

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 – INTRODUCCIÓN	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 – INTRODUCCIÓN

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 Título
- 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).

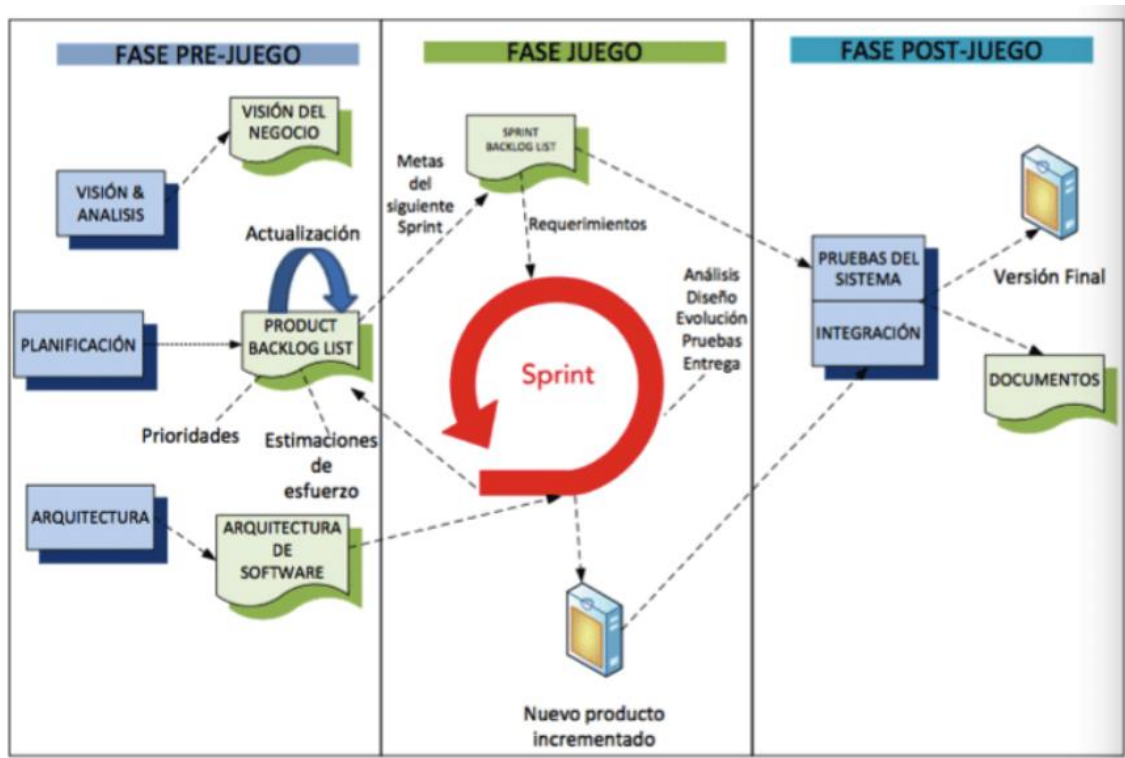
A diferencia de los métodos tradicionales, las metodologías ágiles dividen los proyectos en partes más pequeñas, lo que permite entregas más frecuentes y revisiones constantes. Esto no solo mejora la satisfacción del cliente mostrando progreso de forma continua, sino que también facilita la gestión de 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

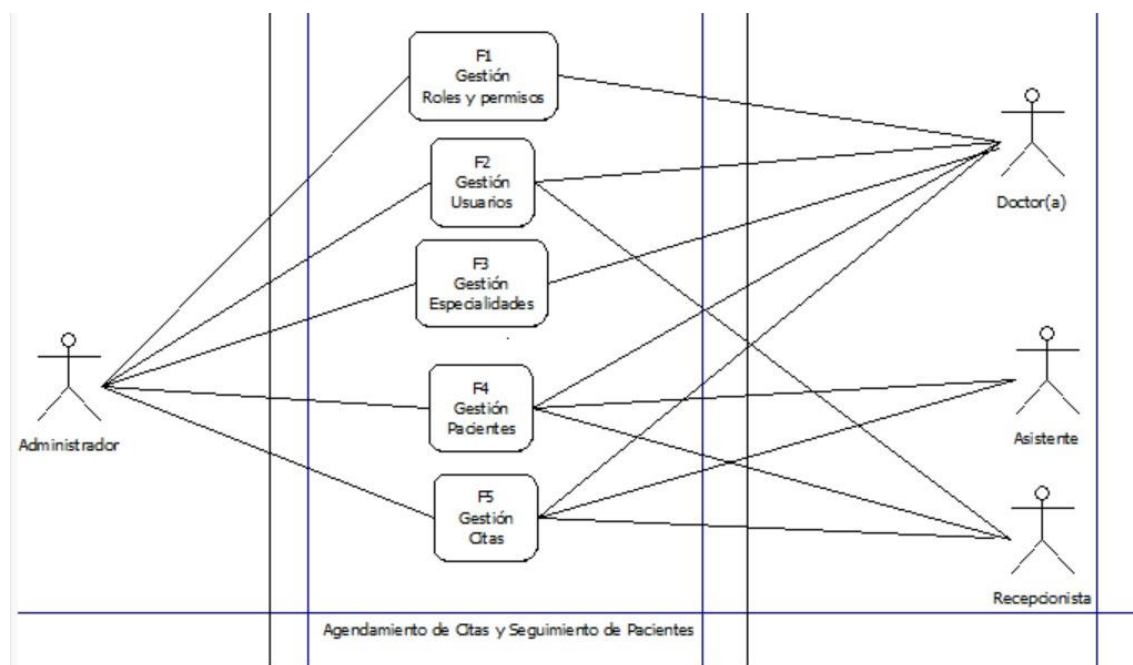
Tras revisar el Product Backlog, procederemos a explorar los casos de uso que describen la implementación práctica de estas funcionalidades, asegurando que el sistema cumpla con las necesidades operativas y de seguridad de la clínica.

