIs para Consultas Médicas**

   - **Propósito:** Gestionar la creación y actualización de registros de consultas médicas, incluyendo diagnósticos y tratamientos.

   - **Implementación:** Se desarrollaron endpoints específicos en Laravel para manejar el ciclo de vida de las consultas médicas. Esto incluyó la creación de registros y la actualización de diagnósticos utilizando métodos POST y PUT respectivamente.

**Figura 60**

*Métodos HTTP para Consultas Médicas*



```
  registerAttention(data:any){
    let headers = new HttpHeaders({'Authorization': 'Bearer '+this.authService.token});
    let URL = URL_SERVICIOS+"/appointmet-attention";
    return this.http.post(URL,data,{headers: headers});
  }

  showAppointmentAttention(appointmet_id:string){
    let headers = new HttpHeaders({'Authorization': 'Bearer '+this.authService.token});
    let URL = URL_SERVICIOS+"/appointmet-attention/"+appointmet_id;
    return this.http.get(URL,{headers: headers});
  }
}
```

*Nota*. Configuración de Métodos HTTP dentro de las Consultas Médicas

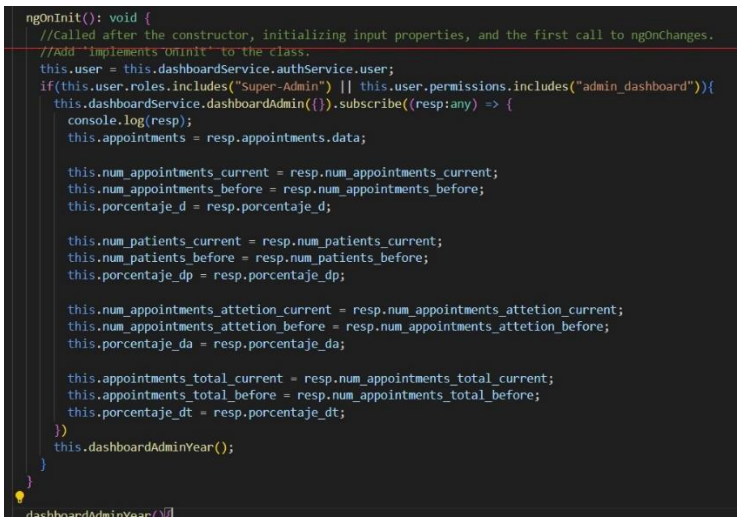
La **Figura 60** muestra la configuración HTTP que tiene en gran mayoría de las funciones en el sistema

4. **Generación de Dashboards**

   - **Propósito:** Crear dashboards para visualizar indicadores clave de rendimiento (KPIs) y otras métricas importantes para la administración de la clínica.
   - **Implementación:** Se utilizó Angular junto con bibliotecas de gráficos como Chart.js para construir dashboards dinámicos que reflejan en tiempo real el estado de varios aspectos operativos de la clínica.

**Figura 61**

*Configuración de la  funcionalidad del Dashboard de Admin*



```
ngOnInit(): void {
  //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
  //Add 'implements OnInit' to the class.
  this.user = this.dashboardService.authService.user;
  if(this.user.roles.includes("Super-Admin") || this.user.permissions.includes("admin_dashboard")){
    this.dashboardService.dashboardAdmin({}).subscribe((resp:any) => {
      console.log(resp);
      this.appointments = resp.appointments.data;

      this.num_appointments_current = resp.num_appointments_current;
      this.num_appointments_before = resp.num_appointments_before;
      this.porcentaje_d = resp.porcentaje_d;

      this.num_patients_current = resp.num_patients_current;
      this.num_patients_before = resp.num_patients_before;
      this.porcentaje_dp = resp.porcentaje_dp;

      this.num_appointments_attetion_current = resp.num_appointments_attetion_current;
      this.num_appointments_attetion_before = resp.num_appointments_attetion_before;
      this.porcentaje_da = resp.porcentaje_da;

      this.appointments_total_current = resp.num_appointments_total_current;
      this.appointments_total_before = resp.num_appointments_total_before;
      this.porcentaje_dt = resp.porcentaje_dt;
    })
    this.dashboardAdminYear();
  }
}
```

*Nota*. Configuración para la presentación del Dashboard de Admin con las citas y pacientes.

En la **Figura 61** muestra la configuración de un Dashboard el cual se relaciones con las citas y los pacientes que ayudaría para la toma de decisiones dentro de la clínica.

Estas herramientas y técnicas son esenciales para los objetivos del Sprint 3, facilitando la gestión eficiente del calendario de citas médicas, las consultas, y el seguimiento de métricas críticas a través de dashboards. Cada componente ha sido integrado cuidadosamente para asegurar la funcionalidad y mejorar la experiencia de los usuarios del sistema.

### 4.3.4 Pruebas 3

Las pruebas en este Sprint se realizaron para garantizar la correcta funcionalidad y usabilidad del calendario de citas médicas, la gestión de consultas médicas, los perfiles detallados de doctores y pacientes, y los dashboards para visualizar KPIs. El proceso de prueba ayudó a identificar y corregir problemas antes del despliegue, optimizando la experiencia del usuario y la eficacia operativa del sistema.

En la **Figura 62** se puede reflejar las pruebas realizadas con los últimos procesos del sistema que son el calendario de citas, atención de citas y la presentación de dashboards con la firma de revisión del propietario de la Clínica Dental.

**Figura 62**

*Pruebas del Sprint 3*

**Pruebas del Sprint 3**

| ID | Prueba | Descripción | Tipo de Prueba | Resultado Esperado | Status |
|---|---|---|---|---|---|
| P1 | Visualización del calendario | Verificar la correcta visualización del calendario. | Funcionalidad | El calendario muestra todas las citas programadas. | ✓ |
| P2 | Agendar nueva cita | Programar una nueva cita médica. | Funcionalidad | La cita se añade correctamente al calendario. | ✓ |
| P3 | Cancelación de cita | Cancelar una cita existente. | Usabilidad | La cita se elimina sin errores del calendario. | ✓ |
| P4 | Registro de diagnóstico | Registrar un diagnóstico para una consulta médica. | Funcionalidad | El diagnóstico se almacena correctamente en la base de datos. | ✓ |
| P5 | Modificación de tratamiento | Modificar un tratamiento existente. | Usabilidad | Los cambios en el tratamiento se guardan correctamente. | ✓ |
| P6 | Consulta de historial médico | Consultar el historial médico de un paciente. | Funcionalidad | Se muestra correctamente todo el historial médico. | ✓ |
| P7 | Visualización de perfil | Verificar la correcta visualización de perfiles. | Usabilidad | Los perfiles de doctores y pacientes se muestran correctamente. | ✓ |
| P8 | Edición de perfil de doctor | Editar el perfil de un doctor. | Funcionalidad | Los cambios en el perfil se actualizan correctamente. | ✓ |
| P9 | Búsqueda de paciente | Buscar perfiles de pacientes usando filtros. | Usabilidad | La búsqueda retorna los resultados correctos. | ✓ |
| P10 | Carga de Dashboard | Cargar los dashboards con KPIs. | Funcionalidad | Los dashboards se cargan con los datos actualizados. | ✓ |
| P11 | Interacción con Dashboard | Interactuar con elementos del dashboard. | Usabilidad | Las interacciones son fluidas y los datos cambian adecuadamente. | ✓ |
| P12 | Actualización de datos en tiempo real | Verificar la actualización dinámica de los KPIs. | Usabilidad | Los KPIs se actualizan en tiempo real sin errores. | ✓ |

Erick Alexander Casa Ortiz
Desarrollador
CI: 1727389627

Dr. Paúl Esteban Casa Ortiz
Propietario Clínica Dental ARPUL
CI: 1721161089

*Nota*. Tabla de pruebas realizadas en el Sprint 3 con las respectivas firmas de los involucrados.

Estas pruebas aseguran que cada componente del sistema funcione como se espera y que los usuarios puedan interactuar con el sistema de manera eficiente. La ejecución de estas pruebas es fundamental para validar la implementación de las nuevas funcionalidades y para garantizar que cumplen con los requerimientos establecidos inicialmente.

# CAPITULO V – CONCLUSIONES Y RECOMENDACIONES

## 5.1 Conclusiones

En cuanto a los requisitos de gestión de citas, se logró una detallada comprensión y documentación que sirvió como base para el desarrollo funcional del sistema. Esto incluyó especificaciones detalladas sobre cómo debían ser manejadas las citas, desde la programación hasta el seguimiento, garantizando que el sistema final estuviera alineado con las operaciones diarias de la clínica.

La interfaz de usuario desarrollada resultó ser intuitiva y amigable, facilitando su adopción por parte de todo el personal de la clínica, independientemente de su habilidad tecnológica. Este diseño centrado en el usuario mejoró significativamente la interacción con el sistema, se sugiere que, en el futuro, cuando se implante, la formación de usuarios con un nivel tecnológico específico sea más simple.

La elección de herramientas para el desarrollo del sistema fue clave para su buen funcionamiento. Utilizar tecnologías como Laravel y Angular facilitó la implementación de funcionalidades complejas y mejoró la eficiencia del desarrollo. Además, la integración de Redis y JWT ayudó a manejar eficazmente las sesiones de usuario y la seguridad de los datos, aspectos críticos en el ámbito clínico. Estas herramientas demostraron ser adecuadas para el proyecto.

Además, la implementación del sistema incluyó rigurosas pruebas para verificar que todas las funcionalidades cumplieran con los criterios establecidos inicialmente. Aunque el sistema aún no se encuentra en producción, estas pruebas fueron cruciales para validar la funcionalidad y la seguridad de la información, elementos clave dados los requisitos de privacidad y seguridad de datos de la clínica.

Respecto a la seguridad, el sistema se diseñó con un enfoque proactivo en la protección de datos, utilizando tecnologías modernas y prácticas recomendadas en la industria para asegurar la confidencialidad e integridad de la información del paciente. Aunque el sistema no se ha implementado en un entorno de producción, las pruebas realizadas indican que los protocolos de seguridad implementados son efectivos para prevenir accesos no autorizados.

## 5.2 Recomendaciones

Es recomendable continuar el desarrollo y refinamiento del sistema basándose en un enfoque iterativo y ágil. Aunque el sistema actual ya cumple con muchos de los requisitos establecidos, siempre hay espacio para la mejora, especialmente a medida que la clínica evoluciona y sus necesidades cambian.

En relación con las herramientas tecnológicas empleadas, se sugiere mantener una evaluación continua de éstas, para asegurarse de que siguen siendo las más adecuadas para los desafíos futuros. La tecnología cambia rápidamente, y el sistema debe ser capaz de adaptarse a nuevas herramientas y técnicas que puedan mejorar su eficacia y seguridad.

Una parte fundamental de la adopción de cualquier nuevo sistema tecnológico es la capacitación de los usuarios. Se recomienda establecer un programa de formación continuo para el personal de la clínica, asegurando que todos los usuarios entiendan cómo utilizar el sistema de manera efectiva. Esto maximizará la eficiencia del sistema y ayudará a evitar errores operativos.

Para los futuros desarrollos, sería beneficioso integrar un sistema de análisis de datos que permita a la clínica obtener insights operativos y clínicos a partir de los datos recogidos. Estos análisis podrían ayudar a la administración a tomar decisiones basadas en datos, optimizando aún más los recursos y mejorando la calidad del servicio ofrecido.

Es también recomendable considerar la expansión de la funcionalidad del sistema para incluir integraciones con otras plataformas, como sistemas de facturación o registros médicos electrónicos, lo que puede proporcionar una visión más holística del paciente y simplificar aún más las operaciones administrativas.

Sería óptimo mantener una comunicación abierta y continua con todos los usuarios del sistema para recoger sus comentarios y experiencias de uso. Establecer un canal formal de feedback no solo permitirá identificar áreas de mejora, sino también fomentar un sentido de propiedad y aceptación del sistema entre el personal, lo que es fundamental para su éxito a largo plazo.

Finalmente, para garantizar la centralización y la seguridad de la información, es aconsejable desarrollar una política clara de acceso a datos y seguridad informática. Esto incluiría protocolos detallados para el manejo de la información del paciente y medidas de seguridad para proteger contra el acceso no autorizado o la pérdida de datos.

# 5.3 Anexos

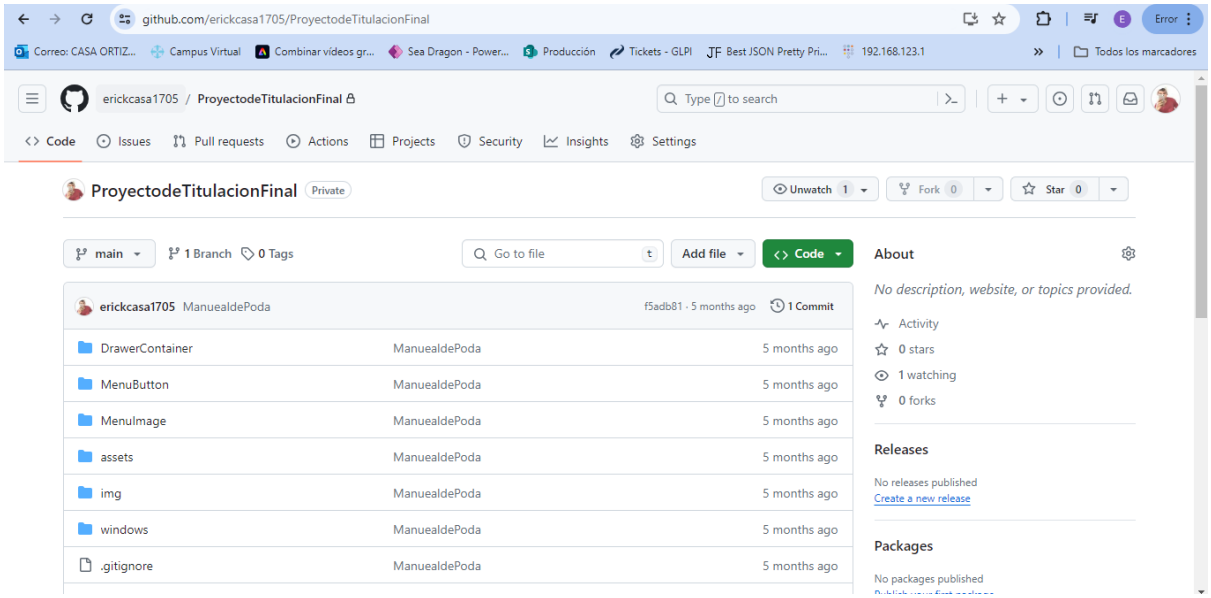
## Anexo A

*Cronograma de Trabajo*

| Actividad/Semana | Mar 1 | Mar 2 | Mar 3 | Mar 4 | Abr 1 | Abr 2 | Abr 3 | Abr 4 | Abr 5 | May 1 | May 2 | May 3 | May 4 | May 5 | Jun 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Capítulo I: Introducción | | | | | | | | | | | | | | | |
| Capítulo II: Marco Teórico | | | | | | | | | | | | | | | |
| Capítulo III: Planificación | | | | | | | | | | | | | | | |
| Capítulo IV: Desarrollo (Sprint 0) | | | | | | | | | | | | | | | |
| Capítulo IV: Desarrollo (Sprint 1) | | | | | | | | | | | | | | | |
| Capítulo IV: Desarrollo (Sprint 2) | | | | | | | | | | | | | | | |
| Capítulo V: Conclusiones | | | | | | | | | | | | | | | |
| Correcciones Finales | | | | | | | | | | | | | | | |

## Anexo B

*Captura de Código subido a GitHub*



**Anexo C**

*Documentación Entrevistas*

**Documentación Entrevistas – Clínica Dental ARPUL**

**Entrevistado:** Dr. Paúl Casa Ortiz

**Entrevistador:** Erick Casa

**Objetivo de la Entrevista:** Obtener información detallada sobre los requerimientos funcionales y no funcionales del sistema de gestión para la clínica ARPUL.

**Preguntas y Respuestas Clave:**

1. **¿Cómo le gustaría que se manejara la autenticación y seguridad en el sistema?**
    - **Respuesta:** Debe existir un mecanismo que permita a los usuarios validar su identidad con el login del sistema, para acceder a las funcionalidades del sistema que correspondan a su rol.
2. **¿Qué características deben tener los roles y permisos dentro del sistema?**
    - **Respuesta:** El sistema debe ofrecer la capacidad de asignar roles específicos a los usuarios, como doctores, enfermeros, recepcionistas y administradores, con funcionalidades accesibles según el rol asignado.
3. **¿Desea que los usuarios puedan modificar su información personal y ajustes de cuenta?**
    - **Respuesta:** Pensaría que no, esto que lo hagamos solo los administradores.
4. **¿Cómo debería ser el sistema de reserva de citas?**
    - **Respuesta:** Se requiere un sistema que permita al personal reservar citas, visualizando claramente las opciones de tiempo y disponibilidad de los doctores.
5. **¿Le gustaría tener un calendario de citas médicas integrado?**
    - **Respuesta:** Sí, un calendario visual que integre todas las citas médicas programadas, proporcionando una visión global de las actividades diarias en la clínica.

6. **¿Cómo debería manejarse el registro y seguimiento de pacientes?**
    - **Respuesta:** El sistema debe facilitar el registro de nuevos pacientes y el seguimiento continuo de su historial médico y tratamientos.
7. **¿Qué herramientas de reportes y análisis de datos son necesarias?**
    - **Respuesta:** Se necesitan herramientas que permitan visualizar reportes sobre la operación de la clínica.
8. **¿Qué importancia tiene la consistencia en la experiencia del usuario?**
    - **Respuesta:** El sistema debe ofrecer una experiencia de usuario consistente en términos de diseño, comportamiento de la interfaz.
9. **¿Con qué navegadores web debe ser compatible el sistema?**
    - **Respuesta:** El sistema debe ser compatible con los principales navegadores web utilizados en la clínica (por ejemplo, Chrome, Firefox, etc).
10. **¿Qué tan intuitivo debería ser el sistema para los usuarios?**
    - **Respuesta:** El sistema debe ser un poco intuitivo para personas que tienen cierta habilidad tecnológica, permitiendo a los usuarios realizar tareas comunes sin necesidad de formación o soporte técnico extensivo.
11. **¿Qué nivel de usabilidad debe tener la interfaz del sistema?**
    - **Respuesta:** La interfaz de usuario del sistema debe ser clara y fácil de navegar para personas con variados niveles de habilidad tecnológica.
12. **¿Cómo se deben gestionar las especialidades médicas dentro de la clínica?**
    - **Respuesta:** Debe haber un sistema para gestionar las especialidades médicas dentro de la clínica, asignando los doctores a sus respectivas especialidades.
13. **¿Cómo desea que se gestionen los usuarios en el sistema?**
    - **Respuesta:** No quiero usuarios para los pacientes. Prefiero que haya usuarios específicos para recepcionistas, asistentes y doctores, con la capacidad de gestionar las especialidades médicas, el sistema solo lo usamos nosotros.

Erick Alexander Casa Ortiz
Desarrollador
CI: 1727389627

Dr. Paúl Esteban Casa Ortiz
Propietario Clínica Dental ARPUL
CI: 1721161089

## Anexo D

*Funcionalidad Login*

```ts
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { AuthService } from 'src/app/shared/auth/auth.service';
import { routes } from 'src/app/shared/routes/routes';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss'],
})
export class LoginComponent implements OnInit {
  public routes = routes;
  public passwordClass = false;
  public ERROR = false;
  form = new FormGroup({
    email: new FormControl('josecode@gmail.com', [
      Validators.required,
      Validators.email,
    ]),
    password: new FormControl('12345678', [Validators.required]),
  });

  get f() {
    return this.form.controls;
  }

  constructor(public auth: AuthService,public router: Router) {}
  ngOnInit(): void {
    if (localStorage.getItem('authenticated')) {
      // localStorage.removeItem('authenticated');
      this.router.navigate([routes.adminDashboard]);
    }
  }

  loginFormSubmit() {
    if (this.form.valid) {
      this.ERROR = false;
      this.auth.login(this.form.value.email ? this.form.value.email : '' ,this.form.value.password ? this.form.value.password: '')
      .subscribe((resp:any) => {
        console.log(resp);
        if(resp){
          // EL LOGIN ES EXITOSO
          setTimeout(() => {
            document.location.reload();
          }, 50);
        }else{
          // EL LOGIN NO ES EXITOSO
```

```ts
  loginFormSubmit() {
    if (this.form.valid) {
      this.ERROR = false;
      this.auth.login(this.form.value.email ? this.form.value.email : '' ,this.form.value.password ? this.form.value.password: '')
      .subscribe((resp:any) => {
        console.log(resp);
        if(resp){
          // EL LOGIN ES EXITOSO
          setTimeout(() => {
            document.location.reload();
          }, 50);
        }else{
          // EL LOGIN NO ES EXITOSO
          this.ERROR = true;
        }
      },error => {
        console.log(error);
      })
      ;
    }
  }
  togglePassword() {
    this.passwordClass = !this.passwordClass;
  }
}
```

**Anexo E**

*Funcionalidad del Header*

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from 'src/app/shared/auth/auth.service';
import { routes } from 'src/app/shared/routes/routes';
import { SideBarService } from 'src/app/shared/side-bar/side-bar.service';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.scss'],
})
export class HeaderComponent {
  public routes = routes;
  public openBox = false;
  public miniSidebar  = false;
  public addClass = false;
  public user:any;

  constructor(public router: Router,private sideBar: SideBarService,public auth: AuthService) {
    this.sideBar.toggleSideBar.subscribe((res: string) => {
      if (res == 'true') {
        this.miniSidebar = true;
      } else {
        this.miniSidebar = false;
      }
    });
    let USER = localStorage.getItem("user");
    this.user = JSON.parse(USER ? USER : '');
    console.log(this.user);
  }
  getRole(){
    let RoleName = "";
    this.user.roles.forEach((rol:any) => {
      RoleName = rol;
    });
    return RoleName;
  }
  openBoxFunc() {
    this.openBox = !this.openBox;
    /* eslint no-var: off */
    var mainWrapper = document.getElementsByClassName('main-wrapper')[0];
    if (this.openBox) {
      mainWrapper.classList.add('open-msg-box');
    } else {
      mainWrapper.classList.remove('open-msg-box');
    }
```

```
  openBoxFunc() {
    this.openBox = !this.openBox;
    /* eslint no-var: off */
    var mainWrapper = document.getElementsByClassName('main-wrapper')[0];
    if (this.openBox) {
      mainWrapper.classList.add('open-msg-box');
    } else {
      mainWrapper.classList.remove('open-msg-box');
    }
  }
  logout(){
    this.auth.logout();
  }
  public toggleSideBar(): void {
    this.sideBar.switchSideMenuPosition();
  }
  public toggleMobileSideBar(): void {
    this.sideBar.switchMobileSideBarPosition();

      this.addClass = !this.addClass;
      /* eslint no-var: off */
      var root = document.getElementsByTagName( 'html' )[0];
      /* eslint no-var: off */
      // eslint-disable-next-line @typescript-eslint/no-explicit-any
      var sidebar:any = document.getElementById('sidebar')

      if (this.addClass) {
        root.classList.add('menu-opened');
        sidebar.classList.add('opened');
      }
      else {
        root.classList.remove('menu-opened');
        sidebar.classList.remove('opened');
      }
    }
  }
```

**Anexo F**

*Funcionalidad del SideBar*

admin_clinica > src > app > common-component > sidebar > TS sidebar.component.ts > ...

```ts
import { Component } from '@angular/core';
import { NavigationEnd, Router } from '@angular/router';
import { AuthService } from 'src/app/shared/auth/auth.service';
import { DataService } from 'src/app/shared/data/data.service';
import { MenuItem, SideBarData } from 'src/app/shared/models/models';
import { routes } from 'src/app/shared/routes/routes';
import { SideBarService } from 'src/app/shared/side-bar/side-bar.service';

@Component({
  selector: 'app-sidebar',
  templateUrl: './sidebar.component.html',
  styleUrls: ['./sidebar.component.scss'],
})
export class SidebarComponent {
  base = '';
  page = '';
  currentUrl = '';
  public classAdd = false;

  public multilevel: Array<boolean> = [false, false, false];

  public routes = routes;
  public sidebarData: Array<SideBarData> = [];
  public user:any;
  constructor(
    private data: DataService,
    private router: Router,
    private sideBar: SideBarService,
    public authService: AuthService,
  ) {
    // this.user = this.authService.user;
    let USER = localStorage.getItem("user");
    this.user = JSON.parse(USER ? USER : '');
    // INICIO
    if(this.user && this.user.roles.includes("Super-Admin")){
      this.sidebarData = this.data.sideBar;
    }else{
      // VAMOS A FILTRAR Y VALIDAR QUE OPCIONES PUEDE VER ESE ROL
      let permissions = this.user.permissions;
```

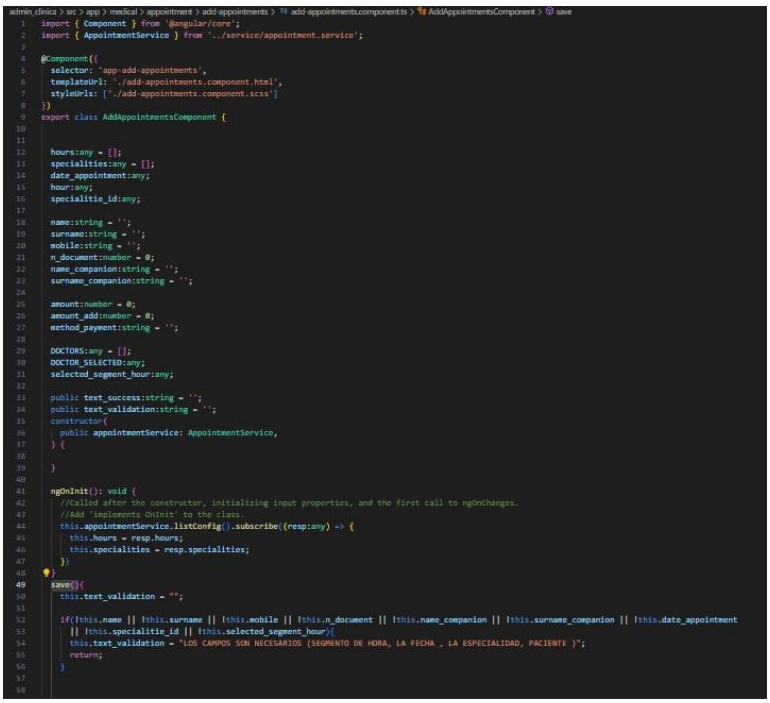
admin_clinica > src > app > common-component > sidebar > TS sidebar.component.ts > ...

```ts
export class SidebarComponent {
  constructor(
    let SIDE_BAR_G:any = [];
    this.data.sideBar.forEach((side:any) => {
      let SIDE_B:any = [];
      side.menu.forEach((menu_s:any) => {
        if(menu_s.subMenus.length > 0){
          let SUB_MENUS = menu_s.subMenus.filter((submenu:any) => permissions.includes(submenu.permision) && submenu.show_nav);
          if(SUB_MENUS.length > 0){
            menu_s.subMenus = SUB_MENUS;
            SIDE_B.push(menu_s);
          }
        }else{
          if(permissions.includes(menu_s.permision)){
            menu_s.subMenus = [];
            SIDE_B.push(menu_s);
          }
        }
      });
      if(SIDE_B.length > 0){
        side.menu = SIDE_B;
        SIDE_BAR_G.push(side);
      }
    })
    this.sidebarData = SIDE_BAR_G;
  }

  // FIN
  router.events.subscribe((event: object) => {
    if (event instanceof NavigationEnd) {
      this.getRoutes(event);
    }
  });
  this.getRoutes(this.router);
}

public expandSubMenus(menu: MenuItem): void {
  sessionStorage.setItem('menuValue', menu.menuValue);
  this.sidebarData.map((mainMenus: SideBarData) => {
    mainMenus.menu.map((resMenu: MenuItem) => {
      if (resMenu.menuValue == menu.menuValue) {
        menu.showSubRoute = !menu.showSubRoute;
      } else {
        resMenu.showSubRoute = false;
      }
    });
  });
}
```

```
  private getRoutes(route: { url: string }): void {
    const bodyTag = document.body;

    bodyTag.classList.remove('slide-nav')
    bodyTag.classList.remove('opened')
    this.currentUrl = route.url;

    const splitVal = route.url.split('/');


    this.base = splitVal[1];
    this.page = splitVal[2];
  }
  public miniSideBarMouseHover(position: string): void {
    if (position == 'over') {
      this.sideBar.expandSideBar.next("true");
    } else {
      this.sideBar.expandSideBar.next("false");
    }
  }

}
```

**Anexo G**

*Funcionalidad de Añadir una Cita*



```
admin_clinica > src > app > medical > appointment > add-appointments > TS add-appointments.component.ts > AddAppointmentsComponent > save
import { Component } from '@angular/core';
import { AppointmentService } from '../service/appointment.service';

@Component({
  selector: 'app-add-appointments',
  templateUrl: './add-appointments.component.html',
  styleUrls: ['./add-appointments.component.scss']
})
export class AddAppointmentsComponent {


  hours:any = [];
  specialities:any = [];
  date_appointment:any;
  hour:any;
  specialitie_id:any;

  name:string = '';
  surname:string = '';
  mobile:string = '';
  n_document:number = 0;
  name_companion:string = '';
  surname_companion:string = '';

  amount:number = 0;
  amount_add:number = 0;
  method_payment:string = '';

  DOCTORS:any = [];
  DOCTOR_SELECTED:any;
  selected_segment_hour:any;

  public text_success:string = '';
  public text_validation:string = '';
  constructor(
    public appointmentService: AppointmentService,
  ) {

  }

  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    this.appointmentService.listConfig().subscribe((resp:any) => {
      this.hours = resp.hours;
      this.specialities = resp.specialities;
    })
  }
  save(){
    this.text_validation = "";

    if(!this.name || !this.surname || !this.mobile || !this.n_document || !this.name_companion || !this.surname_companion || !this.date_appointment
      || !this.specialitie_id || !this.selected_segment_hour){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (SEGMENTO DE HORA, LA FECHA , LA ESPECIALIDAD, PACIENTE )";
      return;
    }
```

admin_clinica > src > app > medical > appointment > add-appointments > add-appointments.component.ts > AddA

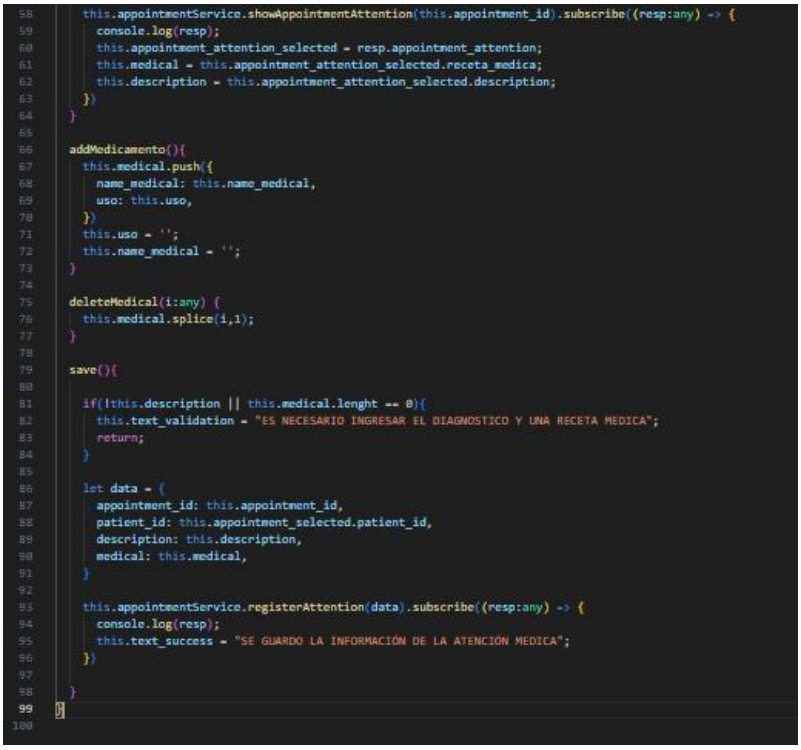
```
export class AddAppointmentsComponent {
  save(){

    let data = {
      "doctor_id": this.DOCTOR_SELECTED.doctor.id,
      // "patient_id",
      name: this.name,
      surname: this.surname,
      mobile: this.mobile,
      n_document: this.n_document,
      name_companion: this.name_companion,
      surname_companion: this.surname_companion,
      "date_appointment": this.date_appointment,
      "specialitie_id": this.specialitie_id,
      "doctor_schedule_join_hour_id": this.selected_segment_hour.id,
      amount: this.amount,
      amount_add: this.amount_add,
      method_payment: this.method_payment,
    }

    this.appointmentService.registerAppointment(data).subscribe((resp:any) => {
      console.log(resp);

      this.text_success = "LA CITA MEDICA SE REGISTRO CON EXITO";
    });
  }

  filtro(){
    let data = {
      date_appointment: this.date_appointment,
      hour: this.hour,
      specialitie_id : this.specialitie_id,
    }
    this.appointmentService.listFilter(data).subscribe((resp:any) => {
      console.log(resp);
      this.DOCTORS = resp.doctors;
    })
  }

  countDisponibilidad(DOCTOR:any){
    let SEGMENTS = [];
    SEGMENTS = DOCTOR.segments.filter((item:any) => !item.is_appointment);
    return SEGMENTS.length;
  }

  showSegment(DOCTOR:any){
    this.DOCTOR_SELECTED = DOCTOR;
  }

  selectSegment(SEGMENT:any){
    this.selected_segment_hour = SEGMENT;
  }
```

```
  filterPatient(){
    this.appointmentService.listPatient(this.n_document+"").subscribe((resp:any) => {
      console.log(resp);
      if(resp.message == 403){
        this.name = '';
        this.surname = '';
        this.mobile = ''
        this.n_document = 0;
      }else{
        this.name = resp.name;
        this.surname = resp.surname;
        this.mobile = resp.mobile;
        this.n_document = resp.n_document;
      }
    })
  }

  resetPatient(){
    this.name = '';
    this.surname = '';
    this.mobile = ''
    this.n_document = 0;
  }
}
```

**Anexo H**

*Funcionalidad de Atención Médica*

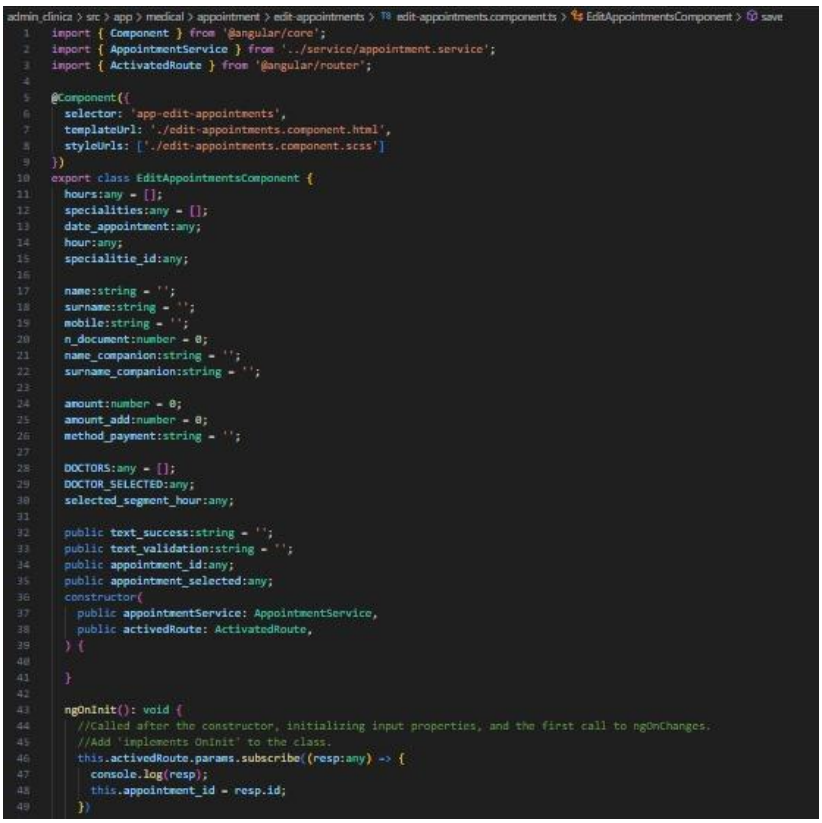
```
admin_clinica > src > app > medical > appointment > atencion-medical > TS atencion-medical.component.ts > ...
import { Component } from '@angular/core';
import { AppointmentService } from '../service/appointment.service';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-atencion-medical',
  templateUrl: './atencion-medical.component.html',
  styleUrls: ['./atencion-medical.component.scss']
})
export class AtencionMedicalComponent {

  name:string = '';
  surname:string = '';
  mobile:string = '';
  n_document:number = 0;
  name_companion:string = '';
  surname_companion:string = '';

  public text_success:string = '';
  public text_validation:string = '';

  public appointment_id:any;
  public appointment_selected:any;

  description:any;
  name_medical:any;
  uso:any;

  public medical:any = [];
  public appointment_attention_selected:any;
  constructor(
    public appointmentService: AppointmentService,
    public activedRoute: ActivatedRoute,
  ) {

  }

  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    this.activedRoute.params.subscribe((resp:any) => {
      console.log(resp);
      this.appointment_id = resp.id;
    })

    this.appointmentService.showAppointment(this.appointment_id).subscribe((resp:any) => {
      console.log(resp);
      this.appointment_selected = resp.appointment;
      // Datos del paciente
      this.name = this.appointment_selected.patient.name;
      this.surname = this.appointment_selected.patient.surname;
      this.mobile = this.appointment_selected.patient.mobile;
      this.n_document = this.appointment_selected.patient.n_document;
      this.name_companion = this.appointment_selected.patient.name_companion;
      this.surname_companion = this.appointment_selected.patient.surname_companion;
    })
```



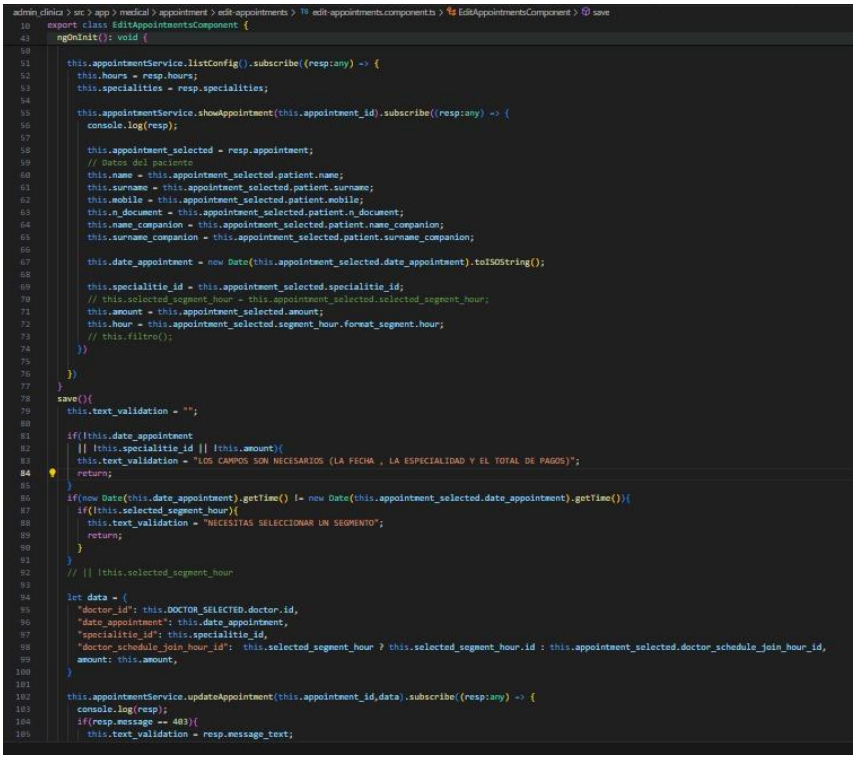
```
    this.appointmentService.showAppointmentAttention(this.appointment_id).subscribe((resp:any) => {
      console.log(resp);
      this.appointment_attention_selected = resp.appointment_attention;
      this.medical = this.appointment_attention_selected.receta_medica;
      this.description = this.appointment_attention_selected.description;
    })
  }

  addMedicamento(){
    this.medical.push({
      name_medical: this.name_medical,
      uso: this.uso,
    })
    this.uso = '';
    this.name_medical = '';
  }

  deleteMedical(i:any) {
    this.medical.splice(i,1);
  }

  save(){

    if(!this.description || this.medical.lenght == 0){
      this.text_validation = "ES NECESARIO INGRESAR EL DIAGNOSTICO Y UNA RECETA MEDICA";
      return;
    }

    let data = {
      appointment_id: this.appointment_id,
      patient_id: this.appointment_selected.patient_id,
      description: this.description,
      medical: this.medical,
    }

    this.appointmentService.registerAttention(data).subscribe((resp:any) => {
      console.log(resp);
      this.text_success = "SE GUARDO LA INFORMACIÓN DE LA ATENCIÓN MEDICA";
    })

  }
}
```

## Anexo I

*Funcionalidad de Editar una Cita*



```
admin_clinica > src > app > medical > appointment > edit-appointments > edit-appointments.component.ts > EditAppointmentsComponent > save
import { Component } from '@angular/core';
import { AppointmentService } from '../service/appointment.service';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-edit-appointments',
  templateUrl: './edit-appointments.component.html',
  styleUrls: ['./edit-appointments.component.scss']
})
export class EditAppointmentsComponent {
  hours:any = [];
  specialities:any = [];
  date_appointment:any;
  hour:any;
  specialitie_id:any;

  name:string = '';
  surname:string = '';
  mobile:string = '';
  n_document:number = 0;
  name_companion:string = '';
  surname_companion:string = '';

  amount:number = 0;
  amount_add:number = 0;
  method_payment:string = '';

  DOCTORS:any = [];
  DOCTOR_SELECTED:any;
  selected_segment_hour:any;

  public text_success:string = '';
  public text_validation:string = '';
  public appointment_id:any;
  public appointment_selected:any;
  constructor(
    public appointmentService: AppointmentService,
    public activedRoute: ActivatedRoute,
  ) {

  }

  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    this.activedRoute.params.subscribe((resp:any) => {
      console.log(resp);
      this.appointment_id = resp.id;
    })
```



```
admin_clinica > src > app > medical > appointment > edit-appointments > edit-appointments.component.ts > EditAppointmentsComponent > save
export class EditAppointmentsComponent {
  ngOnInit(): void {

    this.appointmentService.listConfig().subscribe((resp:any) => {
      this.hours = resp.hours;
      this.specialities = resp.specialities;

      this.appointmentService.showAppointment(this.appointment_id).subscribe((resp:any) => {
        console.log(resp);

        this.appointment_selected = resp.appointment;
        // Datos del paciente
        this.name = this.appointment_selected.patient.name;
        this.surname = this.appointment_selected.patient.surname;
        this.mobile = this.appointment_selected.patient.mobile;
        this.n_document = this.appointment_selected.patient.n_document;
        this.name_companion = this.appointment_selected.patient.name_companion;
        this.surname_companion = this.appointment_selected.patient.surname_companion;

        this.date_appointment = new Date(this.appointment_selected.date_appointment).toISOString();

        this.specialitie_id = this.appointment_selected.specialitie_id;
        // this.selected_segment_hour = this.appointment_selected.selected_segment_hour;
        this.amount = this.appointment_selected.amount;
        this.hour = this.appointment_selected.segment_hour.format_segment.hour;
        // this.filtro();
      })

    })
  }
  save(){
    this.text_validation = "";

    if(!this.date_appointment
      || !this.specialitie_id || !this.amount){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (LA FECHA , LA ESPECIALIDAD Y EL TOTAL DE PAGOS)";
      return;
    }
    if(new Date(this.date_appointment).getTime() != new Date(this.appointment_selected.date_appointment).getTime()){
      if(!this.selected_segment_hour){
        this.text_validation = "NECESITAS SELECCIONAR UN SEGMENTO";
        return;
      }
    }
    // || !this.selected_segment_hour

    let data = {
      "doctor_id": this.DOCTOR_SELECTED.doctor.id,
      "date_appointment": this.date_appointment,
      "specialitie_id": this.specialitie_id,
      "doctor_schedule_join_hour_id":  this.selected_segment_hour ? this.selected_segment_hour.id : this.appointment_selected.doctor_schedule_join_hour_id,
      amount: this.amount,
    }

    this.appointmentService.updateAppointment(this.appointment_id,data).subscribe((resp:any) => {
      console.log(resp);
      if(resp.message == 403){
        this.text_validation = resp.message_text;
```

admin_clinica > src > app > medical > appointment > edit-appointments > edit-appointments.component.ts > EditAppointmentsComponent > save

```
export class EditAppointmentsComponent {
  save(){

    this.appointmentService.updateAppointment(this.appointment_id,data).subscribe((resp:any) => {
      console.log(resp);
      if(resp.message == 403){
        this.text_validation = resp.message_text;
      }else{
        this.text_success = "LA CITA MEDICA SE EDITO CON EXITO";
      }
    });
  }

  onDateChange($event:any){
    this.DOCTORS = [];
    this.selected_segment_hour = null;
    this.DOCTOR_SELECTED = null;
  }
  filtro(){
    let data = {
      date_appointment: this.date_appointment,
      hour: this.hour,
      specialitie_id : this.specialitie_id,
    }
    this.appointmentService.listFilter(data).subscribe((resp:any) => {
      console.log(resp);
      this.DOCTORS = resp.doctors;

      // this.DOCTORS.forEach((doctor:any) => {
      //   if(doctor.doctor.id == this.appointment_selected.doctor_id){
      //     let INDEX = doctor.segments.findIndex((item:any) => item.id == this.appointment_selected.doctor_schedule_join_hour_id)
      //     if(INDEX != -1){
      //       this.showSegment(doctor);
      //     }
      //   }
      // });
    })
  }

  isDoctorSelected(DOCTOR:any){
    if(DOCTOR.doctor.id == this.appointment_selected.doctor_id){
      return true;
    }
    return false;
  }
  isSegmentSelected(SEGMENT:any){
    if(SEGMENT.id == this.appointment_selected.doctor_schedule_join_hour_id){
      return true;
    }
    return false;
  }
  countDisponibilidad(DOCTOR:any){
    let SEGMENTS = [];
    SEGMENTS = DOCTOR.segments.filter((item:any) => !item.is_appointment);
    return SEGMENTS.length;
  }
```

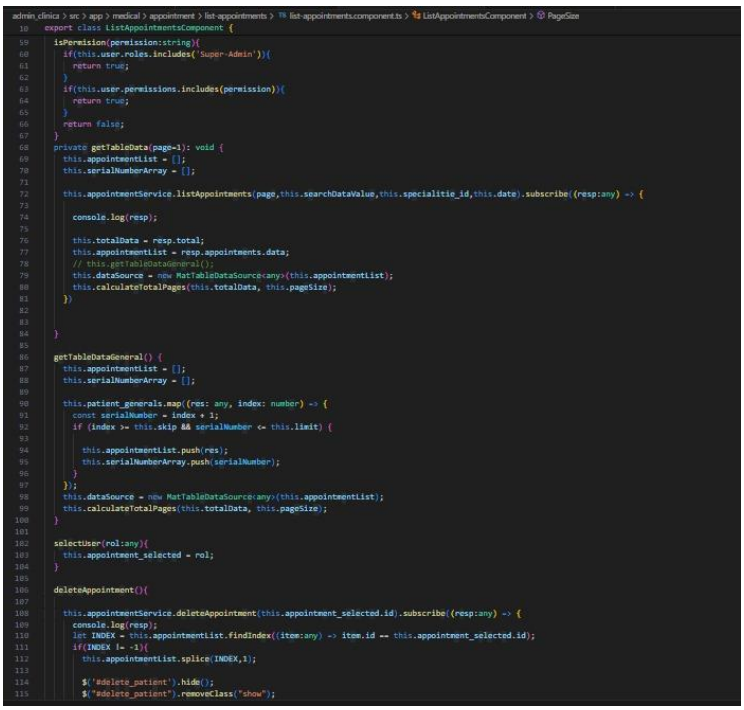
admin_clinica > src > app > medical > appointment > edit-appointments > edit-appointments.component.ts > EditAppointmentsComponent >

```
export class EditAppointmentsComponent {
  filtro(){
    this.appointmentService.listFilter(data).subscribe((resp:any) => {
      // });
    })
  }

  isDoctorSelected(DOCTOR:any){
    if(DOCTOR.doctor.id == this.appointment_selected.doctor_id){
      return true;
    }
    return false;
  }
  isSegmentSelected(SEGMENT:any){
    if(SEGMENT.id == this.appointment_selected.doctor_schedule_join_hour_id){
      return true;
    }
    return false;
  }
  countDisponibilidad(DOCTOR:any){
    let SEGMENTS = [];
    SEGMENTS = DOCTOR.segments.filter((item:any) => !item.is_appointment);
    return SEGMENTS.length;
  }

  showSegment(DOCTOR:any){
    this.DOCTOR_SELECTED = DOCTOR;
  }

  selectSegment(SEGMENT:any){
    this.selected_segment_hour = SEGMENT;
  }

  filterPatient(){
    this.appointmentService.listPatient(this.n_document+"").subscribe((resp:any) => {
      console.log(resp);
      if(resp.message == 403){
        this.name = '';
        this.surname = '';
        this.mobile = ''
        this.n_document = 0;
      }else{
        this.name = resp.name;
        this.surname = resp.surname;
        this.mobile = resp.mobile;
        this.n_document = resp.n_document;
      }
    })
  }

  resetPatient(){
    this.name = '';
    this.surname = '';
    this.mobile = ''
    this.n_document = 0;
  }
}
```

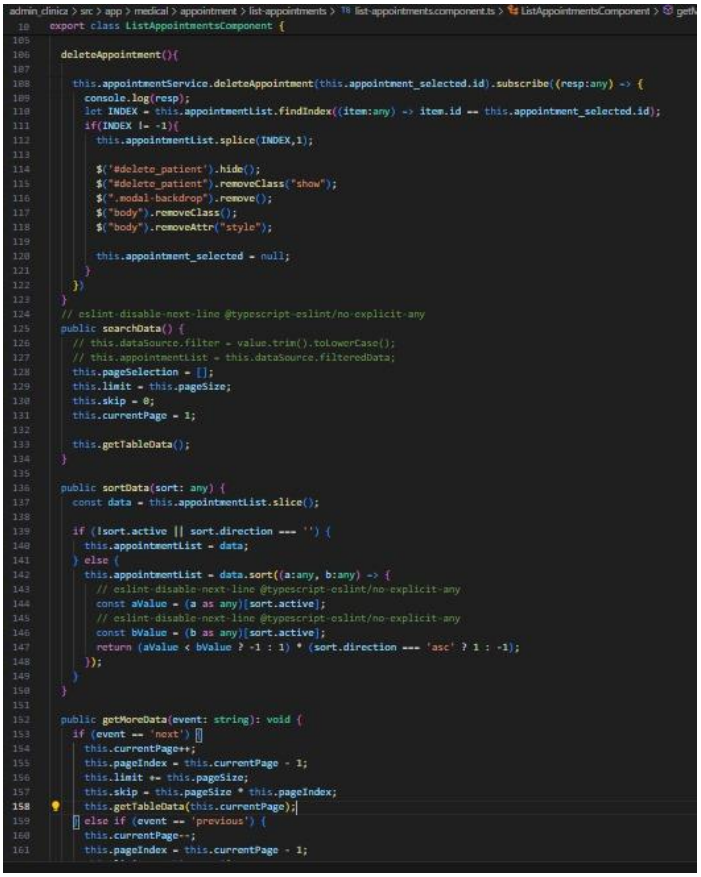
**Anexo J**

*Funcionalidad de Listado de Citas*

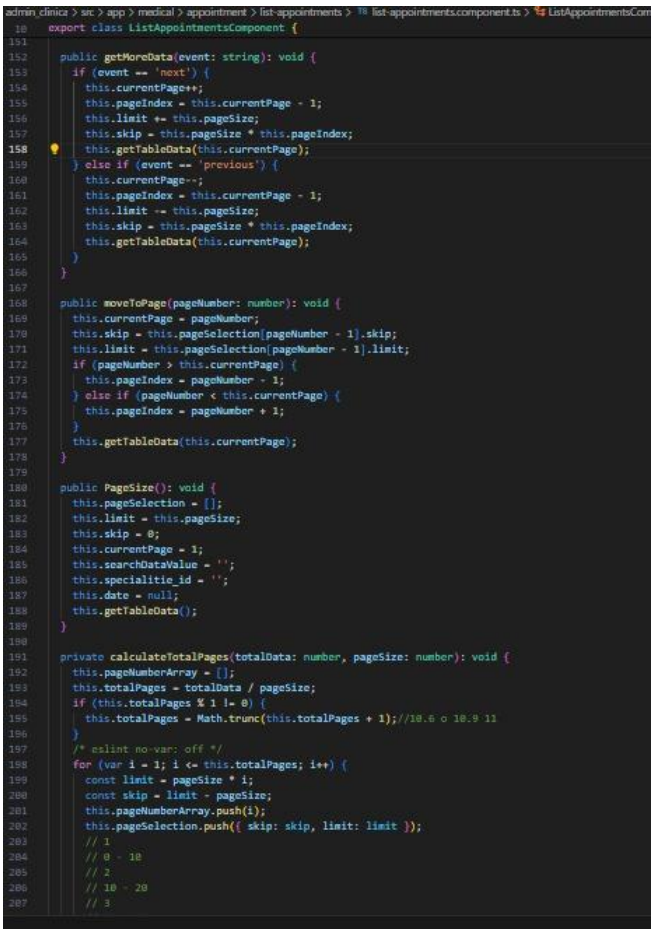
```
admin_clinica > src > app > medical > appointment > list-appointments > list-appointments.component.ts > ListAppointmentsCompone
import { Component } from '@angular/core';
import { AppointmentService } from '../service/appointment.service';
import { MatTableDataSource } from '@angular/material/table';