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

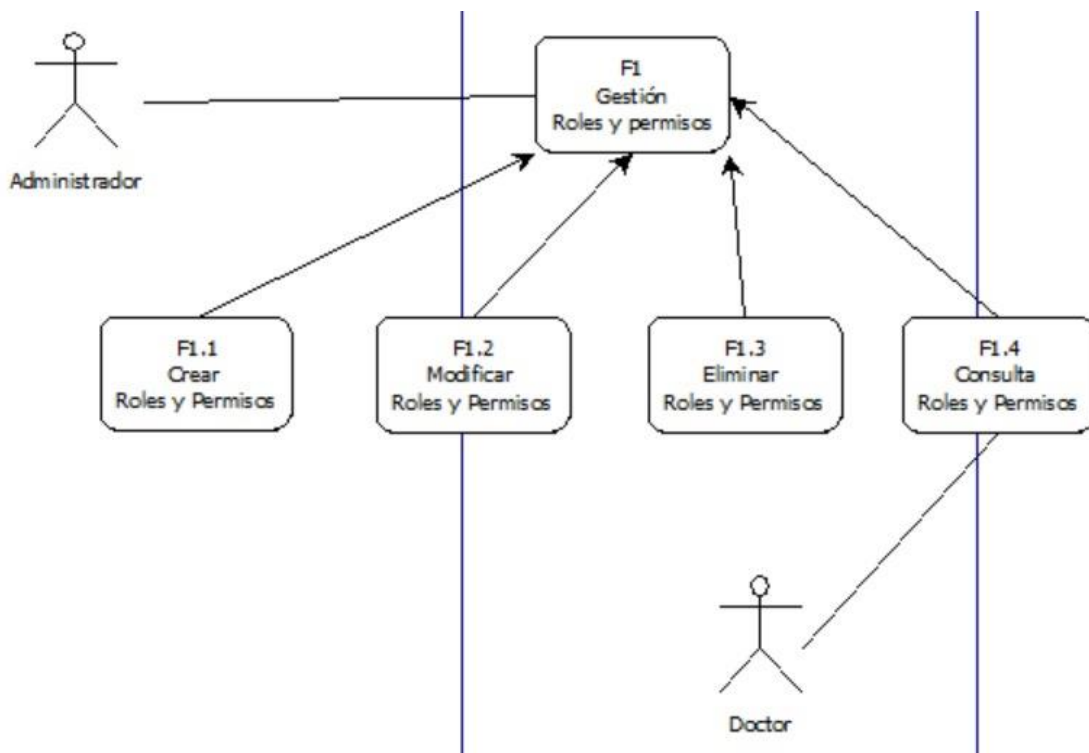


Nota. Diagrama de los Casos de Usos con las interacciones que tiene cada actor

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.

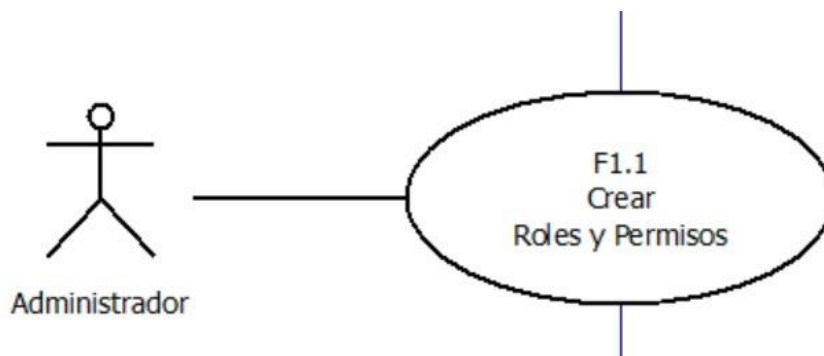
Actores: Administrador y Doctor

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.