@Component({
  selector: 'app-list-appointments',
  templateUrl: './list-appointments.component.html',
  styleUrls: ['./list-appointments.component.scss']
})
export class ListAppointmentsComponent {
  public appointmentList:any = [];
  dataSource!: MatTableDataSource<any>;

  public showFilter = false;
  public searchDataValue = '';
  public specialitie_id = '';
  public date = null;
  public lastIndex = 0;
  public pageSize = 20;
  public totalData = 0;
  public skip = 0;//MIN
  public limit: number = this.pageSize;//MAX
  public pageIndex = 0;
  public serialNumberArray: Array<number> = [];
  public currentPage = 1;
  public pageNumberArray: Array<number> = [];
  public pageSelection: Array<any> = [];
  public totalPages = 0;

  public patient_generals:any = [];
  public appointment_selected:any;

  specialities:any = [];
  public user:any;
  constructor(
    public appointmentService: AppointmentService,
  ){

  }
  ngOnInit() {
    this.getTableData();

    this.appointmentService.listConfig().subscribe((resp:any) => {
      this.specialities = resp.specialities;
    })
    this.user = this.appointmentService.authService.user;
  }

  isPermited(){
    let band = false;
    this.user.roles.forEach((rol:any) => {
      if((rol).toUpperCase().indexOf("DOCTOR") != -1){
        band = true;
      }
    });
    return band;
  }

```



```
admin_clinica > src > app > medical > appointment > list-appointments > list-appointments.component.ts > ListAppointmentsComponent > PageSize
export class ListAppointmentsComponent {
  isPermision(permission:string){
    if(this.user.roles.includes('Super-Admin')){
      return true;
    }
    if(this.user.permissions.includes(permission)){
      return true;
    }
    return false;
  }
  private getTableData(page=1): void {
    this.appointmentList = [];
    this.serialNumberArray = [];

    this.appointmentService.listAppointments(page,this.searchDataValue,this.specialitie_id,this.date).subscribe((resp:any) => {

      console.log(resp);

      this.totalData = resp.total;
      this.appointmentList = resp.appointments.data;
      // this.getTableDataGeneral();
      this.dataSource = new MatTableDataSource<any>(this.appointmentList);
      this.calculateTotalPages(this.totalData, this.pageSize);
    })


  }

  getTableDataGeneral() {
    this.appointmentList = [];
    this.serialNumberArray = [];

    this.patient_generals.map((res: any, index: number) => {
      const serialNumber = index + 1;
      if (index >= this.skip && serialNumber <= this.limit) {

        this.appointmentList.push(res);
        this.serialNumberArray.push(serialNumber);
      }
    });
    this.dataSource = new MatTableDataSource<any>(this.appointmentList);
    this.calculateTotalPages(this.totalData, this.pageSize);
  }

  selectUser(rol:any){
    this.appointment_selected = rol;
  }

  deleteAppointment(){

    this.appointmentService.deleteAppointment(this.appointment_selected.id).subscribe((resp:any) => {
      console.log(resp);
      let INDEX = this.appointmentList.findIndex((item:any) => item.id == this.appointment_selected.id);
      if(INDEX != -1){
        this.appointmentList.splice(INDEX,1);

        $('#delete_patient').hide();
        $("#delete_patient").removeClass("show");
```



```
admin_clinica > src > app > medical > appointment > list-appointments > list-appointments.component.ts > ListAppointmentsComponent > getM
export class ListAppointmentsComponent {

  deleteAppointment(){

    this.appointmentService.deleteAppointment(this.appointment_selected.id).subscribe((resp:any) => {
      console.log(resp);
      let INDEX = this.appointmentList.findIndex((item:any) => item.id == this.appointment_selected.id);
      if(INDEX != -1){
        this.appointmentList.splice(INDEX,1);

        $('#delete_patient').hide();
        $("#delete_patient").removeClass("show");
        $(".modal-backdrop").remove();
        $("body").removeClass();
        $("body").removeAttr("style");

        this.appointment_selected = null;
      }
    })
  }
  // eslint-disable-next-line @typescript-eslint/no-explicit-any
  public searchData() {
    // this.dataSource.filter = value.trim().toLowerCase();
    // this.appointmentList = this.dataSource.filteredData;
    this.pageSelection = [];
    this.limit = this.pageSize;
    this.skip = 0;
    this.currentPage = 1;

    this.getTableData();
  }

  public sortData(sort: any) {
    const data = this.appointmentList.slice();

    if (!sort.active || sort.direction === '') {
      this.appointmentList = data;
    } else {
      this.appointmentList = data.sort((a:any, b:any) => {
        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        const aValue = (a as any)[sort.active];
        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        const bValue = (b as any)[sort.active];
        return (aValue < bValue ? -1 : 1) * (sort.direction === 'asc' ? 1 : -1);
      });
    }
  }

  public getMoreData(event: string): void {
    if (event == 'next') {
      this.currentPage++;
      this.pageIndex = this.currentPage - 1;
      this.limit += this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableData(this.currentPage);
    } else if (event == 'previous') {
      this.currentPage--;
      this.pageIndex = this.currentPage - 1;
```



```
admin_clinica > src > app > medical > appointment > list-appointments > list-appointments.component.ts > ListAppointmentsCom
export class ListAppointmentsComponent {

  public getMoreData(event: string): void {
    if (event == 'next') {
      this.currentPage++;
      this.pageIndex = this.currentPage - 1;
      this.limit += this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableData(this.currentPage);
    } else if (event == 'previous') {
      this.currentPage--;
      this.pageIndex = this.currentPage - 1;
      this.limit -= this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableData(this.currentPage);
    }
  }

  public moveToPage(pageNumber: number): void {
    this.currentPage = pageNumber;
    this.skip = this.pageSelection[pageNumber - 1].skip;
    this.limit = this.pageSelection[pageNumber - 1].limit;
    if (pageNumber > this.currentPage) {
      this.pageIndex = pageNumber - 1;
    } else if (pageNumber < this.currentPage) {
      this.pageIndex = pageNumber + 1;
    }
    this.getTableData(this.currentPage);
  }

  public PageSize(): void {
    this.pageSelection = [];
    this.limit = this.pageSize;
    this.skip = 0;
    this.currentPage = 1;
    this.searchDataValue = '';
    this.specialitie_id = '';
    this.date = null;
    this.getTableData();
  }

  private calculateTotalPages(totalData: number, pageSize: number): void {
    this.pageNumberArray = [];
    this.totalPages = totalData / pageSize;
    if (this.totalPages % 1 != 0) {
      this.totalPages = Math.trunc(this.totalPages + 1);//10.6 o 10.9 11
    }
    /* eslint no-var: off */
    for (var i = 1; i <= this.totalPages; i++) {
      const limit = pageSize * i;
      const skip = limit - pageSize;
      this.pageNumberArray.push(i);
      this.pageSelection.push({ skip: skip, limit: limit });
      // 1
      // 0 - 10
      // 2
      // 10 - 20
      // 3
```

**Anexo K**

*Funcionalidad del Calendario de Citas*

admin_clinica > src > app > medical > calendar-appointment > appointment-calendar > TS appointment-calendar.component.ts > ...

```ts
import { Component } from '@angular/core';
import timeGridPlugin from '@fullcalendar/timegrid';
import interactionPlugin from '@fullcalendar/interaction';
import dayGridPlugin from '@fullcalendar/daygrid';
import { AppointmentPayService } from '../../appointment-pay/service/appointment-pay.service';
import { CalendarAppointmentService } from '../service/calendar-appointment.service';

@Component({
  selector: 'app-appointment-calendar',
  templateUrl: './appointment-calendar.component.html',
  styleUrls: ['./appointment-calendar.component.scss']
})
export class AppointmentCalendarComponent {

  // eslint-disable-next-line @typescript-eslint/no-explicit-any
  options: any;
  // eslint-disable-next-line @typescript-eslint/no-explicit-any
  events: any[] = [];

  public specialities:any  = [];
  public specialitie_id = '';
  public search_doctor = '';
  public search_patient = '';

  public user:any;
  constructor(
    public appointmentPayService: AppointmentPayService,
    public appointmentCalendarService: CalendarAppointmentService,
  ) {
    // eslint-disable-next-line @typescript-eslint/no-explicit-any
    // this.data.getEvents().subscribe((events: any) => {
    //   this.events = events;
    //   this.options = { ...this.options, ...{ events: events.data } };
    // });
    this.options = {
      plugins: [dayGridPlugin, timeGridPlugin, interactionPlugin],
      initialDate: new Date(),
      headerToolbar: {
        left: 'prev,next today',
        center: 'title',
        right: 'dayGridMonth,timeGridWeek,timeGridDay',
      },
      initialView: 'dayGridMonth',
      editable: true,
      selectable: true,
      selectMirror: true,
      dayMaxEvents: true,
      events: [
        { title: 'Meeting', start: new Date() }
      ]
    };
  }
```

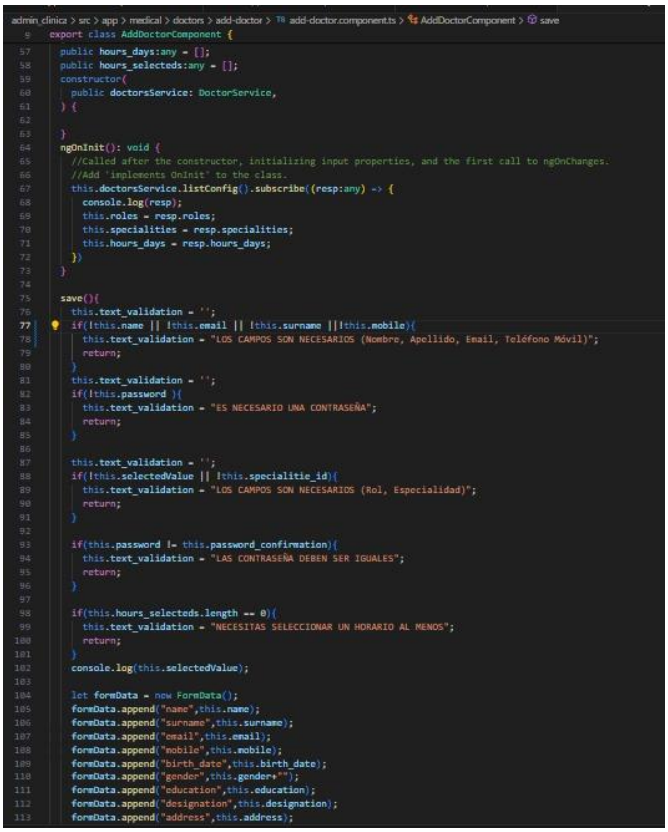
}

```ts
  isPermited(){
    let band = false;
    this.user.roles.forEach((rol:any) => {
      if((rol).toUpperCase().indexOf("DOCTOR") != -1){
        band = true;
      }
    });
    return band;
  }

  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    this.appointmentPayService.listConfig().subscribe((resp:any) => {
      this.specialities = resp.specialities;

    })
    this.user = this.appointmentPayService.authService.user;
    this.calendarAppointment();
  }
  calendarAppointment() {
    let data = {
      specialitie_id: this.specialitie_id,
      search_doctor: this.search_doctor,
      search_patient: this.search_patient,
    }
    this.appointmentCalendarService.calendarAppointment(data).subscribe((resp:any) => {
      console.log(resp);
      this.options = { ...this.options, ...{ events: resp.appointments } };
    })
  }

  searchData(){
    this.calendarAppointment();
  }
}
```

## Anexo L

*Funcionalidad de Añadir un Doctor*

```
admin_clinica > src > app > medical > doctors > add-doctor > TS add-doctor.component.ts > AddDoctorComponent > save
import { Component } from '@angular/core';
import { DoctorService } from '../service/doctor.service';

@Component({
  selector: 'app-add-doctor',
  templateUrl: './add-doctor.component.html',
  styleUrls: ['./add-doctor.component.scss']
})
export class AddDoctorComponent {
  public selectedValue !: string  ;
  public name:string = '';
  public surname:string = '';
  public mobile:string = '';
  public email:string = '';
  public password:string = '';
  public password_confirmation:string = '';

  public birth_date:string = '';
  public gender:number = 1;
  public education:string = '';
  public designation:string = '';
  public address:string = '';

  public roles:any = [];

  public FILE_AVATAR:any;
  public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';

  public specialitie_id:any;
  public specialities:any = [];

  public text_success:string = '';
  public text_validation:string = '';

  public days_week = [
    {
      day: 'Lunes',
      class: 'table-primary'
    },
    {
      day: 'Martes',
      class: 'table-secondary'
    },
    {
      day: 'Miercoles',
      class: 'table-success'
    },
    {
      day: 'Jueves',
      class: 'table-warning'
    },
    {
      day: 'Viernes',
      class: 'table-info'
    }
  ]
  public hours_days:any = [];
  public hours_selecteds:any = [];
```



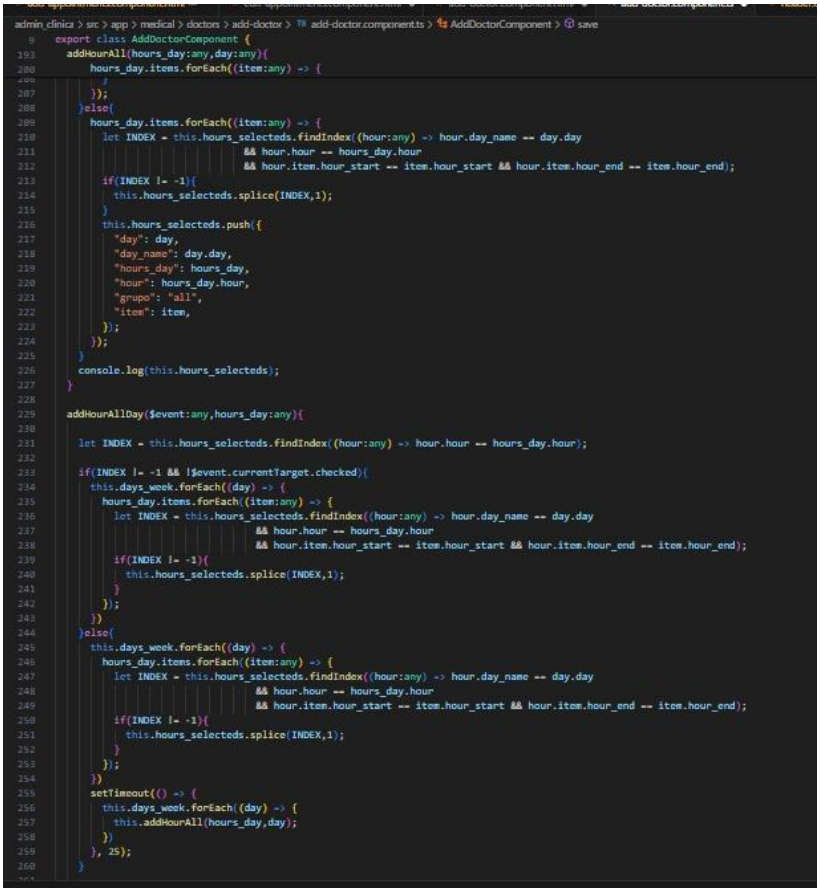
```
admin_clinica > src > app > medical > doctors > add-doctor > TS add-doctor.component.ts > AddDoctorComponent > save
  export class AddDoctorComponent {
  public hours_days:any = [];
  public hours_selecteds:any = [];
  constructor(
    public doctorsService: DoctorService,
  ) {

  }
  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    this.doctorsService.listConfig().subscribe((resp:any) => {
      console.log(resp);
      this.roles = resp.roles;
      this.specialities = resp.specialities;
      this.hours_days = resp.hours_days;
    })
  }

  save(){
    this.text_validation = '';
    if(!this.name || !this.email || !this.surname ||!this.mobile){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (Nombre, Apellido, Email, Teléfono Móvil)";
      return;
    }
    this.text_validation = '';
    if(!this.password ){
      this.text_validation = "ES NECESARIO UNA CONTRASEÑA";
      return;
    }

    this.text_validation = '';
    if(!this.selectedValue || !this.specialitie_id){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (Rol, Especialidad)";
      return;
    }

    if(this.password != this.password_confirmation){
      this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
      return;
    }

    if(this.hours_selecteds.length == 0){
      this.text_validation = "NECESITAS SELECCIONAR UN HORARIO AL MENOS";
      return;
    }
    console.log(this.selectedValue);

    let formData = new FormData();
    formData.append("name",this.name);
    formData.append("surname",this.surname);
    formData.append("email",this.email);
    formData.append("mobile",this.mobile);
    formData.append("birth_date",this.birth_date);
    formData.append("gender",this.gender+"");
    formData.append("education",this.education);
    formData.append("designation",this.designation);
    formData.append("address",this.address);
```

admin_clinica > src > app > medical > doctors > add-doctor > TS add-doctor.component.ts > AddDoctorComponent > save

```
  export class AddDoctorComponent {
    save(){

      let formData = new FormData();
      formData.append("name",this.name);
      formData.append("surname",this.surname);
      formData.append("email",this.email);
      formData.append("mobile",this.mobile);
      formData.append("birth_date",this.birth_date);
      formData.append("gender",this.gender+"");
      formData.append("education",this.education);
      formData.append("designation",this.designation);
      formData.append("address",this.address);
      formData.append("password",this.password);
      formData.append("role_id",this.selectedValue);
      formData.append("specialitie_id",this.specialitie_id);
      formData.append("imagen",this.FILE_AVATAR);

      let HOUR_SCHEDULES:any = [];

      this.days_week.forEach((day:any) => {
        let DAYS_HOURS = this.hours_selecteds.filter((hour_select:any) => hour_select.day_name == day.day);
        HOUR_SCHEDULES.push({
          day_name: day.day,
          children: DAYS_HOURS,
        });
      })

      formData.append("schedule_hours",JSON.stringify(HOUR_SCHEDULES));
      this.doctorsService.registerDoctor(formData).subscribe((resp:any) => {
        console.log(resp);

        if(resp.message == 403){
          this.text_validation = resp.message_text;
        }else{
          this.text_success = 'El usuario ha sido registrado correctamente';

          this.name = '';
          this.surname = '';
          this.email  = '';
          this.mobile  = '';
          this.birth_date  = '';
          this.gender  = 1;
          this.education  = '';
          this.designation  = '';
          this.address  = '';
          this.password  = '';
          this.password_confirmation  = '';
          this.selectedValue  = '';
          this.specialitie_id  = '';
          this.FILE_AVATAR = null;
          this.IMAGEN_PREVIZUALIZA = null;
          this.hours_selecteds = [];
        }

      })
    }
```

adn C:\PROYECTO_FINAL\admin_clinica\src\app\medical\appointment\add-appointments\add-appointments.component.html • Modified — AddDoctorComponent > save

```
    loadFile($event:any){
      if($event.target.files[0].type.indexOf("image") < 0){
        // alert("SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN");
        this.text_validation = "SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN";
        return;
      }
      this.text_validation = '';
      this.FILE_AVATAR = $event.target.files[0];
      let reader = new FileReader();
      reader.readAsDataURL(this.FILE_AVATAR);
      reader.onloadend = () => this.IMAGEN_PREVIZUALIZA = reader.result;
    }

    addHourItem(hours_day:any,day:any,item:any){

      let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour
                                && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
      if(INDEX != -1){
        this.hours_selecteds.splice(INDEX,1);
      }else{
        this.hours_selecteds.push({
          "day": day,
          "day_name": day.day,
          "hours_day": hours_day,
          "hour": hours_day.hour,
          "grupo": "none",
          "item": item,
        });
      }

      console.log(this.hours_selecteds);
    }

    addHourAll(hours_day:any,day:any){
      let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour && hour.grupo == "all");

      let COUNT_SELECTED = this.hours_selecteds.filter((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour).length;
      if(INDEX != -1 && COUNT_SELECTED ==  hours_day.items.length){
        hours_day.items.forEach((item:any) => {
          let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour
                                && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
          if(INDEX != -1){
            this.hours_selecteds.splice(INDEX,1);
          }
        });
      }else{
        hours_day.items.forEach((item:any) => {
          let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour
                                && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
          if(INDEX != -1){
            this.hours_selecteds.splice(INDEX,1);
```

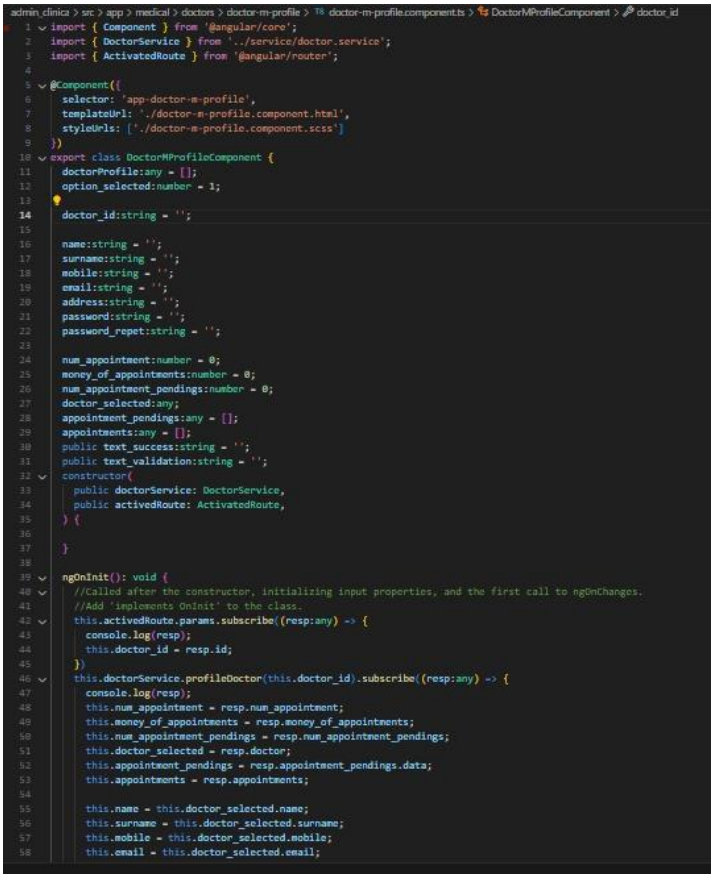


```
admin_clinica > src > app > medical > doctors > add-doctor > TS add-doctor.component.ts > AddDoctorComponent > save
  export class AddDoctorComponent {
    addHourAll(hours_day:any,day:any){
        hours_day.items.forEach((item:any) => {
          /
        });
      }else{
        hours_day.items.forEach((item:any) => {
          let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour
                                && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
          if(INDEX != -1){
            this.hours_selecteds.splice(INDEX,1);
          }
          this.hours_selecteds.push({
            "day": day,
            "day_name": day.day,
            "hours_day": hours_day,
            "hour": hours_day.hour,
            "grupo": "all",
            "item": item,
          });
        });
      }
      console.log(this.hours_selecteds);
    }

    addHourAllDay($event:any,hours_day:any){

      let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.hour == hours_day.hour);

      if(INDEX != -1 && !$event.currentTarget.checked){
        this.days_week.forEach((day) => {
          hours_day.items.forEach((item:any) => {
            let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour
                                && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
            if(INDEX != -1){
              this.hours_selecteds.splice(INDEX,1);
            }
          });
        })
      }else{
        this.days_week.forEach((day) => {
          hours_day.items.forEach((item:any) => {
            let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour
                                && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
            if(INDEX != -1){
              this.hours_selecteds.splice(INDEX,1);
            }
          });
        })
        setTimeout(() => {
          this.days_week.forEach((day) => {
            this.addHourAll(hours_day,day);
          })
        }, 25);
      }
```

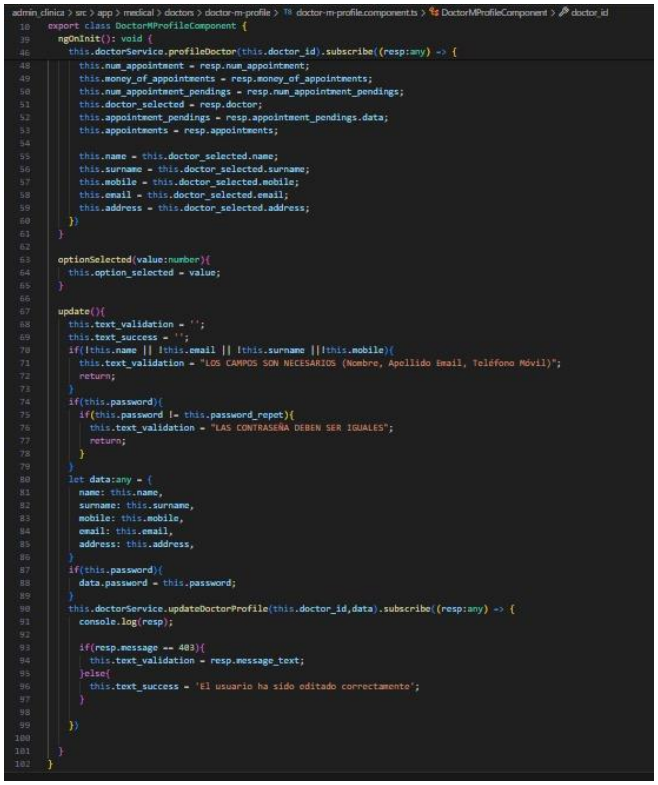
```
  }

  isCheckHourAll(hours_day:any,day:any){
    let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                          && hour.hour == hours_day.hour && hour.grupo == "all");

    let COUNT_SELECTED = this.hours_selecteds.filter((hour:any) => hour.day_name == day.day
                          && hour.hour == hours_day.hour).length;
    if(INDEX != -1 && COUNT_SELECTED ==  hours_day.items.length){
      return true;
    }else{
      return false;
    }
  }

  isCheckHour(hours_day:any,day:any,item:any){
    let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                          && hour.hour == hours_day.hour
                          && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
    if(INDEX != -1){
      return true;
    }else{
      return false;
    }
  }
}
```

**Anexo M**

*Funcionalidad del Perfil de Doctores*



```
admin_clinica > src > app > medical > doctors > doctor-m-profile > TS doctor-m-profile.component.ts > DoctorMProfileComponent > doctor_id
import { Component } from '@angular/core';
import { DoctorService } from '../service/doctor.service';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-doctor-m-profile',
  templateUrl: './doctor-m-profile.component.html',
  styleUrls: ['./doctor-m-profile.component.scss']
})
export class DoctorMProfileComponent {
  doctorProfile:any = [];
  option_selected:number = 1;

  doctor_id:string = '';

  name:string = '';
  surname:string = '';
  mobile:string = '';
  email:string = '';
  address:string = '';
  password:string = '';
  password_repet:string = '';

  num_appointment:number = 0;
  money_of_appointments:number = 0;
  num_appointment_pendings:number = 0;
  doctor_selected:any;
  appointment_pendings:any = [];
  appointments:any = [];
  public text_success:string = '';
  public text_validation:string = '';
  constructor(
    public doctorService: DoctorService,
    public activedRoute: ActivatedRoute,
  ) {