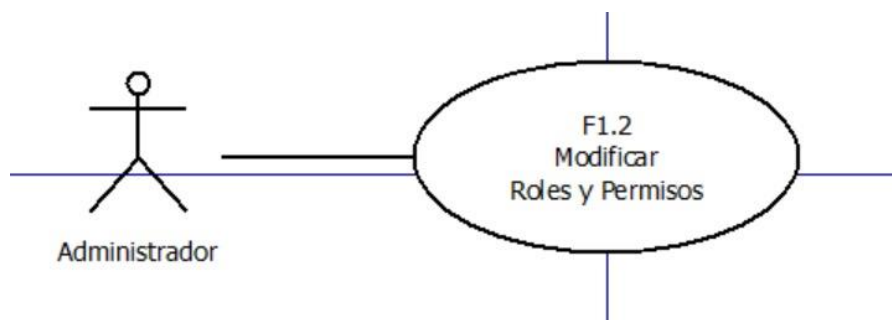
Excepciones:**Tabla 4***Excepciones en el proceso de Crear Roles y Permisos*

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 Alternativo:

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

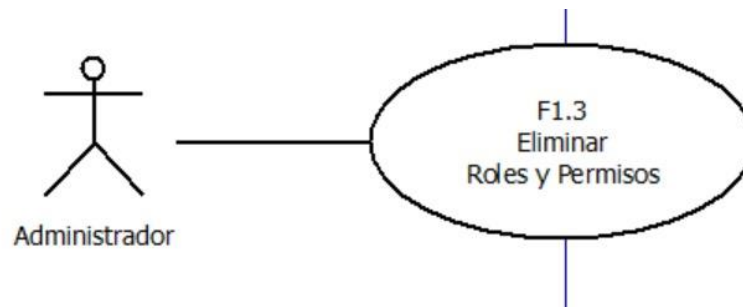
Flujo Alternativo:

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 Administrador

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.

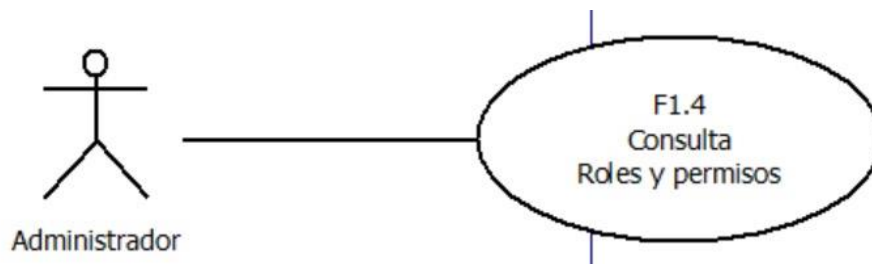
Excepciones:**Tabla 6***Excepciones en el proceso de Eliminar Roles y Permisos*

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 Alternativo:

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.

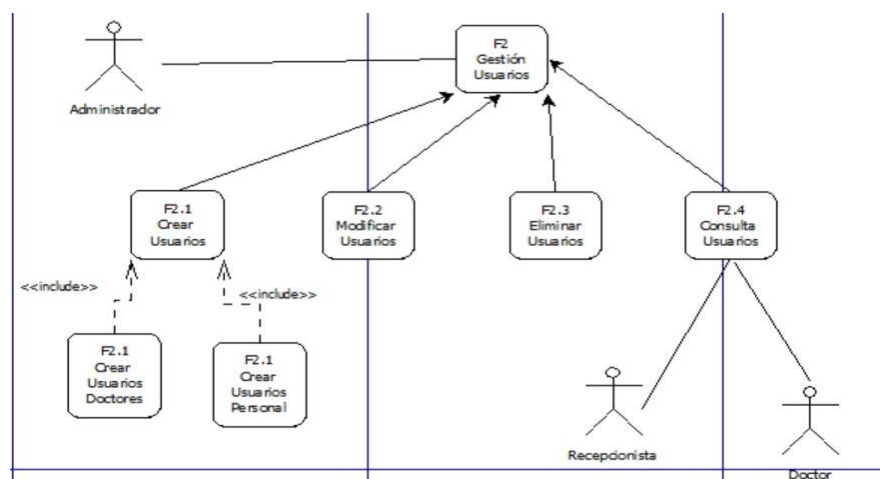
Flujo Alternativo:

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.

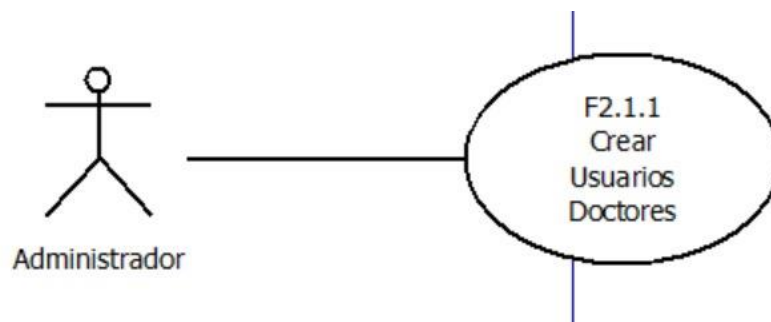
Actores: Administrador, Recepcionista y Doctor

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.