  }

  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    this.activedRoute.params.subscribe((resp:any) => {
      console.log(resp);
      this.doctor_id = resp.id;
    })
    this.doctorService.profileDoctor(this.doctor_id).subscribe((resp:any) => {
      console.log(resp);
      this.num_appointment = resp.num_appointment;
      this.money_of_appointments = resp.money_of_appointments;
      this.num_appointment_pendings = resp.num_appointment_pendings;
      this.doctor_selected = resp.doctor;
      this.appointment_pendings = resp.appointment_pendings.data;
      this.appointments = resp.appointments;

      this.name = this.doctor_selected.name;
      this.surname = this.doctor_selected.surname;
      this.mobile = this.doctor_selected.mobile;
      this.email = this.doctor_selected.email;
```

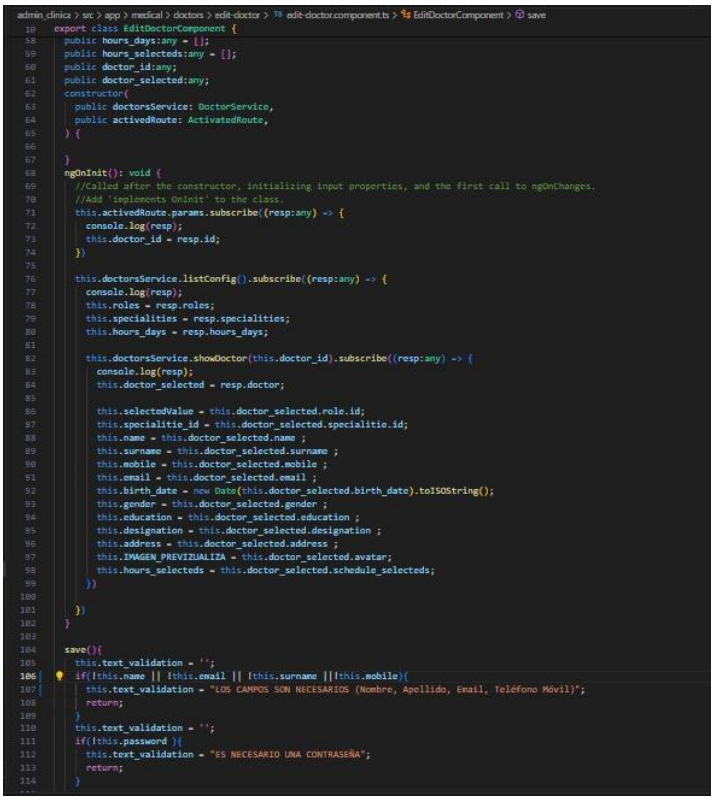


```
admin_clinica > src > app > medical > doctors > doctor-m-profile > TS doctor-m-profile.component.ts > DoctorMProfileComponent > doctor_id
export class DoctorMProfileComponent {
  ngOnInit(): void {
    this.doctorService.profileDoctor(this.doctor_id).subscribe((resp:any) => {
      this.num_appointment = resp.num_appointment;
      this.money_of_appointments = resp.money_of_appointments;
      this.num_appointment_pendings = resp.num_appointment_pendings;
      this.doctor_selected = resp.doctor;
      this.appointment_pendings = resp.appointment_pendings.data;
      this.appointments = resp.appointments;

      this.name = this.doctor_selected.name;
      this.surname = this.doctor_selected.surname;
      this.mobile = this.doctor_selected.mobile;
      this.email = this.doctor_selected.email;
      this.address = this.doctor_selected.address;
    })
  }

  optionSelected(value:number){
    this.option_selected = value;
  }

  update(){
    this.text_validation = '';
    this.text_success = '';
    if(!this.name || !this.email || !this.surname ||!this.mobile){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (Nombre, Apellido Email, Teléfono Móvil)";
      return;
    }
    if(this.password){
      if(this.password != this.password_repet){
        this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
        return;
      }
    }
    let data:any = {
      name: this.name,
      surname: this.surname,
      mobile: this.mobile,
      email: this.email,
      address: this.address,
    }
    if(this.password){
      data.password = this.password;
    }
    this.doctorService.updateDoctorProfile(this.doctor_id,data).subscribe((resp:any) => {
      console.log(resp);

      if(resp.message == 403){
        this.text_validation = resp.message_text;
      }else{
        this.text_success = 'El usuario ha sido editado correctamente';
      }

    })
  }
}
```

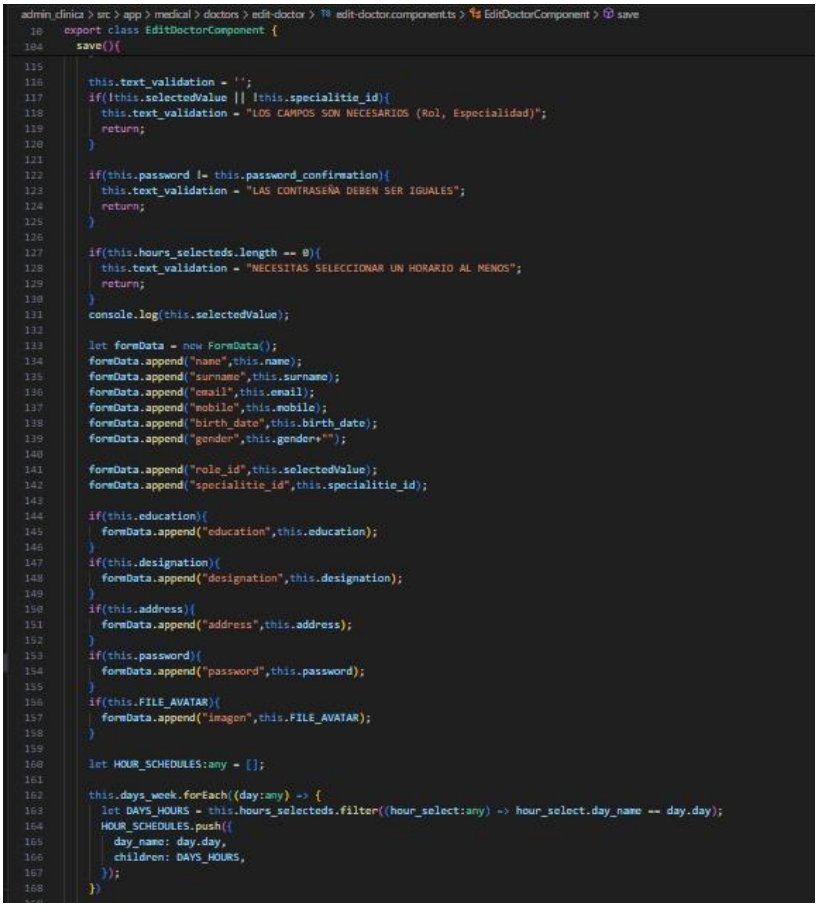
**Anexo N**

*Funcionalidad de Editar un Doctor*

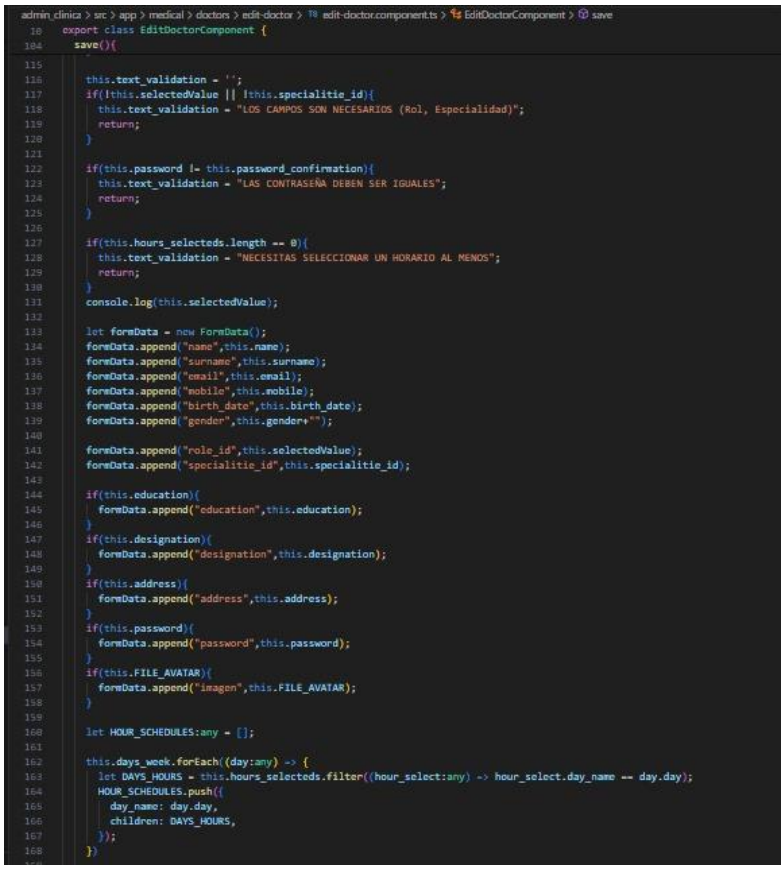
```
admin_clinica > src > app > medical > doctors > edit-doctor > edit-doctor.component.ts > EditDoctorComponent >
import { Component } from '@angular/core';
import { DoctorService } from '../service/doctor.service';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-edit-doctor',
  templateUrl: './edit-doctor.component.html',
  styleUrls: ['./edit-doctor.component.scss']
})
export class EditDoctorComponent {
  public selectedValue !: string  ;
  public name:string = '';
  public surname:string = '';
  public mobile:string = '';
  public email:string = '';
  public password:string = '';
  public password_confirmation:string = '';

  public birth_date:string = '';
  public gender:number = 1;
  public education:string = '';
  public designation:string = '';
  public address:string = '';

  public roles:any = [];

  public FILE_AVATAR:any;
  public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';

  public specialitie_id:any;
  public specialities:any = [];

  public text_success:string = '';
  public text_validation:string = '';

  public days_week = [
    {
      day: 'Lunes',
      class: 'table-primary'
    },
    {
      day: 'Martes',
      class: 'table-secondary'
    },
    {
      day: 'Miercoles',
      class: 'table-success'
    },
    {
      day: 'Jueves',
      class: 'table-warning'
    },
    {
      day: 'Viernes',
      class: 'table-info'
    }
  ]
```



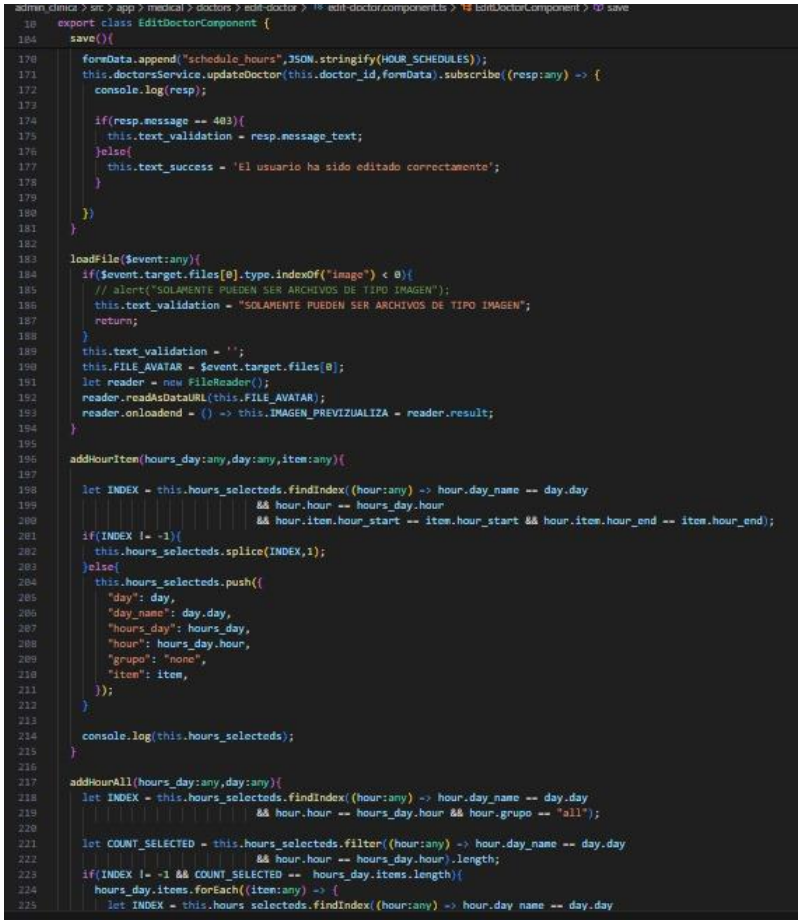
```
admin_clinica > src > app > medical > doctors > edit-doctor > edit-doctor.component.ts > EditDoctorComponent > save
export class EditDoctorComponent {
  public hours_days:any = [];
  public hours_selecteds:any = [];
  public doctor_id:any;
  public doctor_selected:any;
  constructor(
    public doctorsService: DoctorService,
    public activedRoute: ActivatedRoute,
  ) {

  }
  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    this.activedRoute.params.subscribe((resp:any) => {
      console.log(resp);
      this.doctor_id = resp.id;
    })

    this.doctorsService.listConfig().subscribe((resp:any) => {
      console.log(resp);
      this.roles = resp.roles;
      this.specialities = resp.specialities;
      this.hours_days = resp.hours_days;

      this.doctorsService.showDoctor(this.doctor_id).subscribe((resp:any) => {
        console.log(resp);
        this.doctor_selected = resp.doctor;

        this.selectedValue = this.doctor_selected.role.id;
        this.specialitie_id = this.doctor_selected.specialitie.id;
        this.name = this.doctor_selected.name ;
        this.surname = this.doctor_selected.surname ;
        this.mobile = this.doctor_selected.mobile ;
        this.email = this.doctor_selected.email ;
        this.birth_date = new Date(this.doctor_selected.birth_date).toISOString();
        this.gender = this.doctor_selected.gender ;
        this.education = this.doctor_selected.education ;
        this.designation = this.doctor_selected.designation ;
        this.address = this.doctor_selected.address ;
        this.IMAGEN_PREVIZUALIZA = this.doctor_selected.avatar;
        this.hours_selecteds = this.doctor_selected.schedule_selecteds;
      })

    })
  }

  save(){
    this.text_validation = '';
    if(!this.name || !this.email || !this.surname ||!this.mobile){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (Nombre, Apellido, Email, Teléfono Móvil)";
      return;
    }
    this.text_validation = '';
    if(!this.password ){
      this.text_validation = "ES NECESARIO UNA CONTRASEÑA";
      return;
    }
```



```
admin_clinica > src > app > medical > doctors > edit-doctor > edit-doctor.component.ts > EditDoctorComponent > save
export class EditDoctorComponent {
  save(){

    this.text_validation = '';
    if(!this.selectedValue || !this.specialitie_id){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (Rol, Especialidad)";
      return;
    }

    if(this.password != this.password_confirmation){
      this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
      return;
    }

    if(this.hours_selecteds.length == 0){
      this.text_validation = "NECESITAS SELECCIONAR UN HORARIO AL MENOS";
      return;
    }
    console.log(this.selectedValue);

    let formData = new FormData();
    formData.append("name",this.name);
    formData.append("surname",this.surname);
    formData.append("email",this.email);
    formData.append("mobile",this.mobile);
    formData.append("birth_date",this.birth_date);
    formData.append("gender",this.gender+"");

    formData.append("role_id",this.selectedValue);
    formData.append("specialitie_id",this.specialitie_id);

    if(this.education){
      formData.append("education",this.education);
    }
    if(this.designation){
      formData.append("designation",this.designation);
    }
    if(this.address){
      formData.append("address",this.address);
    }
    if(this.password){
      formData.append("password",this.password);
    }
    if(this.FILE_AVATAR){
      formData.append("imagen",this.FILE_AVATAR);
    }

    let HOUR_SCHEDULES:any = [];

    this.days_week.forEach((day:any) => {
      let DAYS_HOURS = this.hours_selecteds.filter((hour_select:any) => hour_select.day_name == day.day);
      HOUR_SCHEDULES.push({
        day_name: day.day,
        children: DAYS_HOURS,
      });
    })
```

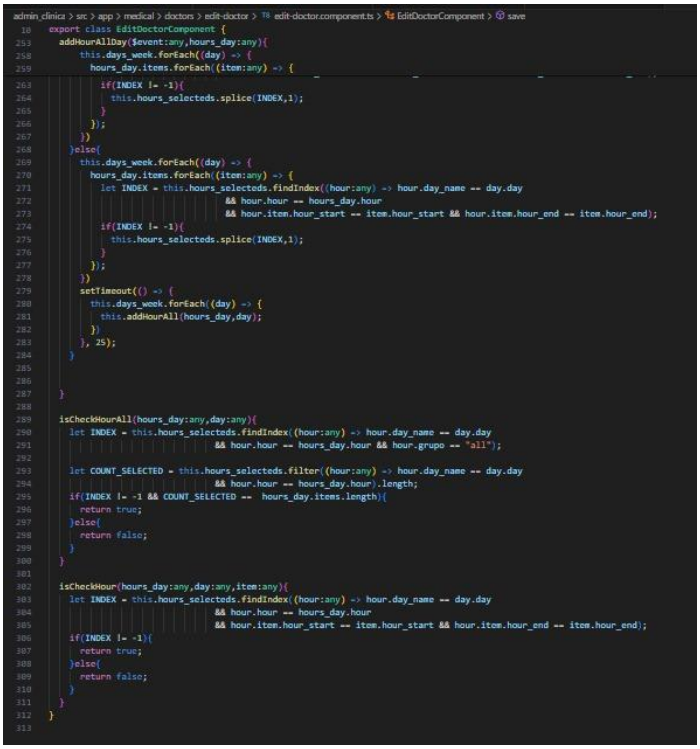


```
admin_clinica > src > app > medical > doctors > edit-doctor > edit-doctor.component.ts > EditDoctorComponent > save
export class EditDoctorComponent {
  save(){

    this.text_validation = '';
    if(!this.selectedValue || !this.specialitie_id){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (Rol, Especialidad)";
      return;
    }

    if(this.password != this.password_confirmation){
      this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
      return;
    }

    if(this.hours_selecteds.length == 0){
      this.text_validation = "NECESITAS SELECCIONAR UN HORARIO AL MENOS";
      return;
    }
    console.log(this.selectedValue);

    let formData = new FormData();
    formData.append("name",this.name);
    formData.append("surname",this.surname);
    formData.append("email",this.email);
    formData.append("mobile",this.mobile);
    formData.append("birth_date",this.birth_date);
    formData.append("gender",this.gender+"");

    formData.append("role_id",this.selectedValue);
    formData.append("specialitie_id",this.specialitie_id);

    if(this.education){
      formData.append("education",this.education);
    }
    if(this.designation){
      formData.append("designation",this.designation);
    }
    if(this.address){
      formData.append("address",this.address);
    }
    if(this.password){
      formData.append("password",this.password);
    }
    if(this.FILE_AVATAR){
      formData.append("imagen",this.FILE_AVATAR);
    }

    let HOUR_SCHEDULES:any = [];

    this.days_week.forEach((day:any) => {
      let DAYS_HOURS = this.hours_selecteds.filter((hour_select:any) => hour_select.day_name == day.day);
      HOUR_SCHEDULES.push({
        day_name: day.day,
        children: DAYS_HOURS,
      });
    })
```



```
admin_clinica > src > app > medical > doctors > edit-doctor > edit-doctor.component.ts > EditDoctorComponent > save
10  export class EditDoctorComponent {
184   save(){
170     formData.append("schedule_hours",JSON.stringify(HOUR_SCHEDULES));
171     this.doctorsService.updateDoctor(this.doctor_id,formData).subscribe((resp:any) => {
172       console.log(resp);
173
174       if(resp.message == 403){
175         this.text_validation = resp.message_text;
176       }else{
177         this.text_success = 'El usuario ha sido editado correctamente';
178       }
179
180     })
181   }
182
183   loadFile($event:any){
184     if($event.target.files[0].type.indexOf("image") < 0){
185       // alert("SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN");
186       this.text_validation = "SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN";
187       return;
188     }
189     this.text_validation = '';
190     this.FILE_AVATAR = $event.target.files[0];
191     let reader = new FileReader();
192     reader.readAsDataURL(this.FILE_AVATAR);
193     reader.onloadend = () => this.IMAGEN_PREVIZUALIZA = reader.result;
194   }
195
196   addHourItem(hours_day:any,day:any,item:any){
197
198     let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
199                               && hour.hour == hours_day.hour
200                               && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
201     if(INDEX != -1){
202       this.hours_selecteds.splice(INDEX,1);
203     }else{
204       this.hours_selecteds.push({
205         "day": day,
206         "day_name": day.day,
207         "hours_day": hours_day,
208         "hour": hours_day.hour,
209         "grupo": "none",
210         "item": item,
211       });
212     }
213
214     console.log(this.hours_selecteds);
215   }
216
217   addHourAll(hours_day:any,day:any){
218     let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
219                               && hour.hour == hours_day.hour && hour.grupo == "all");
220
221     let COUNT_SELECTED = this.hours_selecteds.filter((hour:any) => hour.day_name == day.day
222                               && hour.hour == hours_day.hour).length;
223     if(INDEX != -1 && COUNT_SELECTED ==  hours_day.items.length){
224       hours_day.items.forEach((item:any) => {
225         let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
```

```
admin_clinica > src > app > medical > doctors > edit-doctor > edit-doctor.component.ts > EditDoctorComponent > save
10  export class EditDoctorComponent {
217   addHourAll(hours_day:any,day:any){
218     let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
219                               && hour.hour == hours_day.hour && hour.grupo == "all");
220
221     let COUNT_SELECTED = this.hours_selecteds.filter((hour:any) => hour.day_name == day.day
222                               && hour.hour == hours_day.hour).length;
223     if(INDEX != -1 && COUNT_SELECTED ==  hours_day.items.length){
224       hours_day.items.forEach((item:any) => {
225         let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
226                               && hour.hour == hours_day.hour
227                               && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
228         if(INDEX != -1){
229           this.hours_selecteds.splice(INDEX,1);
230         }
231       });
232     }else{
233       hours_day.items.forEach((item:any) => {
234         let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
235                               && hour.hour == hours_day.hour
236                               && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
237         if(INDEX != -1){
238           this.hours_selecteds.splice(INDEX,1);
239         }
240         this.hours_selecteds.push({
241           "day": day,
242           "day_name": day.day,
243           "hours_day": hours_day,
244           "hour": hours_day.hour,
245           "grupo": "all",
246           "item": item,
247         });
248       });
249     }
250     console.log(this.hours_selecteds);
251   }
252
253   addHourAllDay($event:any,hours_day:any){
254
255     let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.hour == hours_day.hour);
256
257     if(INDEX != -1 && !$event.currentTarget.checked){
258       this.days_week.forEach((day) => {
259         hours_day.items.forEach((item:any) => {
260           let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
261                               && hour.hour == hours_day.hour
262                               && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
263           if(INDEX != -1){
264             this.hours_selecteds.splice(INDEX,1);
265           }
266         });
267       })
268     }else{
269       this.days_week.forEach((day) => {
270         hours_day.items.forEach((item:any) => {
271           let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
272                               && hour.hour == hours_day.hour
273                               && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
```



```
admin_clinica > src > app > medical > doctors > edit-doctor > TS edit-doctor.component.ts > EditDoctorComponent > save
  export class EditDoctorComponent {
    addHourAllDay($event:any,hours_day:any){
      this.days_week.forEach((day) => {
        hours_day.items.forEach((item:any) => {
          if(INDEX != -1){
            this.hours_selecteds.splice(INDEX,1);
          }
        });
      })
    }else{
      this.days_week.forEach((day) => {
        hours_day.items.forEach((item:any) => {
          let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour
                                && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
          if(INDEX != -1){
            this.hours_selecteds.splice(INDEX,1);
          }
        });
      })
      setTimeout(() => {
        this.days_week.forEach((day) => {
          this.addHourAll(hours_day,day);
        })
      }, 25);
    }


  }

  isCheckHourAll(hours_day:any,day:any){
    let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour && hour.grupo == "all");

    let COUNT_SELECTED = this.hours_selecteds.filter((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour).length;
    if(INDEX != -1 && COUNT_SELECTED ==  hours_day.items.length){
      return true;
    }else{
      return false;
    }
  }

  isCheckHour(hours_day:any,day:any,item:any){
    let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
                                && hour.hour == hours_day.hour
                                && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
    if(INDEX != -1){
      return true;
    }else{
      return false;
    }
  }
}
```

## *Anexo O*

*Funcionalidad de Listado de Doctores*

```
admin_clinica > src > app > medical > doctors > list-doctor > TS list-doctor.component.ts > ...
1   import { Component } from '@angular/core';
2   import { DoctorService } from '../service/doctor.service';
3   import { MatTableDataSource } from '@angular/material/table';
4
5   @Component({
6     selector: 'app-list-doctor',
7     templateUrl: './list-doctor.component.html',
8     styleUrls: ['./list-doctor.component.scss']
9   })
10  export class ListDoctorComponent {
11    public usersList:any = [];
12    dataSource!: MatTableDataSource<any>;
13
14    public showFilter = false;
15    public searchDataValue = '';
16    public lastIndex = 0;
17    public pageSize = 10;
18    public totalData = 0;
19    public skip = 0;//MIN
20    public limit: number = this.pageSize;//MAX
21    public pageIndex = 0;
22    public serialNumberArray: Array<number> = [];
23    public currentPage = 1;
24    public pageNumberArray: Array<number> = [];
25    public pageSelection: Array<any> = [];
26    public totalPages = 0;
27
28    public role_generals:any = [];
29    public doctor_selected:any;
30    public user:any;
31    constructor(
32      public doctorService: DoctorService,
33    ){
34
35    }
36    ngOnInit() {
37      this.getTableData();
38      this.user = this.doctorService.authService.user;
39    }
40
41    isPermision(permission:string){
42      if(this.user.roles.includes('Super-Admin')){
43        return true;
44      }
45      if(this.user.permissions.includes(permission)){
46        return true;
47      }
48      return false;
49    }
50    private getTableData(): void {
51      this.usersList = [];
52      this.serialNumberArray = [];
53
54      this.doctorService.listDoctors().subscribe((resp:any) => {
55
56        console.log(resp);
57
58        this.totalData = resp.users.data.length;
```

```
admin_clinica > src > app > medical > doctors > list-doctor > TS list-doctor.component.ts > ...
export class ListDoctorComponent {
  private getTableData(): void {
    this.doctorService.listDoctors().subscribe((resp:any) => {
      this.totalData = resp.users.data.length;
      this.role_generals = resp.users.data;
      this.getTableDataGeneral();
    })


  }

  getTableDataGeneral() {
    this.usersList = [];
    this.serialNumberArray = [];

    this.role_generals.map((res: any, index: number) => {
      const serialNumber = index + 1;
      if (index >= this.skip && serialNumber <= this.limit) {

        this.usersList.push(res);
        this.serialNumberArray.push(serialNumber);
      }
    });
    this.dataSource = new MatTableDataSource<any>(this.usersList);
    this.calculateTotalPages(this.totalData, this.pageSize);
  }

  selectUser(rol:any){
    this.doctor_selected = rol;
  }

  deleteUser(){

    this.doctorService.deleteDoctor(this.doctor_selected.id).subscribe((resp:any) => {
      console.log(resp);
      let INDEX = this.usersList.findIndex((item:any) => item.id == this.doctor_selected.id);
      if(INDEX != -1){
        this.usersList.splice(INDEX,1);

        $('#delete_patient').hide();
        $("#delete_patient").removeClass("show");
        $(".modal-backdrop").remove();
        $("body").removeClass();
        $("body").removeAttr("style");

        this.doctor_selected = null;
      }
    })
  }
  // eslint-disable-next-line @typescript-eslint/no-explicit-any
  public searchData(value: any): void {
    this.dataSource.filter = value.trim().toLowerCase();
    this.usersList = this.dataSource.filteredData;
  }

  public sortData(sort: any) {
    const data = this.usersList.slice();
```

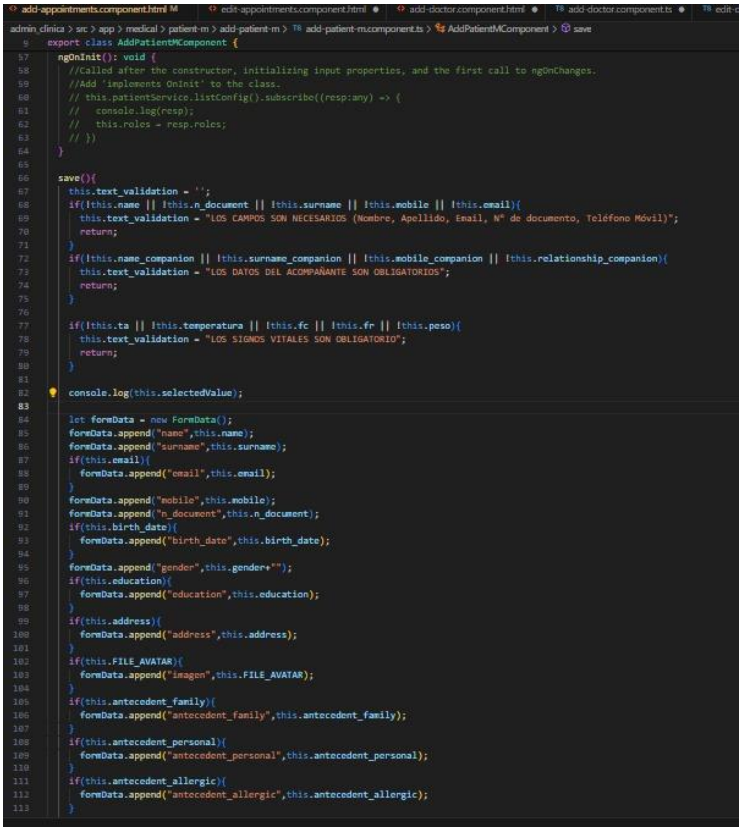
```
admin_clinica > src > app > medical > doctors > list-doctor > TS list-doctor.component.ts > ...
export class ListDoctorComponent {

  public sortData(sort: any) {
    const data = this.usersList.slice();

    if (!sort.active || sort.direction === '') {
      this.usersList = data;
    } else {
      this.usersList = data.sort((a:any, b:any) => {
        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        const aValue = (a as any)[sort.active];
        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        const bValue = (b as any)[sort.active];
        return (aValue < bValue ? -1 : 1) * (sort.direction === 'asc' ? 1 : -1);
      });
    }
  }

  public getMoreData(event: string): void {
    if (event == 'next') {
      this.currentPage++;
      this.pageIndex = this.currentPage - 1;
      this.limit += this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableDataGeneral();
    } else if (event == 'previous') {
      this.currentPage--;
      this.pageIndex = this.currentPage - 1;
      this.limit -= this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableDataGeneral();
    }
  }

  public moveToPage(pageNumber: number): void {
    this.currentPage = pageNumber;
    this.skip = this.pageSelection[pageNumber - 1].skip;
    this.limit = this.pageSelection[pageNumber - 1].limit;
    if (pageNumber > this.currentPage) {
      this.pageIndex = pageNumber - 1;
    } else if (pageNumber < this.currentPage) {
      this.pageIndex = pageNumber + 1;
    }
    this.getTableDataGeneral();
  }

  public PageSize(): void {
    this.pageSelection = [];
    this.limit = this.pageSize;
    this.skip = 0;
    this.currentPage = 1;
    this.searchDataValue = '';
    this.getTableDataGeneral();
  }

  private calculateTotalPages(totalData: number, pageSize: number): void {
    this.pageNumberArray = [];
    this.totalPages = totalData / pageSize;
```

```
admin_clinica > src > app > medical > doctors > list-doctor > TS list-doctor.component.ts > ...
  export class ListDoctorComponent {
    public getMoreData(event: string): void {
      this.pageIndex = this.currentPage + 1;
      this.limit += this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableDataGeneral();
    } else if (event == 'previous') {
      this.currentPage--;
      this.pageIndex = this.currentPage - 1;
      this.limit -= this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableDataGeneral();
    }
  }

  public moveToPage(pageNumber: number): void {
    this.currentPage = pageNumber;
    this.skip = this.pageSelection[pageNumber - 1].skip;
    this.limit = this.pageSelection[pageNumber - 1].limit;
    if (pageNumber > this.currentPage) {
      this.pageIndex = pageNumber - 1;
    } else if (pageNumber < this.currentPage) {
      this.pageIndex = pageNumber + 1;
    }
    this.getTableDataGeneral();
  }

  public PageSize(): void {
    this.pageSelection = [];
    this.limit = this.pageSize;
    this.skip = 0;
    this.currentPage = 1;
    this.searchDataValue = '';
    this.getTableDataGeneral();
  }

  private calculateTotalPages(totalData: number, pageSize: number): void {
    this.pageNumberArray = [];
    this.totalPages = totalData / pageSize;
    if (this.totalPages % 1 != 0) {
      this.totalPages = Math.trunc(this.totalPages + 1);
    }
    /* eslint no-var: off */
    for (var i = 1; i <= this.totalPages; i++) {
      const limit = pageSize * i;
      const skip = limit - pageSize;
      this.pageNumberArray.push(i);
      this.pageSelection.push({ skip: skip, limit: limit });
      // 1
      // 0 - 10
      // 2
      // 10 - 20
    }
  }
}
```

## Anexo P

*Funcionalidad de Añadir Paciente*

```
admin_clinica > src > app > medical > patient-m > add-patient-m > TS add-patient-m.component.ts > AddPatientMComponent > save
import { Component } from '@angular/core';
import { PatientMService } from '../service/patient-m.service';

@Component({
  selector: 'app-add-patient-m',
  templateUrl: './add-patient-m.component.html',
  styleUrls: ['./add-patient-m.component.scss']
})
export class AddPatientMComponent {
  public selectedValue !: string  ;
  public name:string = '';
  public surname:string = '';
  public mobile:string = '';
  public email:string = '';

  public birth_date:string = '';
  public gender:number = 1;
  public education:string = '';
  public address:string = '';

  public antecedent_family:string = '';
  public antecedent_personal:string = '';
  public antecedent_allergic:string = '';

  public name_companion:string = '';
  public surname_companion:string = '';
  public mobile_companion:string = '';
  public relationship_companion:string = '';

  public name_responsible:string = '';
  public surname_responsible:string = '';
  public mobile_responsible:string = '';
  public relationship_responsible:string = '';

  public current_disease:string = '';

  public ta :number = 0;
  public temperatura :number = 0;
  public fc :number = 0;
  public fr :number = 0;
  public peso :number = 0;

  public n_document:any = null;

  public roles:any = [];

  public FILE_AVATAR:any;
  public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';

  public text_success:string = '';
  public text_validation:string = '';
  constructor(
    public patientService: PatientMService,
  ) {

  }
  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
```



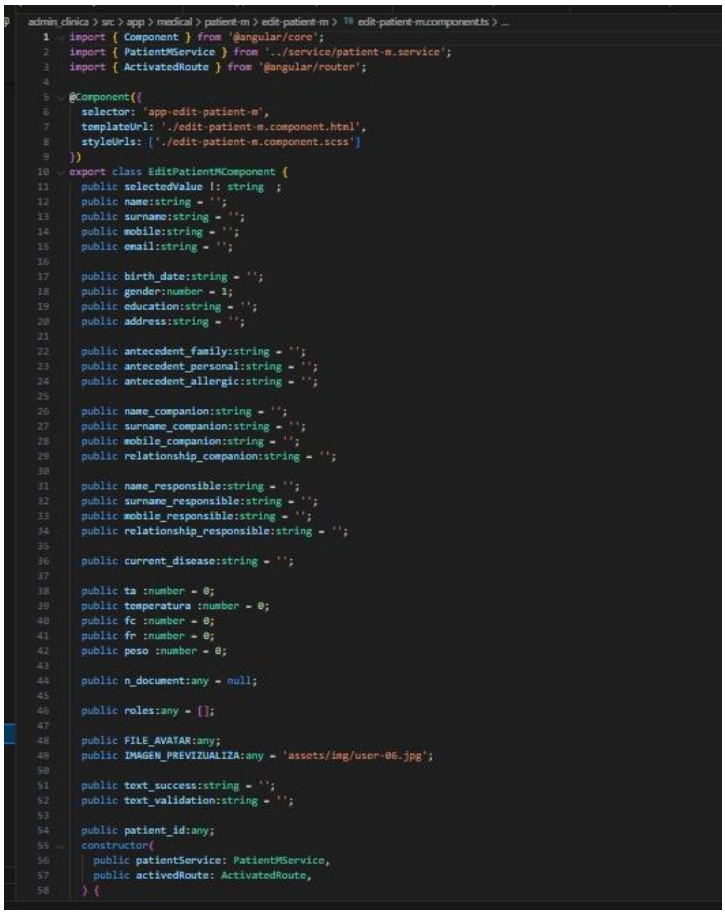
```
export class AddPatientMComponent {
  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    // this.patientService.listConfig().subscribe((resp:any) => {
    //   console.log(resp);
    //   this.roles = resp.roles;
    // })
  }

  save(){
    this.text_validation = '';
    if(!this.name || !this.n_document || !this.surname || !this.mobile || !this.email){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (Nombre, Apellido, Email, N° de documento, Teléfono Móvil)";
      return;
    }
    if(!this.name_companion || !this.surname_companion || !this.mobile_companion || !this.relationship_companion){
      this.text_validation = "LOS DATOS DEL ACOMPAÑANTE SON OBLIGATORIOS";
      return;
    }

    if(!this.ta || !this.temperatura || !this.fc || !this.fr || !this.peso){
      this.text_validation = "LOS SIGNOS VITALES SON OBLIGATORIO";
      return;
    }

    console.log(this.selectedValue);

    let formData = new FormData();
    formData.append("name",this.name);
    formData.append("surname",this.surname);
    if(this.email){
      formData.append("email",this.email);
    }
    formData.append("mobile",this.mobile);
    formData.append("n_document",this.n_document);
    if(this.birth_date){
      formData.append("birth_date",this.birth_date);
    }
    formData.append("gender",this.gender+"");
    if(this.education){
      formData.append("education",this.education);
    }
    if(this.address){
      formData.append("address",this.address);
    }
    if(this.FILE_AVATAR){
      formData.append("imagen",this.FILE_AVATAR);
    }
    if(this.antecedent_family){
      formData.append("antecedent_family",this.antecedent_family);
    }
    if(this.antecedent_personal){
      formData.append("antecedent_personal",this.antecedent_personal);
    }
    if(this.antecedent_allergic){
      formData.append("antecedent_allergic",this.antecedent_allergic);
    }
```

```
export class AddPatientMComponent {
  save(){
    }

    formData.append("name_companion",this.name_companion);
    formData.append("surname_companion",this.surname_companion);
    if(this.mobile_companion){
      formData.append("mobile_companion",this.mobile_companion);
    }
    if(this.relationship_companion){
      formData.append("relationship_companion",this.relationship_companion);
    }
    if(this.name_responsible){
      formData.append("name_responsible",this.name_responsible);
    }
    if(this.surname_responsible){
      formData.append("surname_responsible",this.surname_responsible);
    }
    if(this.mobile_responsible){
      formData.append("mobile_responsible",this.mobile_responsible);
    }
    if(this.relationship_responsible){
      formData.append("relationship_responsible",this.relationship_responsible);
    }
    if(this.current_disease){
      formData.append("current_disease",this.current_disease);
    }

    formData.append("ta",this.ta+"");
    formData.append("temperatura",this.temperatura+"");
    formData.append("fc",this.fc+"");
    formData.append("fr",this.fr+"");
    formData.append("peso",this.peso+"");
    this.patientService.registerPatient(formData).subscribe((resp:any) => {
      console.log(resp);

      if(resp.message == 403){
        this.text_validation = resp.message_text;
      }else{
        this.text_success = 'El paciente ha sido registrado correctamente';

        this.name = '';
        this.surname = '';
        this.email = '';
        this.mobile = '';
        this.birth_date = '';
        this.gender = 1;
        this.education = '';

        this.n_document = '';
        this.antecedent_family = '';
        this.antecedent_personal = '';
        this.antecedent_allergic = '';
        this.name_companion = '';
        this.surname_companion = '';
        this.mobile_companion = '';
        this.relationship_companion = '';
        this.name_responsible = '';
```

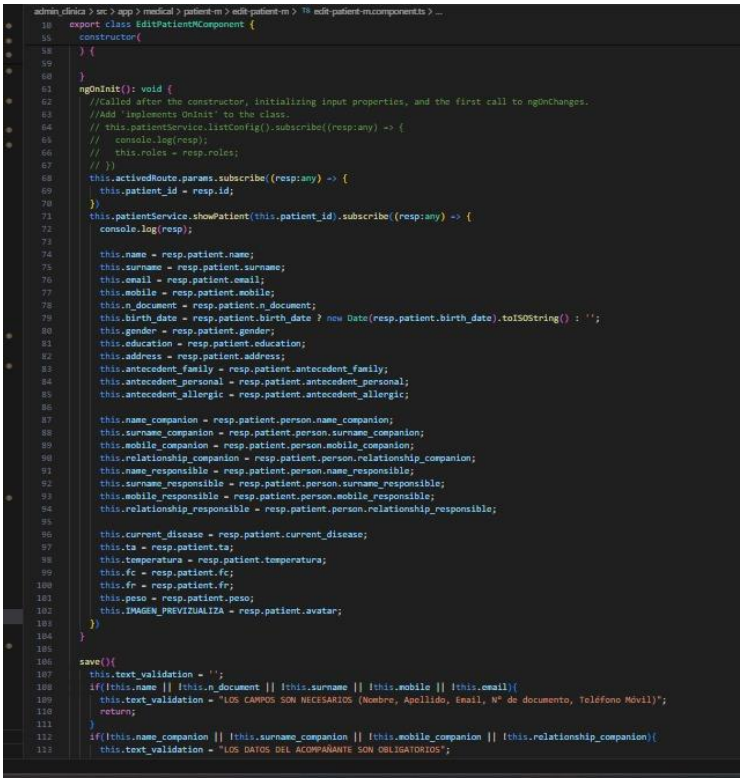
```
admin_clinica > src > app > medical > patient-m > add-patient-m > TS add-patient-m.component.ts > AddPatientMComponent > save
  export class AddPatientMComponent {
    save(){
      this.patientService.registerPatient(formData).subscribe((resp:any) => {

        if(resp.message == 403){
          this.text_validation = resp.message_text;
        }else{
          this.text_success = 'El paciente ha sido registrado correctamente';

          this.name = '';
          this.surname = '';
          this.email  = '';
          this.mobile  = '';
          this.birth_date  = '';
          this.gender  = 1;
          this.education  = '';

          this.n_document  = '';
          this.antecedent_family  = '';
          this.antecedent_personal  = '';
          this.antecedent_allergic  = '';
          this.name_companion  = '';
          this.surname_companion  = '';
          this.mobile_companion  = '';
          this.relationship_companion  = '';
          this.name_responsible  = '';
          this.surname_responsible  = '';
          this.mobile_responsible  = '';
          this.relationship_responsible  = '';
          this.current_disease  = '';
          this.ta = 0;
          this.temperatura = 0;
          this.fc = 0;
          this.fr = 0;
          this.peso = 0;
          this.address  = '';
          this.selectedValue  = '';
          this.FILE_AVATAR = null;
          this.IMAGEN_PREVIZUALIZA = null;
        }
      })
    }

    loadFile($event:any){
      if($event.target.files[0].type.indexOf("image") < 0){
        // alert("SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN");
        this.text_validation = "SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN";
        return;
      }
      this.text_validation = '';
      this.FILE_AVATAR = $event.target.files[0];
      let reader = new FileReader();
      reader.readAsDataURL(this.FILE_AVATAR);
      reader.onloadend = () => this.IMAGEN_PREVIZUALIZA = reader.result;
    }
  }
```

## Anexo Q

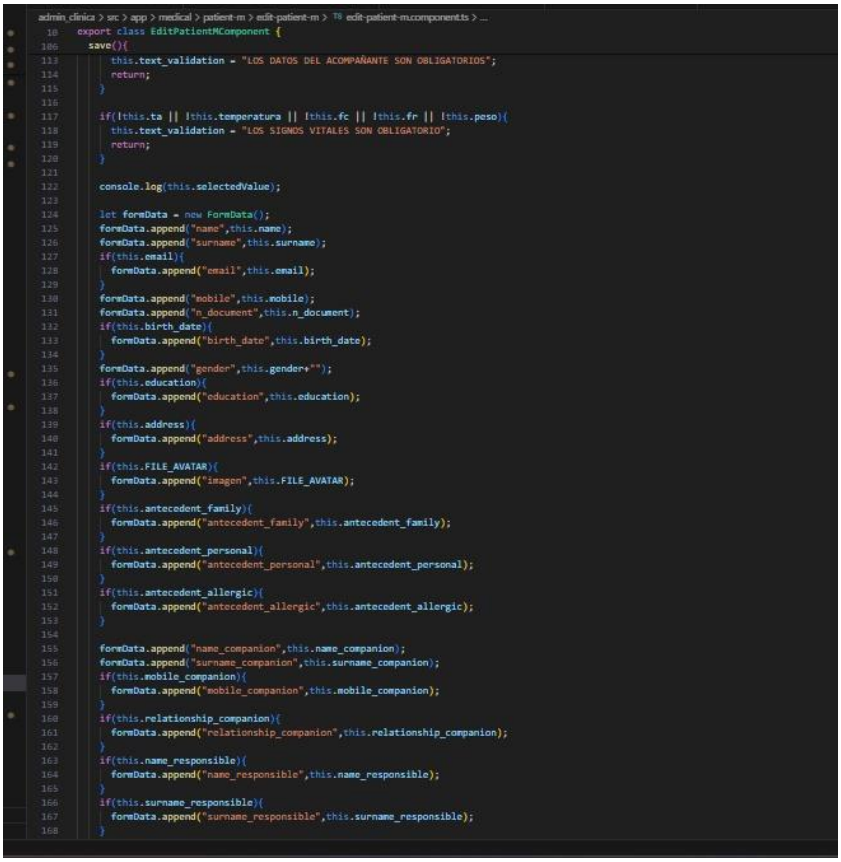
*Funcionalidad de Editar Paciente*



```
admin_clinica > src > app > medical > patient-m > edit-patient-m > TS edit-patient-m.component.ts > ...
import { Component } from '@angular/core';
import { PatientMService } from '../service/patient-m.service';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-edit-patient-m',
  templateUrl: './edit-patient-m.component.html',
  styleUrls: ['./edit-patient-m.component.scss']
})
export class EditPatientMComponent {
  public selectedValue !: string  ;
  public name:string = '';
  public surname:string = '';
  public mobile:string = '';
  public email:string = '';

  public birth_date:string = '';
  public gender:number = 1;
  public education:string = '';
  public address:string = '';

  public antecedent_family:string = '';
  public antecedent_personal:string = '';
  public antecedent_allergic:string = '';

  public name_companion:string = '';
  public surname_companion:string = '';
  public mobile_companion:string = '';
  public relationship_companion:string = '';

  public name_responsible:string = '';
  public surname_responsible:string = '';
  public mobile_responsible:string = '';
  public relationship_responsible:string = '';

  public current_disease:string = '';

  public ta :number = 0;
  public temperatura :number = 0;
  public fc :number = 0;
  public fr :number = 0;
  public peso :number = 0;

  public n_document:any = null;

  public roles:any = [];

  public FILE_AVATAR:any;
  public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';

  public text_success:string = '';
  public text_validation:string = '';

  public patient_id:any;
  constructor(
    public patientService: PatientMService,
    public activedRoute: ActivatedRoute,
  ) {
```



```
admin_clinica > src > app > medical > patient-m > edit-patient-m > TS edit-patient-m.component.ts > ...
export class EditPatientMComponent {
  constructor(
  ) {

  }
  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    // this.patientService.listConfig().subscribe((resp:any) => {
    //   console.log(resp);
    //   this.roles = resp.roles;
    // })
    this.activedRoute.params.subscribe((resp:any) => {
      this.patient_id = resp.id;
    })
    this.patientService.showPatient(this.patient_id).subscribe((resp:any) => {
      console.log(resp);

      this.name = resp.patient.name;
      this.surname = resp.patient.surname;
      this.email = resp.patient.email;
      this.mobile = resp.patient.mobile;
      this.n_document = resp.patient.n_document;
      this.birth_date = resp.patient.birth_date ? new Date(resp.patient.birth_date).toISOString() : '';
      this.gender = resp.patient.gender;
      this.education = resp.patient.education;
      this.address = resp.patient.address;
      this.antecedent_family = resp.patient.antecedent_family;
      this.antecedent_personal = resp.patient.antecedent_personal;
      this.antecedent_allergic = resp.patient.antecedent_allergic;

      this.name_companion = resp.patient.person.name_companion;
      this.surname_companion = resp.patient.person.surname_companion;
      this.mobile_companion = resp.patient.person.mobile_companion;
      this.relationship_companion = resp.patient.person.relationship_companion;
      this.name_responsible = resp.patient.person.name_responsible;
      this.surname_responsible = resp.patient.person.surname_responsible;
      this.mobile_responsible = resp.patient.person.mobile_responsible;
      this.relationship_responsible = resp.patient.person.relationship_responsible;

      this.current_disease = resp.patient.current_disease;
      this.ta = resp.patient.ta;
      this.temperatura = resp.patient.temperatura;
      this.fc = resp.patient.fc;
      this.fr = resp.patient.fr;
      this.peso = resp.patient.peso;
      this.IMAGEN_PREVIZUALIZA = resp.patient.avatar;
    })
  }

  save(){
    this.text_validation = '';
    if(!this.name || !this.n_document || !this.surname || !this.mobile || !this.email){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (Nombre, Apellido, Email, N° de documento, Teléfono Móvil)";
      return;
    }
    if(!this.name_companion || !this.surname_companion || !this.mobile_companion || !this.relationship_companion){
      this.text_validation = "LOS DATOS DEL ACOMPAÑANTE SON OBLIGATORIOS";
```



```
admin_clinica > src > app > medical > patient-m > edit-patient-m > TS edit-patient-m.component.ts > ...
export class EditPatientMComponent {
  save(){
      this.text_validation = "LOS DATOS DEL ACOMPAÑANTE SON OBLIGATORIOS";
      return;
    }

    if(!this.ta || !this.temperatura || !this.fc || !this.fr || !this.peso){
      this.text_validation = "LOS SIGNOS VITALES SON OBLIGATORIO";
      return;
    }

    console.log(this.selectedValue);

    let formData = new FormData();
    formData.append("name",this.name);
    formData.append("surname",this.surname);
    if(this.email){
      formData.append("email",this.email);
    }
    formData.append("mobile",this.mobile);
    formData.append("n_document",this.n_document);
    if(this.birth_date){
      formData.append("birth_date",this.birth_date);
    }
    formData.append("gender",this.gender+"");
    if(this.education){
      formData.append("education",this.education);
    }
    if(this.address){
      formData.append("address",this.address);
    }
    if(this.FILE_AVATAR){
      formData.append("imagen",this.FILE_AVATAR);
    }
    if(this.antecedent_family){
      formData.append("antecedent_family",this.antecedent_family);
    }
    if(this.antecedent_personal){
      formData.append("antecedent_personal",this.antecedent_personal);
    }
    if(this.antecedent_allergic){
      formData.append("antecedent_allergic",this.antecedent_allergic);
    }

    formData.append("name_companion",this.name_companion);
    formData.append("surname_companion",this.surname_companion);
    if(this.mobile_companion){
      formData.append("mobile_companion",this.mobile_companion);
    }
    if(this.relationship_companion){
      formData.append("relationship_companion",this.relationship_companion);
    }
    if(this.name_responsible){
      formData.append("name_responsible",this.name_responsible);
    }
    if(this.surname_responsible){
      formData.append("surname_responsible",this.surname_responsible);
    }
```

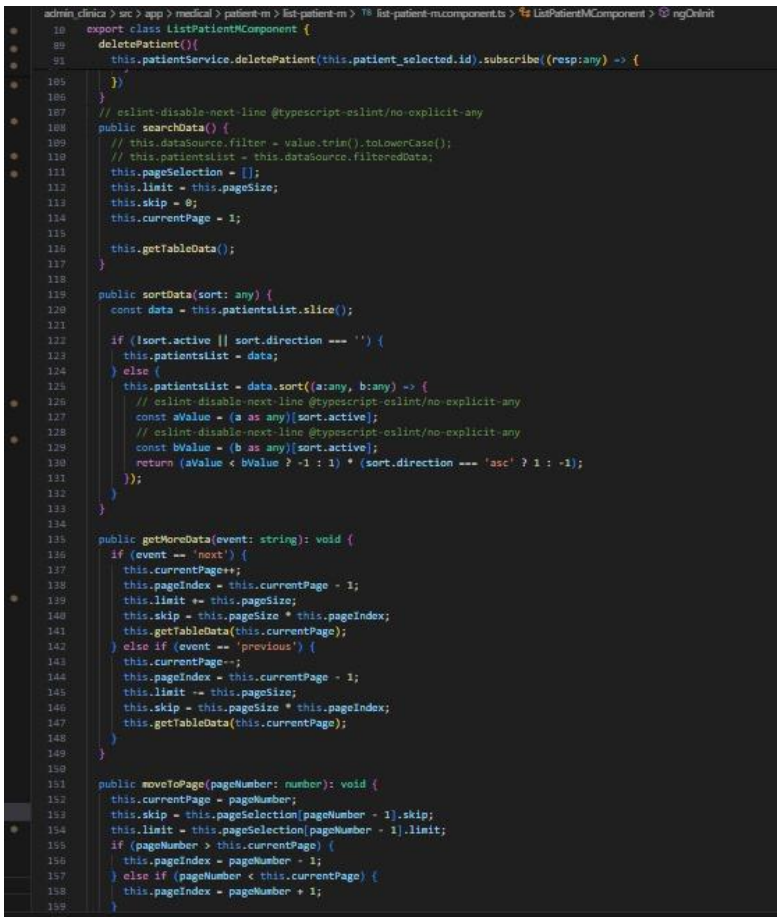
```
admin_clinica > src > app > medical > patient-m > edit-patient-m > TS edit-patient-m.component.ts > ...
  export class EditPatientMComponent {
    save(){

      formData.append("name_companion",this.name_companion);
      formData.append("surname_companion",this.surname_companion);
      if(this.mobile_companion){
        formData.append("mobile_companion",this.mobile_companion);
      }
      if(this.relationship_companion){
        formData.append("relationship_companion",this.relationship_companion);
      }
      if(this.name_responsible){
        formData.append("name_responsible",this.name_responsible);
      }
      if(this.surname_responsible){
        formData.append("surname_responsible",this.surname_responsible);
      }
      if(this.mobile_responsible){
        formData.append("mobile_responsible",this.mobile_responsible);
      }
      if(this.relationship_responsible){
        formData.append("relationship_responsible",this.relationship_responsible);
      }
      if(this.current_disease){
        formData.append("current_disease",this.current_disease);
      }

      formData.append("ta",this.ta+"");
      formData.append("temperatura",this.temperatura+"");
      formData.append("fc",this.fc+"");
      formData.append("fr",this.fr+"");
      formData.append("peso",this.peso+"");
      this.patientService.updatePatient(this.patient_id,formData).subscribe((resp:any) => {
        console.log(resp);

        if(resp.message == 403){
          this.text_validation = resp.message_text;
        }else{
          this.text_success = 'El paciente ha sido editado correctamente';
        }
      })
    }

    loadFile($event:any){
      if($event.target.files[0].type.indexOf("image") < 0){
        // alert("SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN");
        this.text_validation = "SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN";
        return;
      }
      this.text_validation = '';
      this.FILE_AVATAR = $event.target.files[0];
      let reader = new FileReader();
      reader.readAsDataURL(this.FILE_AVATAR);
      reader.onloadend = () => this.IMAGEN_PREVIZUALIZA = reader.result;
    }