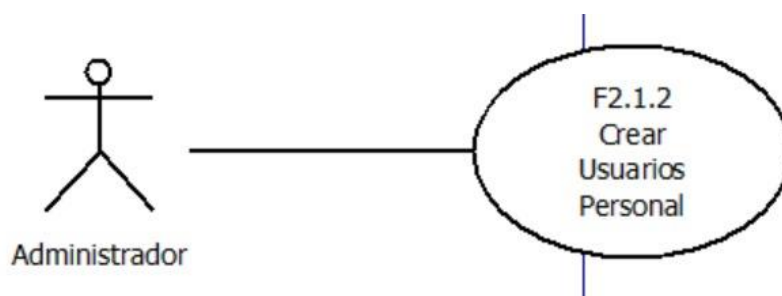
Flujo Alternativo:

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

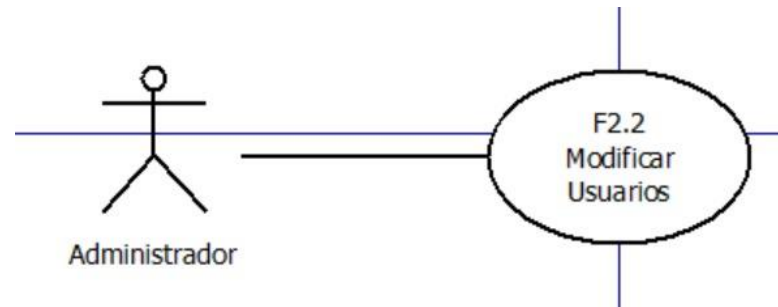
Flujo Alternativo:

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.

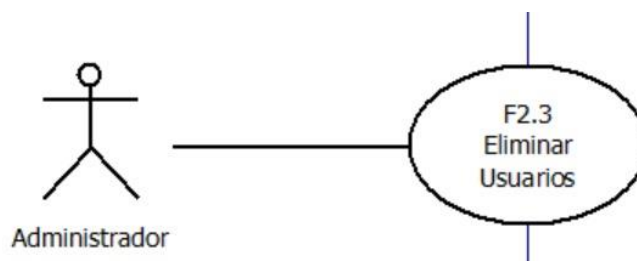
Nota. Tabla de excepciones al Modificar un Usuario con sus causas y soluciones

Flujo Alternativo: 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.

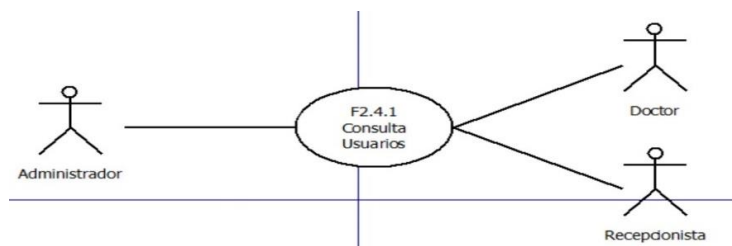
Flujo Alternativo:

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.

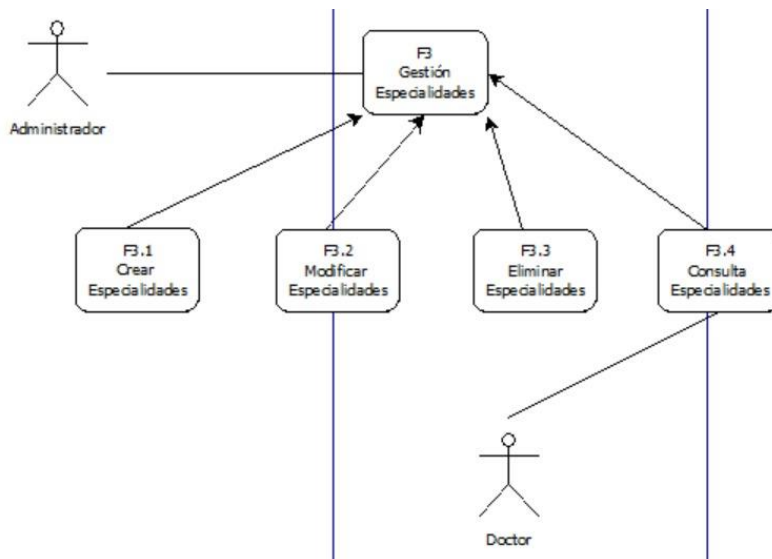
Flujo Alternativo:

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.

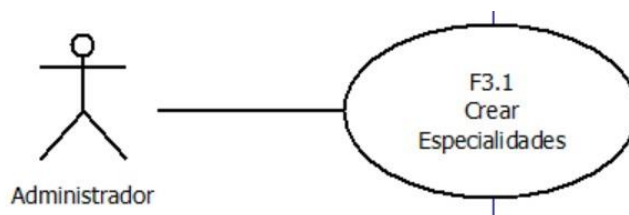
Descripción: Crear un sistema para gestionar las especialidades médicas dentro de la clínica.

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.

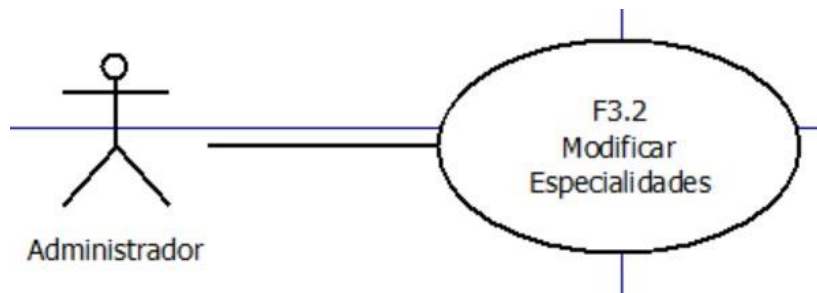
Flujo Alternativo:

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 puede 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.