  }
```

## Anexo R

*Funcionalidad de Listado de Pacientes*

```
admin_clinica > src > app > medical > patient-m > list-patient-m > TS list-patient-m.component.ts > ListPatientMComponent > ngOnInit
  import { Component } from '@angular/core';
  import { PatientMService } from '../service/patient-m.service';
  import { MatTableDataSource } from '@angular/material/table';

  @Component({
    selector: 'app-list-patient-m',
    templateUrl: './list-patient-m.component.html',
    styleUrls: ['./list-patient-m.component.scss']
  })
  export class ListPatientMComponent {

    public patientsList:any = [];
    dataSource!: MatTableDataSource<any>;

    public showFilter = false;
    public searchDataValue = '';
    public lastIndex = 0;
    public pageSize = 20;
    public totalData = 0;
    public skip = 0;//MIN
    public limit: number = this.pageSize;//MAX
    public pageIndex = 0;
    public serialNumberArray: Array<number> = [];
    public currentPage = 1;
    public pageNumberArray: Array<number> = [];
    public pageSelection: Array<any> = [];
    public totalPages = 0;

    public patient_generals:any = [];
    public patient_selected:any;
    public user:any;
    constructor(
      public patientService: PatientMService,
    ){

    }
    ngOnInit() {
      this.getTableData();
      this.user = this.patientService.authService.user;
    }

    isPermision(permission:string){
      if(this.user.roles.includes('Super-Admin')){
        return true;
      }
      if(this.user.permissions.includes(permission)){
        return true;
      }
      return false;
    }
    private getTableData(page=1): void {
      this.patientsList = [];
      this.serialNumberArray = [];

      this.patientService.listPatients(page,this.searchDataValue).subscribe((resp:any) => {

        console.log(resp);
```

```
admin_clinica > src > app > medical > patient-m > list-patient-m > TS list-patient-m.component.ts > ListPatientMComponent > ngOnInit
export class ListPatientMComponent {
  private getTableData(page=1): void {
    this.serialNumberArray = [];

    this.patientService.listPatients(page,this.searchDataValue).subscribe((resp:any) => {

      console.log(resp);

      this.totalData = resp.total;
      this.patientsList = resp.patients.data;
      // this.getTableDataGeneral();
      this.dataSource = new MatTableDataSource<any>(this.patientsList);
      this.calculateTotalPages(this.totalData, this.pageSize);
    })


  }

  getTableDataGeneral() {
    this.patientsList = [];
    this.serialNumberArray = [];

    this.patient_generals.map((res: any, index: number) => {
      const serialNumber = index + 1;
      if (index >= this.skip && serialNumber <= this.limit) {

        this.patientsList.push(res);
        this.serialNumberArray.push(serialNumber);
      }
    });
    this.dataSource = new MatTableDataSource<any>(this.patientsList);
    this.calculateTotalPages(this.totalData, this.pageSize);
  }

  selectUser(rol:any){
    this.patient_selected = rol;
  }

  deletePatient(){

    this.patientService.deletePatient(this.patient_selected.id).subscribe((resp:any) => {
      console.log(resp);
      let INDEX = this.patientsList.findIndex((item:any) => item.id == this.patient_selected.id);
      if(INDEX != -1){
        this.patientsList.splice(INDEX,1);

        $('#delete_patient').hide();
        $("#delete_patient").removeClass("show");
        $(".modal-backdrop").remove();
        $("body").removeClass();
        $("body").removeAttr("style");

        this.patient_selected = null;
      }
    })
  }
  // eslint-disable-next-line @typescript-eslint/no-explicit-any
```



```
admin_clinica > src > app > medical > patient-m > list-patient-m > TS list-patient-m.component.ts > ListPatientMComponent > ngOnInit
export class ListPatientMComponent {
  deletePatient(){
    this.patientService.deletePatient(this.patient_selected.id).subscribe((resp:any) => {
    })
  }
  // eslint-disable-next-line @typescript-eslint/no-explicit-any
  public searchData() {
    // this.dataSource.filter = value.trim().toLowerCase();
    // this.patientsList = this.dataSource.filteredData;
    this.pageSelection = [];
    this.limit = this.pageSize;
    this.skip = 0;
    this.currentPage = 1;

    this.getTableData();
  }

  public sortData(sort: any) {
    const data = this.patientsList.slice();

    if (!sort.active || sort.direction === '') {
      this.patientsList = data;
    } else {
      this.patientsList = data.sort((a:any, b:any) => {
        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        const aValue = (a as any)[sort.active];
        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        const bValue = (b as any)[sort.active];
        return (aValue < bValue ? -1 : 1) * (sort.direction === 'asc' ? 1 : -1);
      });
    }
  }

  public getMoreData(event: string): void {
    if (event == 'next') {
      this.currentPage++;
      this.pageIndex = this.currentPage - 1;
      this.limit += this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableData(this.currentPage);
    } else if (event == 'previous') {
      this.currentPage--;
      this.pageIndex = this.currentPage - 1;
      this.limit -= this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableData(this.currentPage);
    }
  }

  public moveToPage(pageNumber: number): void {
    this.currentPage = pageNumber;
    this.skip = this.pageSelection[pageNumber - 1].skip;
    this.limit = this.pageSelection[pageNumber - 1].limit;
    if (pageNumber > this.currentPage) {
      this.pageIndex = pageNumber - 1;
    } else if (pageNumber < this.currentPage) {
      this.pageIndex = pageNumber + 1;
    }
```

```
admin_clinica > src > app > medical > patient-m > list-patient-m > TS list-patient-m.component.ts > ListPatientMComponent >
export class ListPatientMComponent {
  public getMoreData(event: string): void {
    this.getTableData(this.currentPage);
  } else if (event == 'previous') {
    this.currentPage--;
    this.pageIndex = this.currentPage - 1;
    this.limit -= this.pageSize;
    this.skip = this.pageSize * this.pageIndex;
    this.getTableData(this.currentPage);
  }
 }

 public moveToPage(pageNumber: number): void {
  this.currentPage = pageNumber;
  this.skip = this.pageSelection[pageNumber - 1].skip;
  this.limit = this.pageSelection[pageNumber - 1].limit;
  if (pageNumber > this.currentPage) {
   this.pageIndex = pageNumber - 1;
  } else if (pageNumber < this.currentPage) {
   this.pageIndex = pageNumber + 1;
  }
  this.getTableData(this.currentPage);
 }

 public PageSize(): void {
  this.pageSelection = [];
  this.limit = this.pageSize;
  this.skip = 0;
  this.currentPage = 1;
  this.searchDataValue = '';
  this.getTableData();
 }

 private calculateTotalPages(totalData: number, pageSize: number): void {
  this.pageNumberArray = [];
  this.totalPages = totalData / pageSize;
  if (this.totalPages % 1 != 0) {
   this.totalPages = Math.trunc(this.totalPages + 1);//10.6 o 10.9 11
  }
  /* eslint no-var: off */
  for (var i = 1; i <= this.totalPages; i++) {
   const limit = pageSize * i;
   const skip = limit - pageSize;
   this.pageNumberArray.push(i);
   this.pageSelection.push({ skip: skip, limit: limit });
   // 1
   // 0 - 10
   // 2
   // 10 - 20
   // 3
   // 20 - 30
  }
 }

}
```

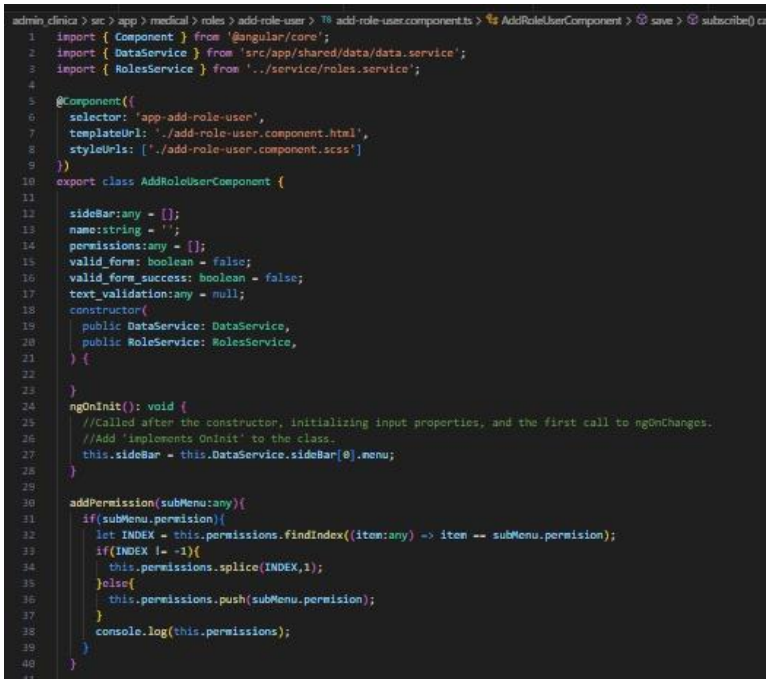
## Anexo S

*Funcionalidad del Perfil del Paciente*

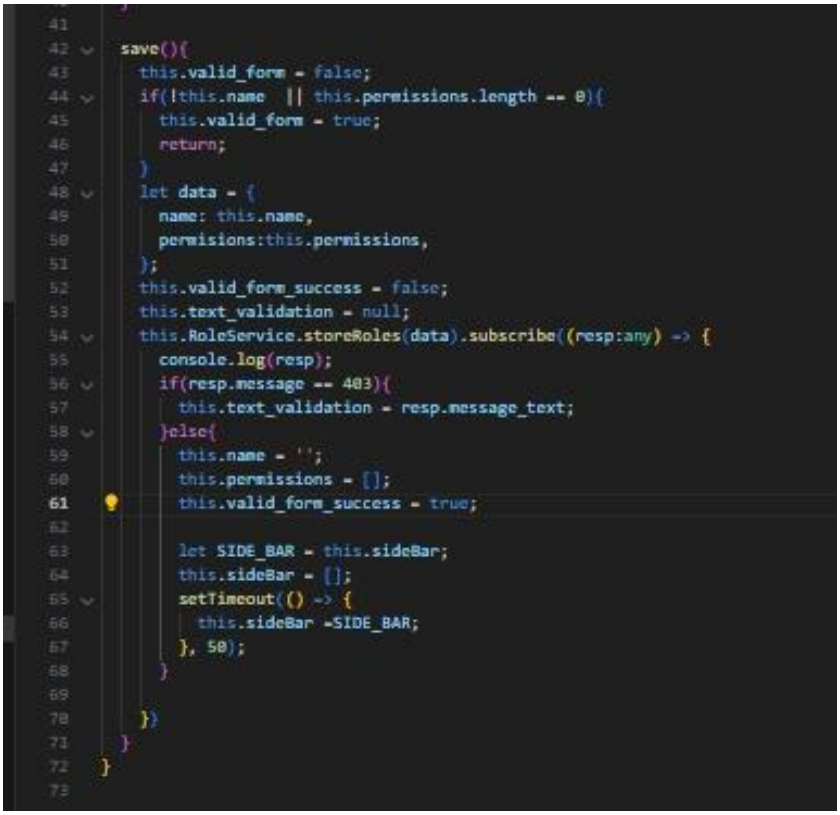
```
admin_clinica > src > app > medical > patient-m > patient-m-profile > TS patient-m-profile.component.ts > PatientMProfileComponent
import { Component } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { PatientMService } from '../service/patient-m.service';

@Component({
  selector: 'app-patient-m-profile',
  templateUrl: './patient-m-profile.component.html',
  styleUrls: ['./patient-m-profile.component.scss']
})
export class PatientMProfileComponent {
  patientProfile:any = [];
  option_selected:number = 1;

  patient_id:string = '';
  num_appointment:number = 0;
  money_of_appointments:number = 0;
  num_appointment_pendings:number = 0;
  patient_selected:any;
  appointment_pendings:any = [];
  appointments:any = [];

  constructor(
    public patientService: PatientMService,
    public activedRoute: ActivatedRoute,
  ) {

  }

  ngOnInit(): void {
    this.activedRoute.params.subscribe((resp: any) => {
      console.log(resp);
      this.patient_id = resp.id;
      // Asegúrate de llamar al servicio después de obtener el patient_id
      this.patientService.profiloPatient(this.patient_id).subscribe((resp: any) => {
        console.log(resp);
        this.num_appointment = resp.num_appointment;
        this.money_of_appointments = resp.money_of_appointments;
        this.num_appointment_pendings = resp.num_appointment_pendings;
        this.patient_selected = resp.patient;
        this.appointment_pendings = resp.appointment_pendings.data;
        this.appointments = resp.appointments;
      }, error => {
        console.error('ERROR:', error); // Manejo de errores de la llamada
      });
    });
  }

  optionSelected(value:number){
    this.option_selected = value;
  }
}
```

**Anexo T**

*Funcionalidad de Añadir un Rol*



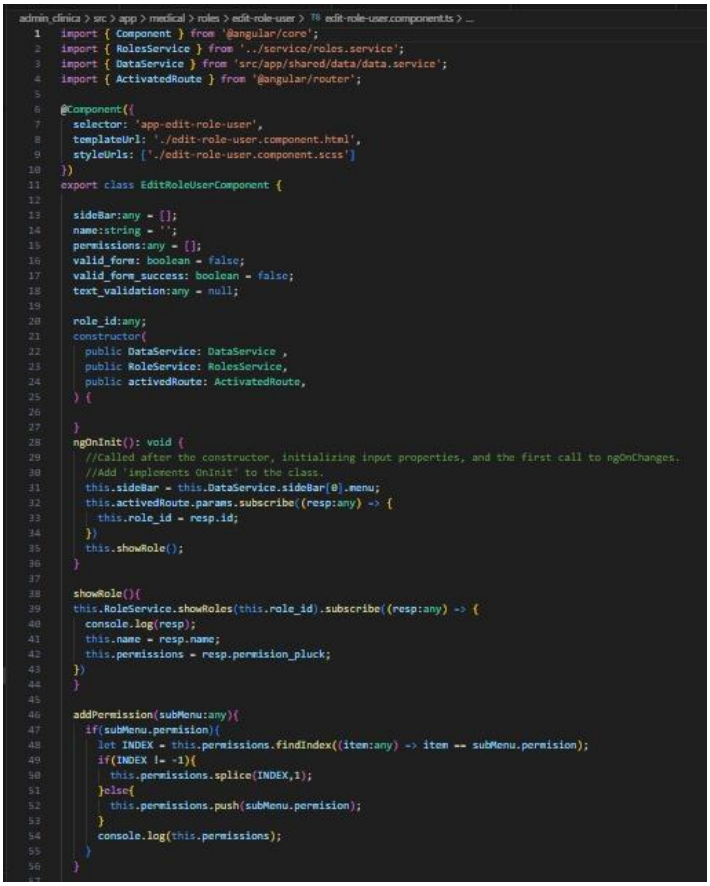
```
admin_clinica > src > app > medical > roles > add-role-user > add-role-user.component.ts > AddRoleUserComponent > save > subscribe() ca
import { Component } from '@angular/core';
import { DataService } from 'src/app/shared/data/data.service';
import { RolesService } from '../service/roles.service';

@Component({
  selector: 'app-add-role-user',
  templateUrl: './add-role-user.component.html',
  styleUrls: ['./add-role-user.component.scss']
})
export class AddRoleUserComponent {

  sideBar:any = [];
  name:string = '';
  permissions:any = [];
  valid_form: boolean = false;
  valid_form_success: boolean = false;
  text_validation:any = null;
  constructor(
    public DataService: DataService,
    public RoleService: RolesService,
  ) {

  }
  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    this.sideBar = this.DataService.sideBar[0].menu;
  }

  addPermission(subMenu:any){
    if(subMenu.permision){
      let INDEX = this.permissions.findIndex((item:any) => item == subMenu.permision);
      if(INDEX != -1){
        this.permissions.splice(INDEX,1);
      }else{
        this.permissions.push(subMenu.permision);
      }
      console.log(this.permissions);
    }
  }
```



```
  save(){
    this.valid_form = false;
    if(!this.name  || this.permissions.length == 0){
      this.valid_form = true;
      return;
    }
    let data = {
      name: this.name,
      permisions:this.permissions,
    };
    this.valid_form_success = false;
    this.text_validation = null;
    this.RoleService.storeRoles(data).subscribe((resp:any) => {
      console.log(resp);
      if(resp.message == 403){
        this.text_validation = resp.message_text;
      }else{
        this.name = '';
        this.permissions = [];
        this.valid_form_success = true;

        let SIDE_BAR = this.sideBar;
        this.sideBar = [];
        setTimeout(() => {
          this.sideBar =SIDE_BAR;
        }, 50);
      }

    })
  }
}
```

**Anexo U**

*Funcionalidad de Editar un Rol*



```
admin_clinica > src > app > medical > roles > edit-role-user > TS edit-role-user.component.ts > ...
import { Component } from '@angular/core';
import { RolesService } from '../service/roles.service';
import { DataService } from 'src/app/shared/data/data.service';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-edit-role-user',
  templateUrl: './edit-role-user.component.html',
  styleUrls: ['./edit-role-user.component.scss']
})
export class EditRoleUserComponent {

  sideBar:any = [];
  name:string = '';
  permissions:any = [];
  valid_form: boolean = false;
  valid_form_success: boolean = false;
  text_validation:any = null;

  role_id:any;
  constructor(
    public DataService: DataService ,
    public RoleService: RolesService,
    public activedRoute: ActivatedRoute,
  ) {

  }
  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    this.sideBar = this.DataService.sideBar[0].menu;
    this.activedRoute.params.subscribe((resp:any) => {
      this.role_id = resp.id;
    })
    this.showRole();
  }

  showRole(){
  this.RoleService.showRoles(this.role_id).subscribe((resp:any) => {
    console.log(resp);
    this.name = resp.name;
    this.permissions = resp.permision_pluck;
  })
  }

  addPermission(subMenu:any){
    if(subMenu.permision){
      let INDEX = this.permissions.findIndex((item:any) => item == subMenu.permision);
      if(INDEX != -1){
        this.permissions.splice(INDEX,1);
      }else{
        this.permissions.push(subMenu.permision);
      }
      console.log(this.permissions);
    }
  }
```

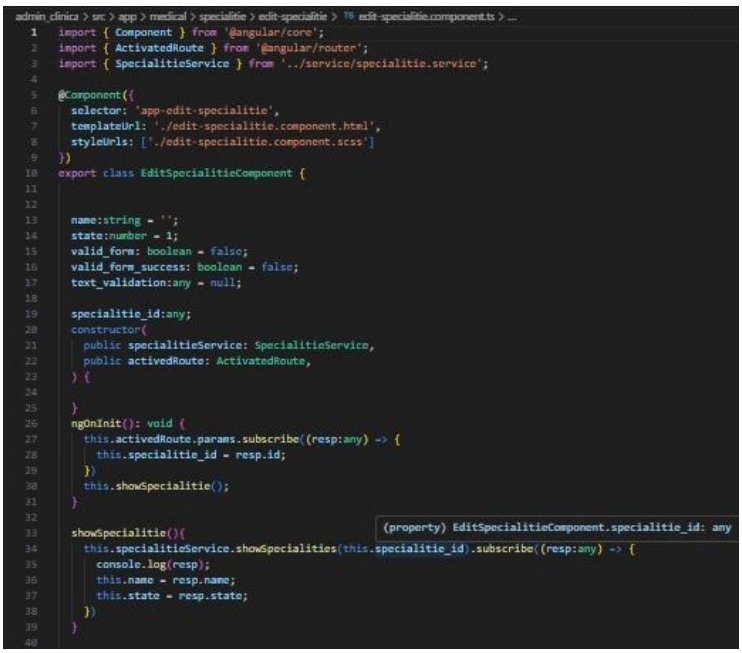
```
save(){
  this.valid_form = false;
  if(!this.name  || this.permissions.length == 0){
    this.valid_form = true;
    return;
  }
  let data = {
    name: this.name,
    permisions:this.permissions,
  };
  this.valid_form_success = false;
  this.text_validation = null;
  this.RoleService.editRoles(data,this.role_id).subscribe((resp:any) => {
    console.log(resp);
    if(resp.message == 403){
      this.text_validation = resp.message_text;
      return ;
    }
    this.valid_form_success = true;
  })
}

}
```

**Anexo V**

*Funcionalidad de Añadir Especialidad*


**Anexo W**

*Funcionalidad de Editar Especialidad*



## Anexo X

*Funcionalidad de Listado de Especialidades*

```
admin_clinica > src > app > medical > specialitie > list-specialitie > TS list-specialitie.component.ts > ...
  import { Component } from '@angular/core';
  import { SpecialitieService } from '../service/specialitie.service';
  import { MatTableDataSource } from '@angular/material/table';

  @Component({
    selector: 'app-list-specialitie',
    templateUrl: './list-specialitie.component.html',
    styleUrls: ['./list-specialitie.component.scss']
  })
  export class ListSpecialitieComponent {
    public specialitiesList:any = [];
    dataSource!: MatTableDataSource<any>;

    public showFilter = false;
    public searchDataValue = '';
    public lastIndex = 0;
    public pageSize = 10;
    public totalData = 0;
    public skip = 0;//MIN
    public limit: number = this.pageSize;//MAX
    public pageIndex = 0;
    public serialNumberArray: Array<number> = [];
    public currentPage = 1;
    public pageNumberArray: Array<number> = [];
    public pageSelection: Array<any> = [];
    public totalPages = 0;

    public specialitie_generals:any = [];
    public specialitie_selected:any;
    public user:any;

    constructor(
      public specialitiesService: SpecialitieService,
    ){

    }
    ngOnInit() {
      this.getTableData();
      this.user = this.specialitiesService.authService.user;
    }
    isPermision(permission:string){
      if(this.user.roles.includes('Super-Admin')){
        return true;
      }
      if(this.user.permissions.includes(permission)){
        return true;
      }
      return false;
    }
    private getTableData(): void {
      this.specialitiesList = [];
      this.serialNumberArray = [];

      this.specialitiesService.listSpecialities().subscribe((resp:any) => {

        console.log(resp);

        this.totalData = resp.specialities.length;
```

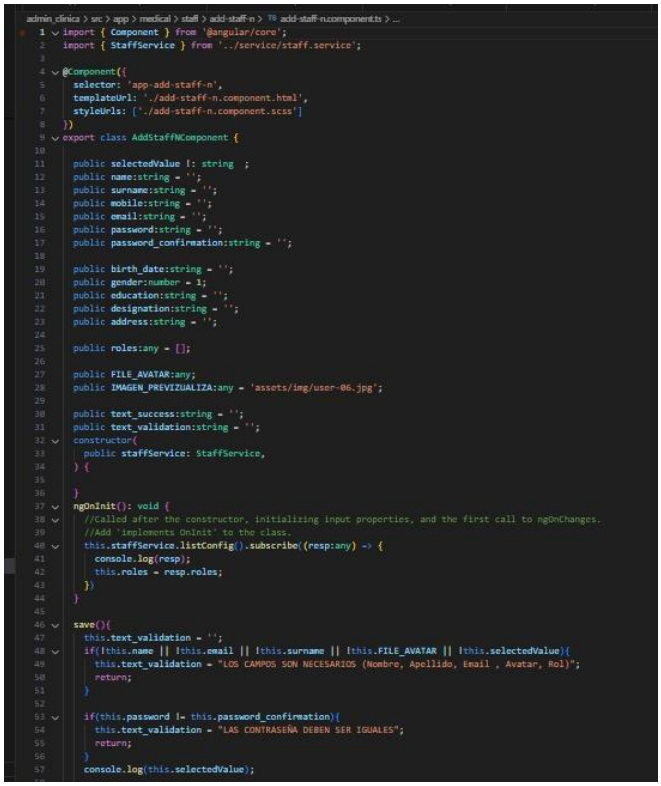
```
admin_clinica > src > app > medical > specialitie > list-specialitie > TS list-specialitie.component.ts > ...
  export class ListSpecialitieComponent {
    isPermision(permission:string){
  }
  private getTableData(): void {
    this.specialitiesList = [];
    this.serialNumberArray = [];

    this.specialitiesService.listSpecialities().subscribe((resp:any) => {

      console.log(resp);

      this.totalData = resp.specialities.length;
      this.specialitie_generals = resp.specialities;
      this.getTableDataGeneral();
    })


  }

  getTableDataGeneral() {
    this.specialitiesList = [];
    this.serialNumberArray = [];

    this.specialitie_generals.map((res: any, index: number) => {
      const serialNumber = index + 1;
      if (index >= this.skip && serialNumber <= this.limit) {

        this.specialitiesList.push(res);
        this.serialNumberArray.push(serialNumber);
      }
    });
    this.dataSource = new MatTableDataSource<any>(this.specialitiesList);
    this.calculateTotalPages(this.totalData, this.pageSize);
  }

  selectSpecialitie(rol:any){
    this.specialitie_selected = rol;
  }

  deleteSpecialitie(){

    this.specialitiesService.deleteSpecialities(this.specialitie_selected.id).subscribe((resp:any) => {
      console.log(resp);
      let INDEX = this.specialitiesList.findIndex((item:any) => item.id == this.specialitie_selected.id);
      if(INDEX != -1){
        this.specialitiesList.splice(INDEX,1);

        $('#delete_patient').hide();
        $("#delete_patient").removeClass("show");
        $(".modal-backdrop").remove();
        $("body").removeClass();
        $("body").removeAttr("style");

        this.specialitie_selected = null;
      }
    })
  }
  // eslint-disable-next-line @typescript-eslint/no-explicit-any
```

```
admin_clinica > src > app > medical > specialitie > list-specialitie > TS list-specialitie.component.ts > ...
  export class ListSpecialitieComponent {
  }
  // eslint-disable-next-line @typescript-eslint/no-explicit-any
  public searchData(value: any): void {
    this.dataSource.filter = value.trim().toLowerCase();
    this.specialitiesList = this.dataSource.filteredData;
  }

  public sortData(sort: any) {
    const data = this.specialitiesList.slice();

    if (!sort.active || sort.direction === '') {
      this.specialitiesList = data;
    } else {
      this.specialitiesList = data.sort((a:any, b:any) => {
        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        const aValue = (a as any)[sort.active];
        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        const bValue = (b as any)[sort.active];
        return (aValue < bValue ? -1 : 1) * (sort.direction === 'asc' ? 1 : -1);
      });
    }
  }

  public getMoreData(event: string): void {
    if (event == 'next') {
      this.currentPage++;
      this.pageIndex = this.currentPage - 1;
      this.limit += this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableDataGeneral();
    } else if (event == 'previous') {
      this.currentPage--;
      this.pageIndex = this.currentPage - 1;
      this.limit -= this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableDataGeneral();
    }
  }

  public moveToPage(pageNumber: number): void {
    this.currentPage = pageNumber;
    this.skip = this.pageSelection[pageNumber - 1].skip;
    this.limit = this.pageSelection[pageNumber - 1].limit;
    if (pageNumber > this.currentPage) {
      this.pageIndex = pageNumber - 1;
    } else if (pageNumber < this.currentPage) {
      this.pageIndex = pageNumber + 1;
    }
    this.getTableDataGeneral();
  }

  public PageSize(): void {
    this.pageSelection = [];
    this.limit = this.pageSize;
    this.skip = 0;
    this.currentPage = 1;
    this.searchDataValue = '';
```

```
admin_clinica > src > app > medical > specialitie > list-specialitie > TS list-specialitie.component.ts > ...
  export class ListSpecialitieComponent {
  public getMoreData(event: string): void {
    if (event == 'next') {
      this.currentPage++;
      this.pageIndex = this.currentPage - 1;
      this.limit += this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableDataGeneral();
    } else if (event == 'previous') {
      this.currentPage--;
      this.pageIndex = this.currentPage - 1;
      this.limit -= this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableDataGeneral();
    }
  }

  public moveToPage(pageNumber: number): void {
    this.currentPage = pageNumber;
    this.skip = this.pageSelection[pageNumber - 1].skip;
    this.limit = this.pageSelection[pageNumber - 1].limit;
    if (pageNumber > this.currentPage) {
      this.pageIndex = pageNumber - 1;
    } else if (pageNumber < this.currentPage) {
      this.pageIndex = pageNumber + 1;
    }
    this.getTableDataGeneral();
```

## Anexo Y

*Funcionalidad de Añadir Personal*



```
admin_clinica > src > app > medical > staff > add-staff-n > TS add-staff-n.component.ts > ...
import { Component } from '@angular/core';
import { StaffService } from '../service/staff.service';

@Component({
  selector: 'app-add-staff-n',
  templateUrl: './add-staff-n.component.html',
  styleUrls: ['./add-staff-n.component.scss']
})
export class AddStaffNComponent {

  public selectedValue !: string  ;
  public name:string = '';
  public surname:string = '';
  public mobile:string = '';
  public email:string = '';
  public password:string = '';
  public password_confirmation:string = '';

  public birth_date:string = '';
  public gender:number = 1;
  public education:string = '';
  public designation:string = '';
  public address:string = '';

  public roles:any = [];

  public FILE_AVATAR:any;
  public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';

  public text_success:string = '';
  public text_validation:string = '';
  constructor(
    public staffService: StaffService,
  ) {

  }
  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    this.staffService.listConfig().subscribe((resp:any) => {
      console.log(resp);
      this.roles = resp.roles;
    })
  }

  save(){
    this.text_validation = '';
    if(!this.name || !this.email || !this.surname || !this.FILE_AVATAR || !this.selectedValue){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (Nombre, Apellido, Email , Avatar, Rol)";
      return;
    }

    if(this.password != this.password_confirmation){
      this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
      return;
    }
    console.log(this.selectedValue);
```

```
admin_clinica > src > app > medical > staff > add-staff-n > TS add-staff-n.component.ts > ...
  export class AddStaffNComponent {
    save(){

      let formData = new FormData();
      formData.append("name",this.name);
      formData.append("surname",this.surname);
      formData.append("email",this.email);
      formData.append("mobile",this.mobile);
      formData.append("birth_date",this.birth_date);
      formData.append("gender",this.gender+"");
      formData.append("education",this.education);
      formData.append("designation",this.designation);
      formData.append("address",this.address);
      formData.append("password",this.password);
      formData.append("role_id",this.selectedValue);
      formData.append("imagen",this.FILE_AVATAR);

      this.staffService.registerUser(formData).subscribe((resp:any) => {
        console.log(resp);

        if(resp.message == 403){
          this.text_validation = resp.message_text;
        }else{
          this.text_success = 'El usuario ha sido registrado correctamente';

          this.name = '';
          this.surname = '';
          this.email = '';
          this.mobile = '';
          this.birth_date = '';
          this.gender = 1;
          this.education = '';
          this.designation = '';
          this.address = '';
          this.password = '';
          this.password_confirmation = '';
          this.selectedValue = '';
          this.FILE_AVATAR = null;
          this.IMAGEN_PREVIZUALIZA = null;
        }

      })
    }

    loadFile($event:any){
      if($event.target.files[0].type.indexOf("image") < 0){
        // alert("SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN");
        this.text_validation = "SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN";
        return;
      }
      this.text_validation = '';
      this.FILE_AVATAR = $event.target.files[0];
      let reader = new FileReader();
      reader.readAsDataURL(this.FILE_AVATAR);
      reader.onloadend = () => this.IMAGEN_PREVIZUALIZA = reader.result;
    }
  }
```

## Anexo Z

*Funcionalidad de Editar Personal*

```
admin_clinica > src > app > medical > staff > edit-staff-n > TS edit-staff-n.component.ts > ...
import { Component } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { StaffService } from '../service/staff.service';

@Component({
  selector: 'app-edit-staff-n',
  templateUrl: './edit-staff-n.component.html',
  styleUrls: ['./edit-staff-n.component.scss']
})
export class EditStaffNComponent {

  public selectedValue !: string  ;
  public name:string = '';
  public surname:string = '';
  public mobile:string = '';
  public email:string = '';
  public password:string = '';
  public password_confirmation:string = '';

  public birth_date:string = '';
  public gender:number = 1;
  public education:string = '';
  public designation:string = '';
  public address:string = '';

  public roles:any = [];

  public FILE_AVATAR:any;
  public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';

  public text_success:string = '';
  public text_validation:string = '';

  public staff_id:any;
  public staff_selected:any;
  constructor(
    public staffService: StaffService,
    public activedRoute: ActivatedRoute
  ) {

  }
  ngOnInit(): void {
    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
    //Add 'implements OnInit' to the class.
    this.activedRoute.params.subscribe((resp:any) => {
      console.log(resp);
      this.staff_id = resp.id;
    })

    this.staffService.showUser(this.staff_id).subscribe((resp:any) => {
      console.log(resp);
      this.staff_selected = resp.user;

      this.selectedValue = this.staff_selected.role.id;
      this.name = this.staff_selected.name ;
      this.surname = this.staff_selected.surname ;
      this.mobile = this.staff_selected.mobile ;
      this.email = this.staff_selected.email ;
```

```
admin_clinica > src > app > medical > staff > edit-staff-n > TS edit-staff-n.component.ts > ...
export class EditStaffNComponent {
  ngOnInit(): void {
    this.staffService.showUser(this.staff_id).subscribe((resp:any) => {
      this.name = this.staff_selected.name ;
      this.surname = this.staff_selected.surname ;
      this.mobile = this.staff_selected.mobile ;
      this.email = this.staff_selected.email ;
      this.birth_date = new Date(this.staff_selected.birth_date).toISOString();
      this.gender = this.staff_selected.gender ;
      this.education = this.staff_selected.education ;
      this.designation = this.staff_selected.designation ;
      this.address = this.staff_selected.address ;
      this.IMAGEN_PREVIZUALIZA = this.staff_selected.avatar;
    })

    this.staffService.listConfig().subscribe((resp:any) => {
      console.log(resp);
      this.roles = resp.roles;
    })
  }

  save(){
    this.text_validation = '';
    if(!this.name || !this.email || !this.surname){
      this.text_validation = "LOS CAMPOS SON NECESARIOS (name,surname,email)";
      return;
    }
    if(this.password){
      if(this.password != this.password_confirmation){
        this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
        return;
      }
    }
    console.log(this.selectedValue);

    let formData = new FormData();
    formData.append("name",this.name);
    formData.append("surname",this.surname);
    formData.append("email",this.email);
    formData.append("mobile",this.mobile);
    formData.append("birth_date",this.birth_date);
    formData.append("gender",this.gender+"");
    if(this.education){
      formData.append("education",this.education);
    }
    if(this.designation){
      formData.append("designation",this.designation);
    }
    if(this.address){
      formData.append("address",this.address);
    }
    if(this.password){
      formData.append("password",this.password);
    }
    formData.append("role_id",this.selectedValue);
    if(this.FILE_AVATAR){
      formData.append("imagen",this.FILE_AVATAR);
```

```
admin_clinica > src > app > medical > staff > edit-staff-n > TS edit-staff-n.component.ts > ...
export class EditStaffNComponent {
  save(){
        this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
        return;
      }
    }
    console.log(this.selectedValue);

    let formData = new FormData();
    formData.append("name",this.name);
    formData.append("surname",this.surname);
    formData.append("email",this.email);
    formData.append("mobile",this.mobile);
    formData.append("birth_date",this.birth_date);
    formData.append("gender",this.gender+"");
    if(this.education){
      formData.append("education",this.education);
    }
    if(this.designation){
      formData.append("designation",this.designation);
    }
    if(this.address){
      formData.append("address",this.address);
    }
    if(this.password){
      formData.append("password",this.password);
    }
    formData.append("role_id",this.selectedValue);
    if(this.FILE_AVATAR){
      formData.append("imagen",this.FILE_AVATAR);
    }

    this.staffService.updateUser(this.staff_id,formData).subscribe((resp:any) => {
      console.log(resp);

      if(resp.message == 403){
        this.text_validation = resp.message_text;
      }else{
        this.text_success = 'El usuario ha editado correctamente';
      }

    })
  }

  loadFile($event:any){
    if($event.target.files[0].type.indexOf("image") < 0){
      // alert("SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN");
      this.text_validation = "SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN";
      return;
    }
    this.text_validation = '';
    this.FILE_AVATAR = $event.target.files[0];
    let reader = new FileReader();
    reader.readAsDataURL(this.FILE_AVATAR);
    reader.onloadend = () => this.IMAGEN_PREVIZUALIZA = reader.result;
  }

}
```

## Anexo AA

*Funcionalidad de Listado de Personal*

```
admin_clinica > src > app > medical > staff > list-staff-n > TS list-staff-n.component.ts > ...
import { Component } from '@angular/core';
import { StaffService } from '../service/staff.service';
import { MatTableDataSource } from '@angular/material/table';

@Component({
  selector: 'app-list-staff-n',
  templateUrl: './list-staff-n.component.html',
  styleUrls: ['./list-staff-n.component.scss']
})
export class ListStaffNComponent {

  public usersList:any = [];
  dataSource!: MatTableDataSource<any>;

  public showFilter = false;
  public searchDataValue = '';
  public lastIndex = 0;
  public pageSize = 10;
  public totalData = 0;
  public skip = 0;//MIN
  public limit: number = this.pageSize;//MAX
  public pageIndex = 0;
  public serialNumberArray: Array<number> = [];
  public currentPage = 1;
  public pageNumberArray: Array<number> = [];
  public pageSelection: Array<any> = [];
  public totalPages = 0;

  public role_generals:any = [];
  public staff_selected:any;
  public user:any;
  constructor(
    public staffService: StaffService,
  ){

  }
  ngOnInit() {
    this.getTableData();
    this.user = this.staffService.authService.user;
  }
  private getTableData(): void {
    this.usersList = [];
    this.serialNumberArray = [];

    this.staffService.listUsers().subscribe((resp:any) => {

      console.log(resp);

      this.totalData = resp.users.data.length;
      this.role_generals = resp.users.data;
      this.getTableDataGeneral();
    })
```

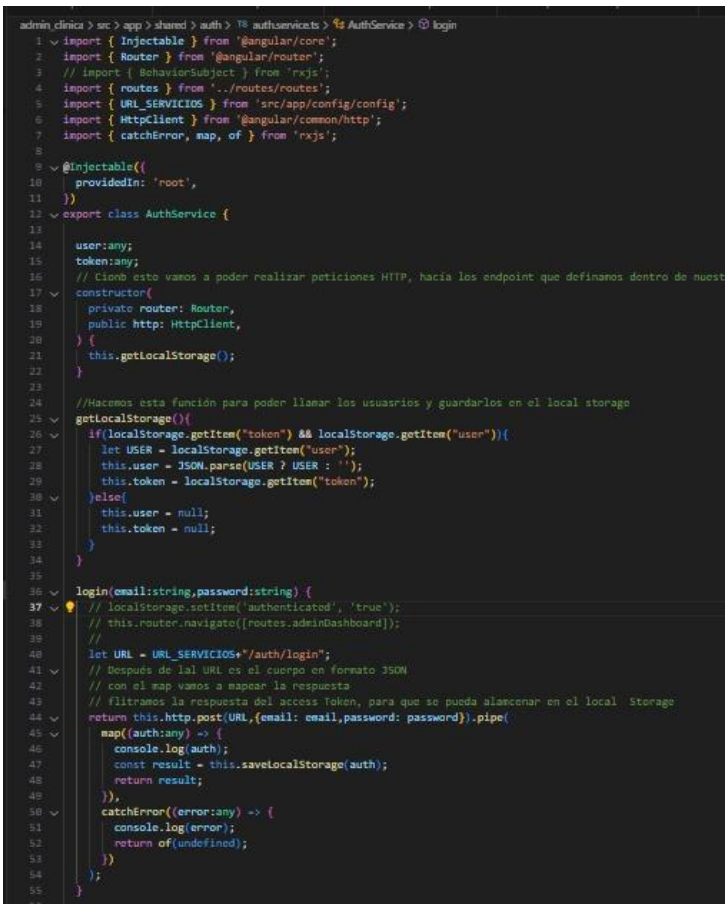
```
admin_clinica > src > app > medical > staff > list-staff-n > TS list-staff-n.component.ts > ...
export class ListStaffNComponent {
  private getTableData(): void {
    this.staffService.listUsers().subscribe((resp:any) => {
      this.role_generals = resp.users.data;
      this.getTableDataGeneral();
    })


  }
  isPermision(permission:string){
    if(this.user.roles.includes('Super-Admin')){
      return true;
    }
    if(this.user.permissions.includes(permission)){
      return true;
    }
    return false;
  }
  getTableDataGeneral() {
    this.usersList = [];
    this.serialNumberArray = [];

    this.role_generals.map((res: any, index: number) => {
      const serialNumber = index + 1;
      if (index >= this.skip && serialNumber <= this.limit) {

        this.usersList.push(res);
        this.serialNumberArray.push(serialNumber);
      }
    });
    this.dataSource = new MatTableDataSource<any>(this.usersList);
    this.calculateTotalPages(this.totalData, this.pageSize);
  }

  selectUser(rol:any){
    this.staff_selected = rol;
  }

  deleteUser(){

    this.staffService.deleteUser(this.staff_selected.id).subscribe((resp:any) => {
      console.log(resp);
      let INDEX = this.usersList.findIndex((item:any) => item.id == this.staff_selected.id);
      if(INDEX != -1){
        this.usersList.splice(INDEX,1);

        $('#delete_patient').hide();
        $("#delete_patient").removeClass("show");
        $(".modal-backdrop").remove();
        $("body").removeClass();
        $("body").removeAttr("style");

        this.staff_selected = null;
      }
    })
  }
  // eslint-disable-next-line @typescript-eslint/no-explicit-any
  public searchData(value: any): void {
```

```
admin_clinica > src > app > medical > staff > list-staff-n > TS list-staff-n.component.ts > ...
export class ListStaffNComponent {
  deleteUser(){
    this.staffService.deleteUser(this.staff_selected.id).subscribe((resp:any) => {
      }
    })
  }
  // eslint-disable-next-line @typescript-eslint/no-explicit-any
  public searchData(value: any): void {
    this.dataSource.filter = value.trim().toLowerCase();
    this.usersList = this.dataSource.filteredData;
  }

  public sortData(sort: any) {
    const data = this.userslist.slice();

    if (!sort.active || sort.direction === '') {
      this.usersList = data;
    } else {
      this.usersList = data.sort((a:any, b:any) => {
        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        const aValue = (a as any)[sort.active];
        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        const bValue = (b as any)[sort.active];
        return (aValue < bValue ? -1 : 1) * (sort.direction === 'asc' ? 1 : -1);
      });
    }
  }

  public getMoreData(event: string): void {
    if (event == 'next') {
      this.currentPage++;
      this.pageIndex = this.currentPage - 1;
      this.limit += this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableDataGeneral();
    } else if (event == 'previous') {
      this.currentPage--;
      this.pageIndex = this.currentPage - 1;
      this.limit -= this.pageSize;
      this.skip = this.pageSize * this.pageIndex;
      this.getTableDataGeneral();
    }
  }

  public moveToPage(pageNumber: number): void {
    this.currentPage = pageNumber;
    this.skip = this.pageSelection[pageNumber - 1].skip;
    this.limit = this.pageSelection[pageNumber - 1].limit;
    if (pageNumber > this.currentPage) {
      this.pageIndex = pageNumber - 1;
    } else if (pageNumber < this.currentPage) {
      this.pageIndex = pageNumber + 1;
    }
    this.getTableDataGeneral();
  }

  public PageSize(): void {
    this.pageSelection = [];
```

```
admin_clinica > src > app > medical > staff > list-staff-n > TS list-staff-n.component.ts > ...
  10  export class ListStaffNComponent {
 125    public getMoreData(event: string): void {
 127      this.currentPage++;
 128      this.pageIndex = this.currentPage - 1;
 129      this.limit += this.pageSize;
 130      this.skip = this.pageSize * this.pageIndex;
 131      this.getTableDataGeneral();
 132    } else if (event == 'previous') {
 133      this.currentPage--;
 134      this.pageIndex = this.currentPage - 1;
 135      this.limit -= this.pageSize;
 136      this.skip = this.pageSize * this.pageIndex;
 137      this.getTableDataGeneral();
 138    }
 139  }
 140
 141  public moveToPage(pageNumber: number): void {
 142    this.currentPage = pageNumber;
 143    this.skip = this.pageSelection[pageNumber - 1].skip;
 144    this.limit = this.pageSelection[pageNumber - 1].limit;
 145    if (pageNumber > this.currentPage) {
 146      this.pageIndex = pageNumber - 1;
 147    } else if (pageNumber < this.currentPage) {
 148      this.pageIndex = pageNumber + 1;
 149    }
 150    this.getTableDataGeneral();
 151  }
 152
 153  public PageSize(): void {
 154    this.pageSelection = [];
 155    this.limit = this.pageSize;
 156    this.skip = 0;
 157    this.currentPage = 1;
 158    this.searchDataValue = '';
 159    this.getTableDataGeneral();
 160  }
 161
 162  private calculateTotalPages(totalData: number, pageSize: number): void {
 163    this.pageNumberArray = [];
 164    this.totalPages = totalData / pageSize;
 165    if (this.totalPages % 1 != 0) {
 166      this.totalPages = Math.trunc(this.totalPages + 1);
 167    }
 168    /* eslint no-var: off */
 169    for (var i = 1; i <= this.totalPages; i++) {
 170      const limit = pageSize * i;
 171      const skip = limit - pageSize;
 172      this.pageNumberArray.push(i);
 173      this.pageSelection.push({ skip: skip, limit: limit });
 174      // 1
 175      // 0 - 10
 176      // 2
 177      // 10 - 20
 178    }
 179  }
 180
 181  }
```

**Anexo BB**

*Gestión de Autenticación con Angular: auth.service.ts*



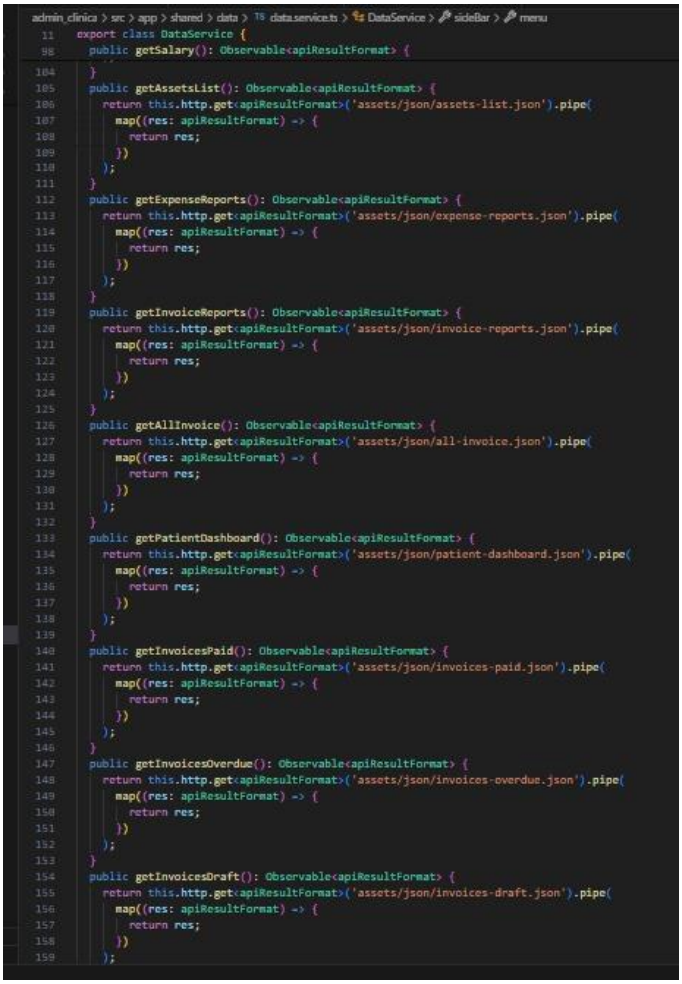
```
admin_clinica > src > app > shared > auth > TS auth.service.ts > AuthService > login
  1  import { Injectable } from '@angular/core';
  2  import { Router } from '@angular/router';
  3  // import { BehaviorSubject } from 'rxjs';
  4  import { routes } from '../routes/routes';
  5  import { URL_SERVICIOS } from 'src/app/config/config';
  6  import { HttpClient } from '@angular/common/http';
  7  import { catchError, map, of } from 'rxjs';
  8
  9  @Injectable({
 10    providedIn: 'root',
 11  })
 12  export class AuthService {
 13
 14    user:any;
 15    token:any;
 16    // Cionb esto vamos a poder realizar peticiones HTTP, hacia los endpoint que definamos dentro de nuest
 17    constructor(
 18      private router: Router,
 19      public http: HttpClient,
 20    ) {
 21      this.getLocalStorage();
 22    }
 23
 24    //Hacemos esta función para poder llamar los usuasrios y guardarlos en el local storage
 25    getLocalStorage(){
 26      if(localStorage.getItem("token") && localStorage.getItem("user")){
 27        let USER = localStorage.getItem("user");
 28        this.user = JSON.parse(USER ? USER : '');
 29        this.token = localStorage.getItem("token");
 30      }else{
 31        this.user = null;
 32        this.token = null;
 33      }
 34    }
 35
 36    login(email:string,password:string) {
 37      // localStorage.setItem('authenticated', 'true');
 38      // this.router.navigate([routes.adminDashboard]);
 39      //
 40      let URL = URL_SERVICIOS+"/auth/login";
 41      // Después de lal URL es el cuerpo en formato JSON
 42      // con el map vamos a mapear la respuesta
 43      // filtramos la respuesta del access Token, para que se pueda alamcenar en el local  Storage
 44      return this.http.post(URL,{email: email,password: password}).pipe(
 45        map((auth:any) => {
 46          console.log(auth);
 47          const result = this.saveLocalStorage(auth);
 48          return result;
 49        }),
 50        catchError((error:any) => {
 51          console.log(error);
 52          return of(undefined);
 53        })
 54      );
 55    }
```

```
saveLocalStorage(auth:any){
  if(auth && auth.access_token){
    localStorage.setItem("token",auth.access_token);
    localStorage.setItem("user",JSON.stringify(auth.user));
    //Aquí habilito la autenticación, ya que se está guardando en el Local Storage
    localStorage.setItem('authenticated', 'true');
    return true;
  }
  return false;
}

logout(){
  localStorage.removeItem("token");
  localStorage.removeItem("user");
  localStorage.removeItem('authenticated');
  this.router.navigate([routes.login]);
}
}
```

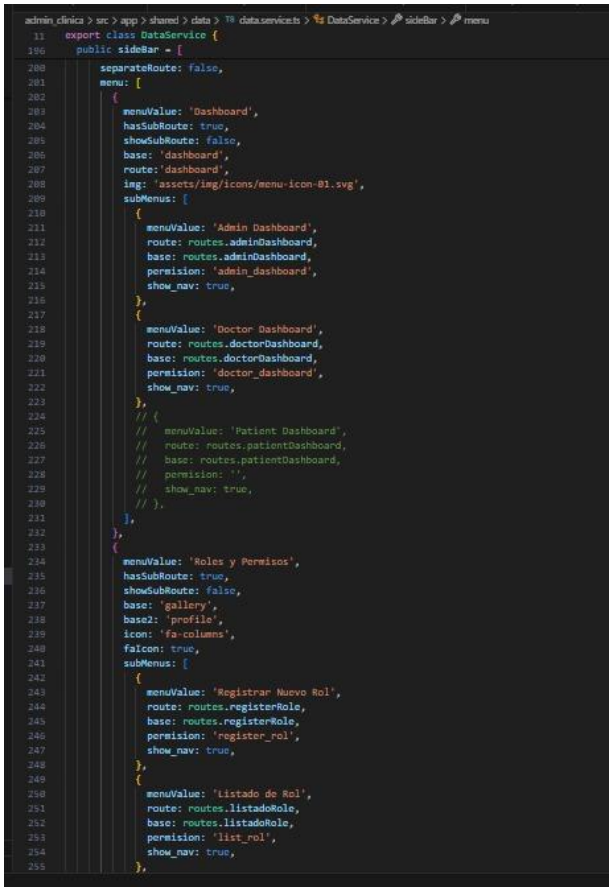
**Anexo CC**

*Servicio de Gestión de Datos en Angular: data.service.ts*