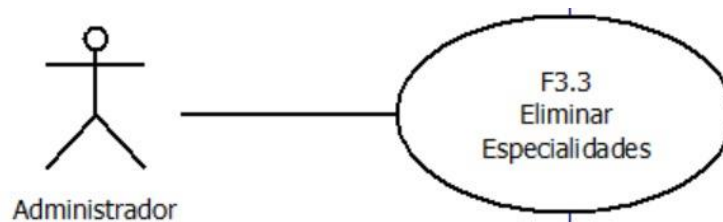
Excepciones:**Tabla 14***Excepciones en el proceso de Modificar Especialidades*

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 Alternativo:

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 Alternativo:

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 Alternativo:

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

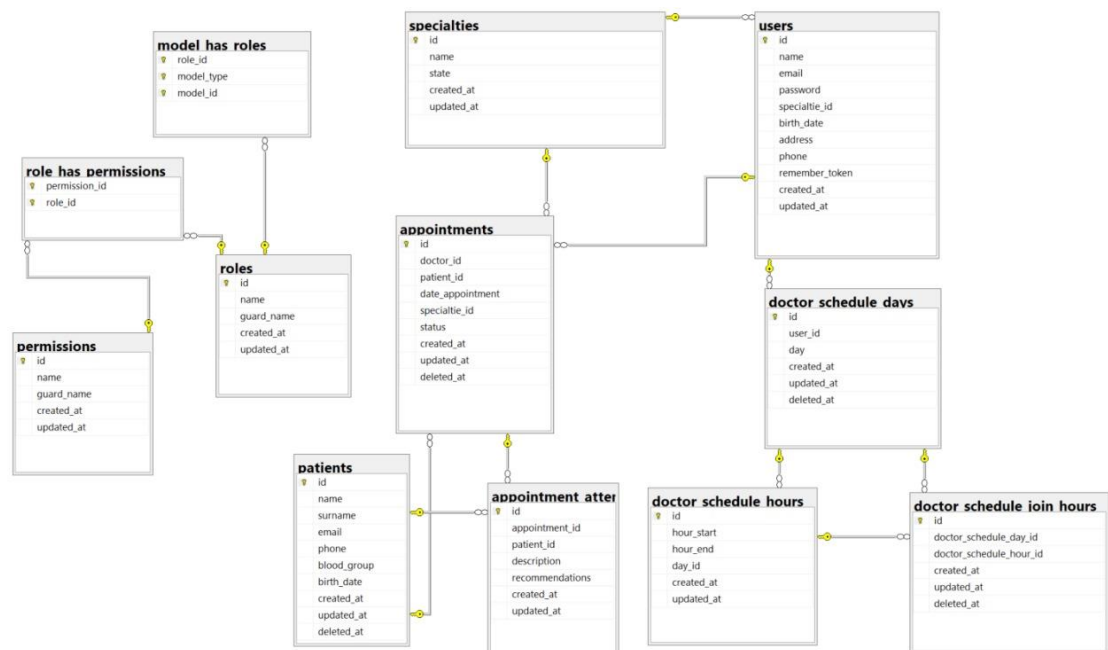
El diseño en el Sprint 1 establece la arquitectura de la base de datos y las interfaces de usuario necesarias para soportar la autenticación de usuarios, gestión de roles y permisos, manejo de la información del personal médico y administrativo, y la organización de especialidades médicas. Este diseño 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_rols 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

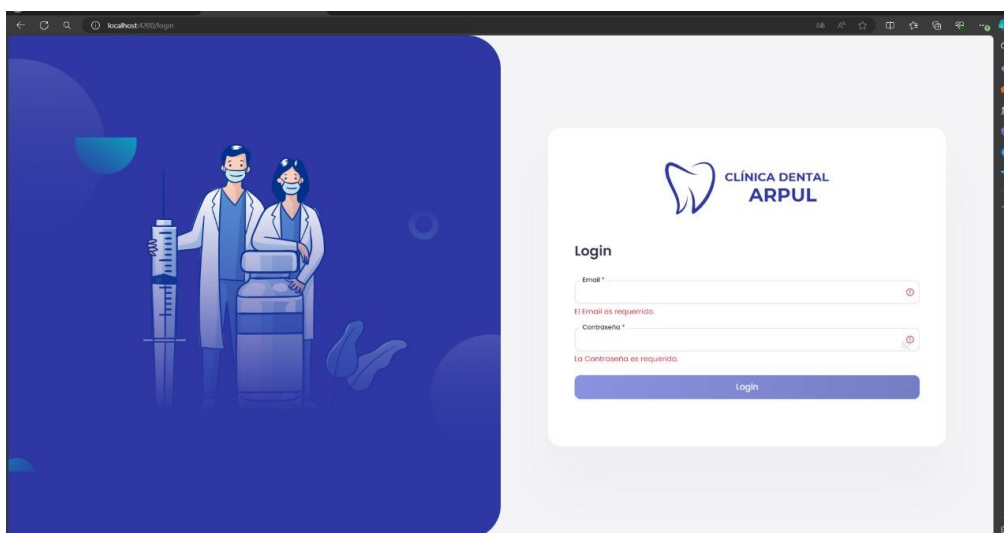
Las interfaces de usuario están diseñadas para facilitar la interacción eficiente con el sistema, proporcionando accesibilidad y seguridad en la gestión de datos y operaciones del sistema. Se describen a continuación las interfaces específicas para cada funcionalidad relevante del Sprint 1:

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

Sección	Permisos
Dashboard	<input type="checkbox"/> Admin Dashboard <input type="checkbox"/> Doctor Dashboard
Roles y Permisos	<input type="checkbox"/> Registrar Nuevo Rol <input type="checkbox"/> Listado de Rol <input type="checkbox"/> Editar Rol <input type="checkbox"/> Eliminar Rol
Personal	<input type="checkbox"/> Listado de Personal <input type="checkbox"/> Añadir Personal <input type="checkbox"/> Editar Personal <input type="checkbox"/> Eliminar Personal
Especialidades	<input type="checkbox"/> Listado de Especialidades <input type="checkbox"/> Añadir Especialidad <input type="checkbox"/> Editar Especialidades <input type="checkbox"/> Eliminar Especialidades
Doctores	<input type="checkbox"/> Listado de Doctores <input type="checkbox"/> Añadir Doctores <input type="checkbox"/> Editar Doctores <input type="checkbox"/> Eliminar Doctores <input type="checkbox"/> Perfil del Doctores

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

Nombre	Permisos	Fecha
DOCTOR PEDIATRA	register_patient,list_patient,edit_patient,delete_patient,profile_patient	2020-01-01
ENFERMERO	register_patient,list_patient,edit_patient,delete_patient,profile_patient,register_appointment,list_appointment,edit_appointment,delete_appointment	2020-01-01
DOCTOR	list_roledir_rol,list_patient,edit_patient,profile_patient,register_appointment,list_appointment,edit_appointment,register_specialty,list_specialty,edit_specialty,delete_specialty,calendar	2020-01-01
Super-Admin	TODOS LOS PERMISOS	2020-01-01

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

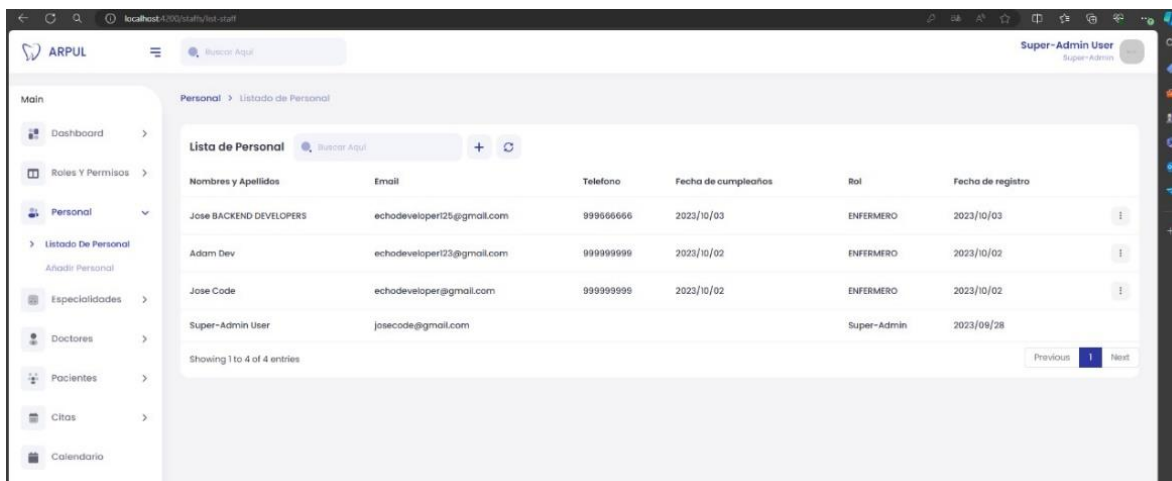
La **Figura 22** presenta la pestaña de listado de roles con sus permisos correspondientes.

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



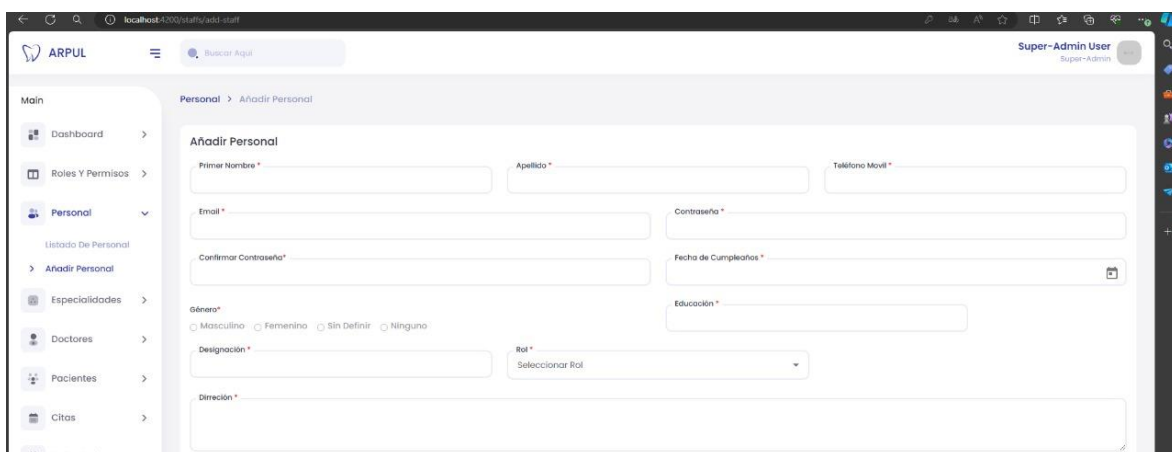
Nombres y Apellidos	Email	Telefono	Fecha de cumpleaños	Rol	Fecha de registro
Jose BACKEND DEVELOPERS	echodeveloper125@gmail.com	999666666	2023/10/03	ENFERMERO	2023/10/03
Adam Dev	echodeveloper123@gmail.com	999999999	2023/10/02	ENFERMERO	2023/10/02
Jose Code	echodeveloper@gmail.com	999999999	2023/10/02	ENFERMERO	2023/10/02
Super-Admin User	josecode@gmail.com			Super-Admin	2023/09/28

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

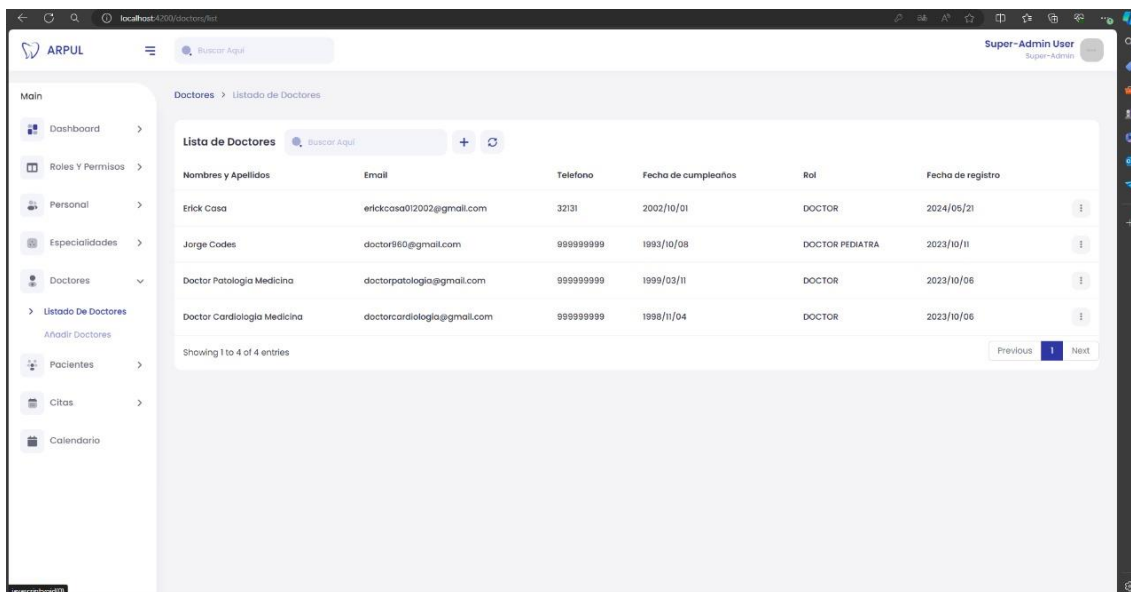


Nota. Pestaña que muestra cómo se Registra un Usuario de Personal.

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

Nombre	Estado	Fecha de registro
Ginecología y Obstetricia	Inactivo	2023-10-04 02:23:57
Gastroenterología	Activo	2023-10-04 02:23:28
Dermatología	Activo	2023-10-04 02:23:21
Cirugía Pediátrica	Activo	2023-10-04 02:23:09
Cardiología Intervencionista	Activo	2023-10-04 02:23:05
Anatomía Patológica	Activo	2023-10-04 02:22:58
Anestesiología	Activo	2023-10-04 02:18:43

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.

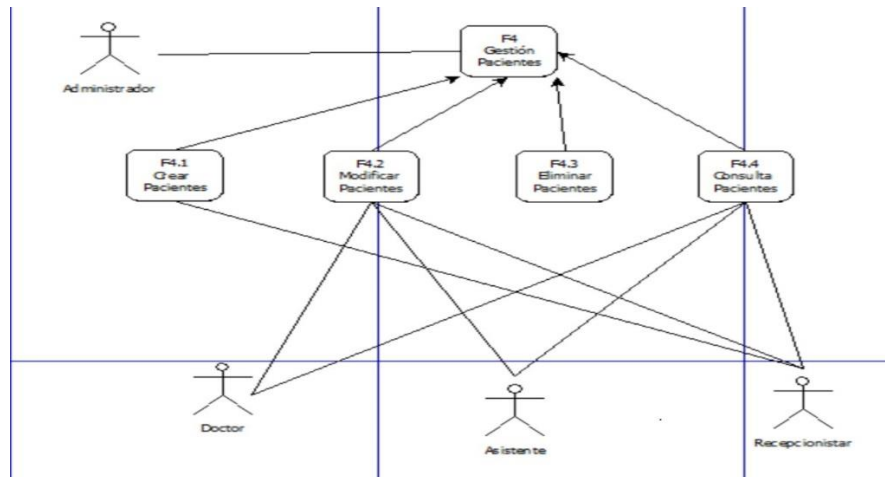
Tras revisar el Product Backlog, procederemos a explorar los casos de uso que describen la implementación práctica de estas funcionalidades, asegurando que el sistema cumpla con las necesidades operativas y de seguridad de la clínica.