```
admin_clinica > src > app > shared > data > data.service.ts > DataService > sideBar > menu
export class DataService {
  public getSchedule(): Observable<apiResultFormat> {
    //
    );
  }
  public getInvoices(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/invoices.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getPayments(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/payments.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getExpenses(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/expenses.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getTaxes(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/taxes.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getProvidentFund(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/provident-fund.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getDepartmentList(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/department-list.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getSalary(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/salary.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getAssetsList(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/assets-list.json').pipe(
      map((res: apiResultFormat) => {
        return res;
```



```
admin_clinica > src > app > shared > data > data.service.ts > DataService > sideBar > menu
export class DataService {
  public getSalary(): Observable<apiResultFormat> {
  }
  public getAssetsList(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/assets-list.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getExpenseReports(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/expense-reports.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getInvoiceReports(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/invoice-reports.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getAllInvoice(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/all-invoice.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getPatientDashboard(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/patient-dashboard.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getInvoicesPaid(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/invoices-paid.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getInvoicesOverdue(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/invoices-overdue.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
  }
  public getInvoicesDraft(): Observable<apiResultFormat> {
    return this.http.get<apiResultFormat>('assets/json/invoices-draft.json').pipe(
      map((res: apiResultFormat) => {
        return res;
      })
    );
```



```
admin_clinica > src > app > shared > data > TS data.service.ts > DataService > sideBar > menu
export class DataService {
  public sideBar = [
    separateRoute: false,
    menu: [
      {
        menuValue: 'Dashboard',
        hasSubRoute: true,
        showSubRoute: false,
        base: 'dashboard',
        route:'dashboard',
        img: 'assets/img/icons/menu-icon-01.svg',
        subMenus: [
          {
            menuValue: 'Admin Dashboard',
            route: routes.adminDashboard,
            base: routes.adminDashboard,
            permision: 'admin_dashboard',
            show_nav: true,
          },
          {
            menuValue: 'Doctor Dashboard',
            route: routes.doctorDashboard,
            base: routes.doctorDashboard,
            permision: 'doctor_dashboard',
            show_nav: true,
          },
          // {
          //   menuValue: 'Patient Dashboard',
          //   route: routes.patientDashboard,
          //   base: routes.patientDashboard,
          //   permision: '',
          //   show_nav: true,
          // },
        ],
      },
      {
        menuValue: 'Roles y Permisos',
        hasSubRoute: true,
        showSubRoute: false,
        base: 'gallery',
        base2: 'profile',
        icon: 'fa-columns',
        faIcon: true,
        subMenus: [
          {
            menuValue: 'Registrar Nuevo Rol',
            route: routes.registerRole,
            base: routes.registerRole,
            permision: 'register_rol',
            show_nav: true,
          },
          {
            menuValue: 'Listado de Rol',
            route: routes.listadoRole,
            base: routes.listadoRole,
            permision: 'list_rol',
            show_nav: true,
          },
```

```
admin_clinica > src > app > shared > data > TS data.service.ts > DataService > sideBar > menu
export class DataService {
  public sideBar = [
    menu: [
      subMenus: [
        },
        {
          menuValue: 'Listado de Rol',
          route: routes.listadoRole,
          base: routes.listadoRole,
          permision: 'list_rol',
          show_nav: true,
        },
        {
          menuValue: 'Editar Rol',
          route: '',
          base: '',
          permision: 'edit_rol',
          show_nav: false,
        },
        {
          menuValue: 'Eliminar Rol',
          route: '',
          base: '',
          permision: 'delete_rol',
          show_nav: false,
        },
      ],
    },
    {
      menuValue: 'Personal',
      hasSubRoute: true,
      showSubRoute: false,
      base: 'staffs',
      img: 'assets/img/icons/menu-icon-08.svg',
      subMenus: [
        {
          menuValue: 'Listado de Personal',
          route: routes.staffList,
          base: routes.staffList,
          permision: 'list_staff',
          show_nav: true,
        },
        {
          menuValue: 'Añadir Personal',
          route: routes.addStaff,
          base: routes.addStaff,
          permision: 'register_staff',
          show_nav: true,
        },
        {
          menuValue: 'Editar Personal',
          route: '',
          base: '',
          permision: 'edit_staff',
          show_nav: false,
        },
        {
          menuValue: 'Eliminar Personal'
```

admin_clinica > src > app > shared > data > data.service.ts > DataService > sideBar > menu

```
export class DataService {
  public sideBar = [
    menu: [
      subMenus: [
        permision: 'edit_specialty',
        show_nav: false,
      },
      {
        menuValue: 'Eliminar Especialidades',
        route: '',
        base: '',
        permision: 'delete_specialty',
        show_nav: false,
      },
    ],
  },
  {
    menuValue: 'Doctores',
    hasSubRoute: true,
    showSubRoute: false,
    base: 'doctor',
    img: 'assets/img/icons/menu-icon-02.svg',
    subMenus: [
      {
        menuValue: 'Listado de Doctores',
        route: routes.doctorsList,
        base: routes.doctorsList,
        permision: 'list_doctor',
        show_nav: true,
      },
      {
        menuValue: 'Añadir Doctores',
        route: routes.addDoctor,
        base: routes.addDoctor,
        permision: 'register_doctor',
        show_nav: true,
      },

      {
        menuValue: 'Editar Doctores',
        route: '',
        base: '',
        permision: 'edit_doctor',
        show_nav: false,
      },
      {
        menuValue: 'Eliminar Doctores',
        route: '',
        base: '',
        permision: 'delete_doctor',
        show_nav: false,
      },
      {
        menuValue: 'Perfil del Doctores',
        route: '',
        base: '',
        permision: 'profile_doctor',
        show_nav: false,
```

admin_clinica > src > app > shared > data > data.service.ts > DataService > sideBar > menu

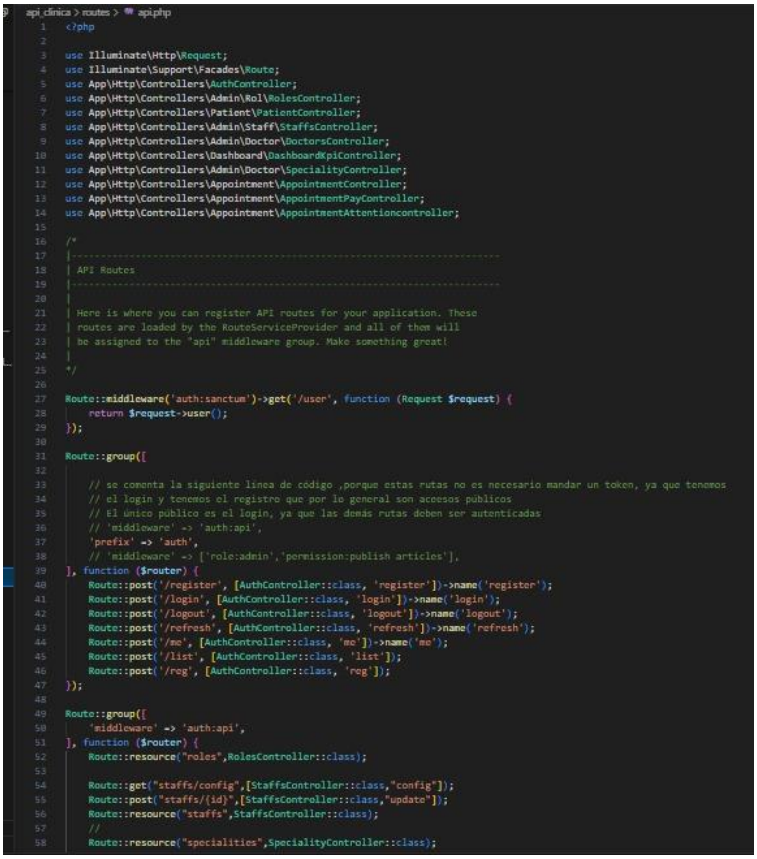
```
export class DataService {
  public sideBar = [
    menu: [
      subMenus: [
        show_nav: false,
      },
      {
        menuValue: 'Perfil del Doctores',
        route: '',
        base: '',
        permision: 'profile_doctor',
        show_nav: false,
      },
    ],
  },
  {
    menuValue: 'Pacientes',
    hasSubRoute: true,
    showSubRoute: false,
    base: 'patient',
    img: 'assets/img/icons/menu-icon-03.svg',
    subMenus: [
      {
        menuValue: 'Listado de Pacientes',
        route: routes.patientsList,
        base: routes.patientsList,
        permision: 'list_patient',
        show_nav: true,
      },
      {
        menuValue: 'Añadir Pacientes',
        route: routes.addPatient,
        base: routes.addPatient,
        permision: 'register_patient',
        show_nav: true,
      },
      {
        menuValue: 'Editar Patientes',
        route: '',
        base: '',
        permision: 'edit_patient',
        show_nav: false,
      },
      {
        menuValue: 'Eliminar Patientes',
        route: '',
        base: '',
        permision: 'delete_patient',
        show_nav: false,
      },
      {
        menuValue: 'Perfil del Paciente',
        route: '',
        base: '',
        permision: 'profile_patient',
        show_nav: false,
      },
    ],
```

```
admin_clinica > src > app > shared > data > data.service.ts > DataService > sideBar > menu
export class DataService {
  public sideBar = [
      menu: [
        subMenus: [
          {
            menuValue: 'Perfil del Paciente',
            route: '',
            base: '',
            permision: 'profile_patient',
            show_nav: false,
          },
        ],
      },
      {
        menuValue: 'Citas',
        hasSubRoute: true,
        showSubRoute: false,
        base: 'appointments',
        img: 'assets/img/icons/menu-icon-04.svg',
        subMenus: [
          {
            menuValue: 'Listado de Citas',
            route: routes.appointmentList,
            base: routes.appointmentList,
            permision: 'list_appointment',
            show_nav: true,
          },
          {
            menuValue: 'Añadir Citas',
            route: routes.addAppointment,
            base: routes.addAppointment,
            permision: 'register_appointment',
            show_nav: true,
          },
          {
            menuValue: 'Editar Citas',
            route: '',
            base: '',
            permision: 'edit_appointment',
            show_nav: false,
          },
          {
            menuValue: 'Eliminar Citas',
            route: '',
            base: '',
            permision: 'delete_appointment',
            show_nav: false,
          },
          {
            menuValue: 'Atención de Citas',
            route: '',
            base: '',
            permision: 'attention_appointment',
            show_nav: false,
          },
        ],
      },
      // {
```

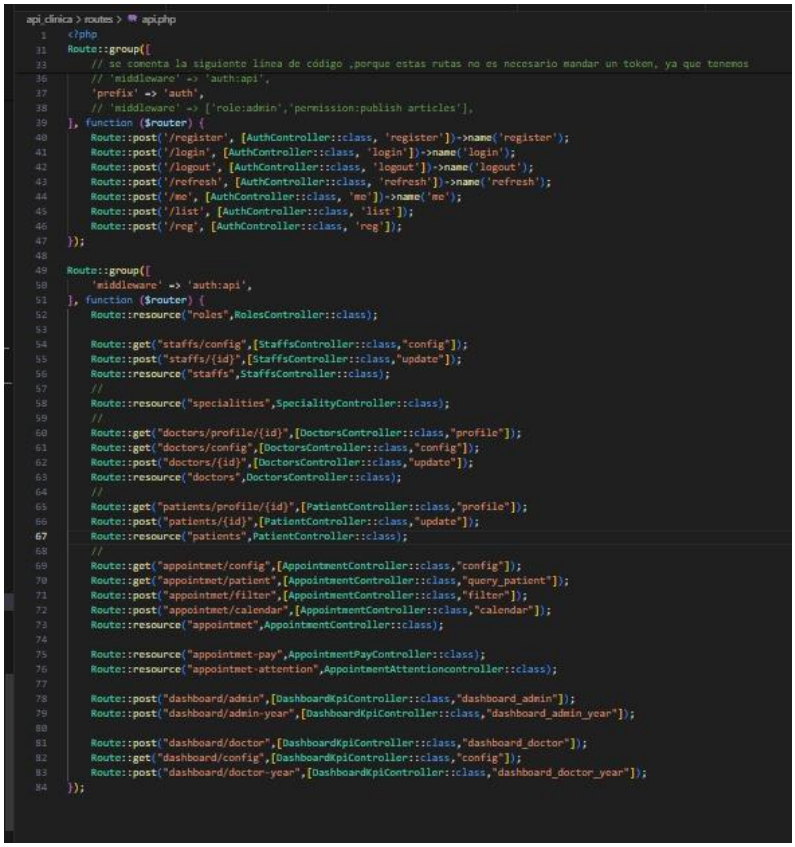
```
    // },
    {
      menuValue: 'Calendario',
      route: routes.calendar,
      hasSubRoute: false,
      showSubRoute: false,
      icon: 'fa-calendar',
      faIcon: true,
      base: 'calendar',
      permision: 'calendar',
      show_nav: true,
      subMenus: [],
    },

    ],
  },
];
// public sideBarList = [
```

**Anexo DD**

*Punto de Entrada para API en Laravel: api.php*



```
api_clinica > routes > api.php
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\AuthController;
use App\Http\Controllers\Admin\Rol\RolesController;
use App\Http\Controllers\Patient\PatientController;
use App\Http\Controllers\Admin\Staff\StaffsController;
use App\Http\Controllers\Admin\Doctor\DoctorsController;
use App\Http\Controllers\Dashboard\DashboardKpiController;
use App\Http\Controllers\Admin\Doctor\SpecialityController;
use App\Http\Controllers\Appointment\AppointmentController;
use App\Http\Controllers\Appointment\AppointmentPayController;
use App\Http\Controllers\Appointment\AppointmentAttentioncontroller;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});

Route::group([

    // se comenta la siguiente linea de código ,porque estas rutas no es necesario mandar un token, ya que tenemos
    // el login y tenemos el registro que por lo general son accesos públicos
    // El único público es el login, ya que las demás rutas deben ser autenticadas
    // 'middleware' => 'auth:api',
    'prefix' => 'auth',
    // 'middleware' => ['role:admin','permission:publish articles'],
], function ($router) {
    Route::post('/register', [AuthController::class, 'register'])->name('register');
    Route::post('/login', [AuthController::class, 'login'])->name('login');
    Route::post('/logout', [AuthController::class, 'logout'])->name('logout');
    Route::post('/refresh', [AuthController::class, 'refresh'])->name('refresh');
    Route::post('/me', [AuthController::class, 'me'])->name('me');
    Route::post('/list', [AuthController::class, 'list']);
    Route::post('/reg', [AuthController::class, 'reg']);
});

Route::group([
    'middleware' => 'auth:api',
], function ($router) {
    Route::resource("roles",RolesController::class);

    Route::get("staffs/config",[StaffsController::class,"config"]);
    Route::post("staffs/{id}",[StaffsController::class,"update"]);
    Route::resource("staffs",StaffsController::class);
    //
    Route::resource("specialities",SpecialityController::class);
```



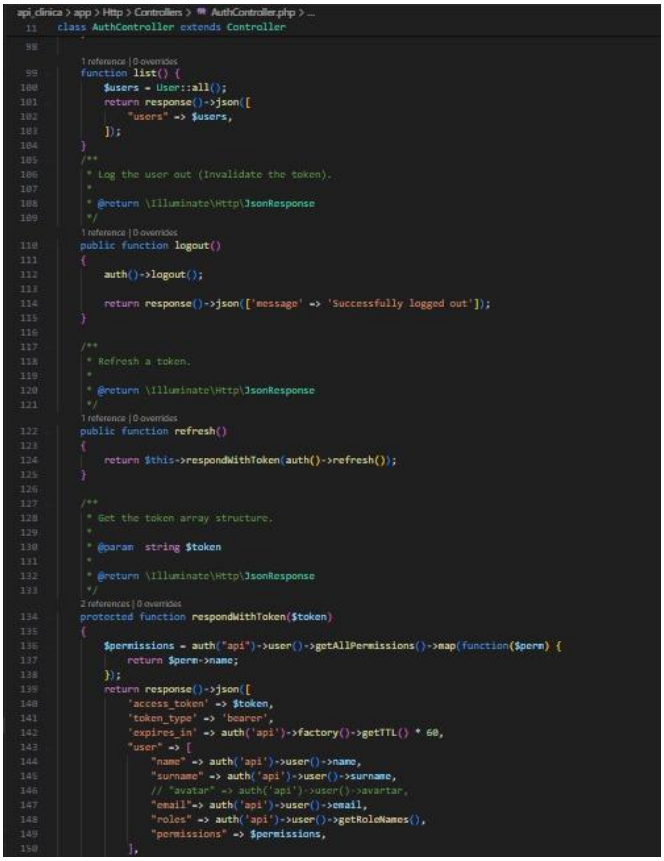
```
api_clinica > routes > api.php
<?php
Route::group([
    // se comenta la siguiente linea de código ,porque estas rutas no es necesario mandar un token, ya que tenemos
    // 'middleware' => 'auth:api',
    'prefix' => 'auth',
    // 'middleware' => ['role:admin','permission:publish articles'],
], function ($router) {
    Route::post('/register', [AuthController::class, 'register'])->name('register');
    Route::post('/login', [AuthController::class, 'login'])->name('login');
    Route::post('/logout', [AuthController::class, 'logout'])->name('logout');
    Route::post('/refresh', [AuthController::class, 'refresh'])->name('refresh');
    Route::post('/me', [AuthController::class, 'me'])->name('me');
    Route::post('/list', [AuthController::class, 'list']);
    Route::post('/reg', [AuthController::class, 'reg']);
});

Route::group([
    'middleware' => 'auth:api',
], function ($router) {
    Route::resource("roles",RolesController::class);

    Route::get("staffs/config",[StaffsController::class,"config"]);
    Route::post("staffs/{id}",[StaffsController::class,"update"]);
    Route::resource("staffs",StaffsController::class);
    //
    Route::resource("specialities",SpecialityController::class);
    //
    Route::get("doctors/profile/{id}",[DoctorsController::class,"profile"]);
    Route::get("doctors/config",[DoctorsController::class,"config"]);
    Route::post("doctors/{id}",[DoctorsController::class,"update"]);
    Route::resource("doctors",DoctorsController::class);
    //
    Route::get("patients/profile/{id}",[PatientController::class,"profile"]);
    Route::post("patients/{id}",[PatientController::class,"update"]);
    Route::resource("patients",PatientController::class);
    //
    Route::get("appointmet/config",[AppointmentController::class,"config"]);
    Route::get("appointmet/patient",[AppointmentController::class,"query_patient"]);
    Route::post("appointmet/filter",[AppointmentController::class,"filter"]);
    Route::post("appointmet/calendar",[AppointmentController::class,"calendar"]);
    Route::resource("appointmet",AppointmentController::class);

    Route::resource("appointmet-pay",AppointmentPayController::class);
    Route::resource("appointmet-attention",AppointmentAttentioncontroller::class);

    Route::post("dashboard/admin",[DashboardKpiController::class,"dashboard_admin"]);
    Route::post("dashboard/admin-year",[DashboardKpiController::class,"dashboard_admin_year"]);

    Route::post("dashboard/doctor",[DashboardKpiController::class,"dashboard_doctor"]);
    Route::get("dashboard/config",[DashboardKpiController::class,"config"]);
    Route::post("dashboard/doctor-year",[DashboardKpiController::class,"dashboard_doctor_year"]);
});
```

**Anexo EE**

*Controlador de Autenticación en Laravel: AuthController.php*

```
api_clinica > app > Http > Controllers > AuthController.php > ...
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\Auth;
use App\Http\Controllers\Controller;
use App\Models\User;
use Validator;


8 references | 0 implementations
class AuthController extends Controller
{
    /**
     * Create a new AuthController instance.
     *
     * @return void
     */
    0 references | 0 overrides
    public function __construct()
    {
        $this->middleware('auth:api', ['except' => ['login','register']]);
    }


    /**
     * Register a User.
     *
     * @return \Illuminate\Http\JsonResponse
     */
    1 reference | 0 overrides
    public function register() {

        $validator = Validator::make(request()->all(), [
            'name' => 'required',
            'email' => 'required|email|unique:users',
            'password' => 'required|min:8',
        ]);

        if($validator->fails()){
            return response()->json($validator->errors()->toJson(), 400);
        }

        $user = new User;
        $user->name = request()->name;
        $user->email = request()->email;
        $user->password = bcrypt(request()->password);
        $user->save();

        return response()->json($user, 201);
    }

    1 reference | 0 overrides
    public function reg() {
    $this->authorize('create', User::class);

    $validator = Validator::make(request()->all(), [
        'name' => 'required',
        'email' => 'required|email|unique:users',
```

```
api_clinica > app > Http > Controllers > AuthController.php > ...
class AuthController extends Controller
    1 reference | 0 overrides
    public function reg() {
    $this->authorize('create', User::class);

    $validator = Validator::make(request()->all(), [
        'name' => 'required',
        'email' => 'required|email|unique:users',
        'password' => 'required|min:8',
    ]);

    if ($validator->fails()) {
        return response()->json($validator->errors()->toJson(), 400);
    }

    $user = new User;
    $user->name = request()->name;
    $user->email = request()->email;
    $user->password = bcrypt(request()->password);
    $user->save();

    return response()->json($user, 201);
}


    /**
     * Get a JWT via given credentials.
     *
     * @return \Illuminate\Http\JsonResponse
     */
    1 reference | 0 overrides
    public function login()
    {
        $credentials = request(['email', 'password']);

        if (! $token = auth('api')->attempt($credentials)) {
            return response()->json(['error' => 'Unauthorized'], 401);
        }

        return $this->respondWithToken($token);
    }

    /**
     * Get the authenticated User.
     *
     * @return \Illuminate\Http\JsonResponse
     */
    1 reference | 0 overrides
    public function me()
    {
        return response()->json(auth('api')->user());
    }

    1 reference | 0 overrides
    function list() {
        $users = User::all();
        return response()->json([
            "users" => $users,
```



```
api_clinica > app > Http > Controllers > AuthController.php > ...
class AuthController extends Controller
{

    1 reference | 0 overrides
    function list() {
        $users = User::all();
        return response()->json([
            "users" => $users,
        ]);
    }
    /**
     * Log the user out (Invalidate the token).
     *
     * @return \Illuminate\Http\JsonResponse
     */
    1 reference | 0 overrides
    public function logout()
    {
        auth()->logout();

        return response()->json(['message' => 'Successfully logged out']);
    }

    /**
     * Refresh a token.
     *
     * @return \Illuminate\Http\JsonResponse
     */
    1 reference | 0 overrides
    public function refresh()
    {
        return $this->respondWithToken(auth()->refresh());
    }

    /**
     * Get the token array structure.
     *
     * @param  string $token
     *
     * @return \Illuminate\Http\JsonResponse
     */
    2 references | 0 overrides
    protected function respondWithToken($token)
    {
        $permissions = auth("api")->user()->getAllPermissions()->map(function($perm) {
            return $perm->name;
        });
        return response()->json([
            'access_token' => $token,
            'token_type' => 'bearer',
            'expires_in' => auth('api')->factory()->getTTL() * 60,
            "user" => [
                "name" => auth('api')->user()->name,
                "surname" => auth('api')->user()->surname,
                // "avatar" => auth('api')->user()->avartar,
                "email"=> auth('api')->user()->email,
                "roles" => auth('api')->user()->getRoleNames(),
                "permissions" => $permissions,
            ],
```

# BIBLIOGRAFÍA

Angular Team. (n.d.). What is Angular? https://angular.io/guide/what-is-angular.

Arsys. (2023). Optimizar base de datos MySQL: guía práctica y fácil. Recuperado de https://www.arsys.es

Avilés Matute, S., Avila-Pesantez, D., & Avila, L. M. (2020). Desarrollo de sistema Web basado en los frameworks de Laravel y VueJs, para la gestión por procesos: Un estudio de caso. Revista Peruana de Computación y Sistemas, 3(2), 3-10. http://dx.doi.org/10.15381/rpcs.v3i2.19256.

Bautista-Villegas, E. (2022). Metodologías ágiles XP y Scrum, empleadas para el desarrollo de páginas web, bajo MVC, con lenguaje PHP y framework Laravel. Revista Amazonía Digital, 1(1), e168. https://doi.org/10.55873/rad.v1i1.168.

Bautista-Villegas, E. (2022). Metodologías ágiles XP y Scrum, empleadas para el desarrollo de páginas web, bajo MVC, con lenguaje PHP y framework Laravel. Revista Amazonía Digital, 1(1), e168. https://doi.org/10.55873/rad.v1i1.168.

Bean, M. (2015). Laravel 5 Essentials. Packt Publishing.

Ciclo de vida scrum. (s. f.). ResearchGate. https://www.researchgate.net/figure/Figura-27-Ciclo-de-Vida-Scrum_fig3_318280962

EcuRed. (n. d.). XAMPP - ECURed. https://www.ecured.cu/XAMPP

Haro, E., Guarda, T., Zambrano Peñaherrera, A. O., & Ninahualpa Quiña, G. (2019). Desarrollo backend para aplicaciones web, Servicios Web Restful: Node.js vs Spring Boot. RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação, E17, 309-321.

Korva, J. (2016). Developing a web application with Angular 2. Oulu University of Applied Sciences.

Lei, K., Ma, Y., & Tan, Z. (2014). Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js. 2014 IEEE 17th International Conference on Computational Science and Engineering. DOI: 10.1109/CSE.2014.142.

López Gil, A. (2018). Estudio comparativo de metodologías tradicionales y ágiles para proyectos de Desarrollo de Software. Universidad de Valladolid, Escuela de Ingenierías Industriales.

Mariño, S. I., & Alfonzo, P. L. (2014). Implementación de SCRUM en el diseño del proyecto del Trabajo Final de Aplicación. Scientia Et Technica, 19(4), 413-418.

Molina Montero, B., Vite Cevallos, H., & Dávila Cuesta, J. (2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. Espirales revista multidisciplinaria de investigación, enviado marzo 2018 – revisado abril 2018 – publicado junio 2018.

ProximaHost. (2023). MySQL, el sistema para gestión de bases de datos más popular. Recuperado de https://proximahost.es

Rodríguez, C., & Dorado, R. (n.d.). ¿Por qué implementar Scrum? Ontare, 125-144.

Schwaber, K., & Sutherland, J. (2020). La Guía Scrum: La Guía Definitiva de Scrum: Las Reglas del Juego. Scrum Guides.

Sunardi, A., & Suharjito. (2019). MVC Architecture: A Comparative Study Between Laravel Framework and Slim Framework in Freelancer Project Monitoring System Web Based. Procedia Computer Science, 157, 134–141. https://doi.org/10.1016/j.procs.2019.08.150.