Detalle de Casos de Uso

Caso de Uso: Gestión de Pacientes

Figura 34

Diagrama de Gestión de Pacientes



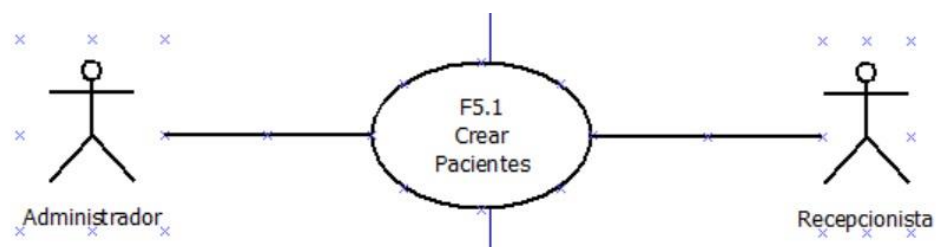
Nota. Diagrama detallado sobre la primera función de Gestión de Pacientes.

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.

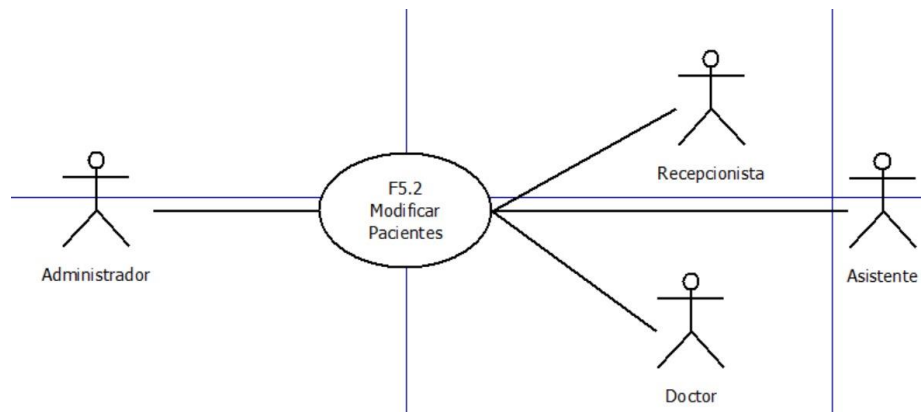
Flujo Alternativo:

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.

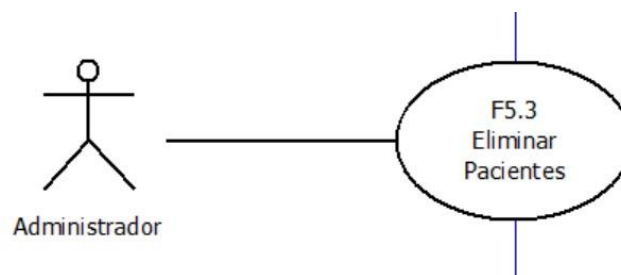
Excepciones:**Tabla 19***Excepciones en el Proceso de Modificar Pacientes*

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 Alternativo:

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.

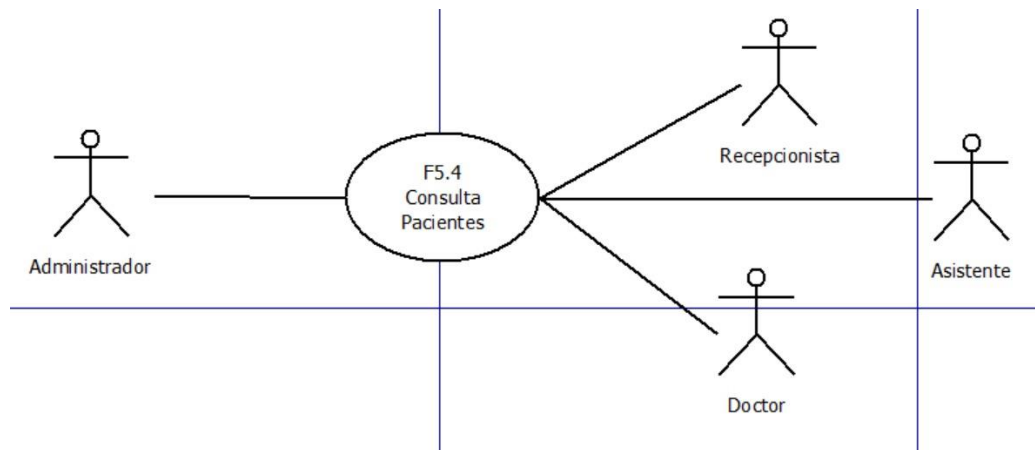
Flujo Alternativo:

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.

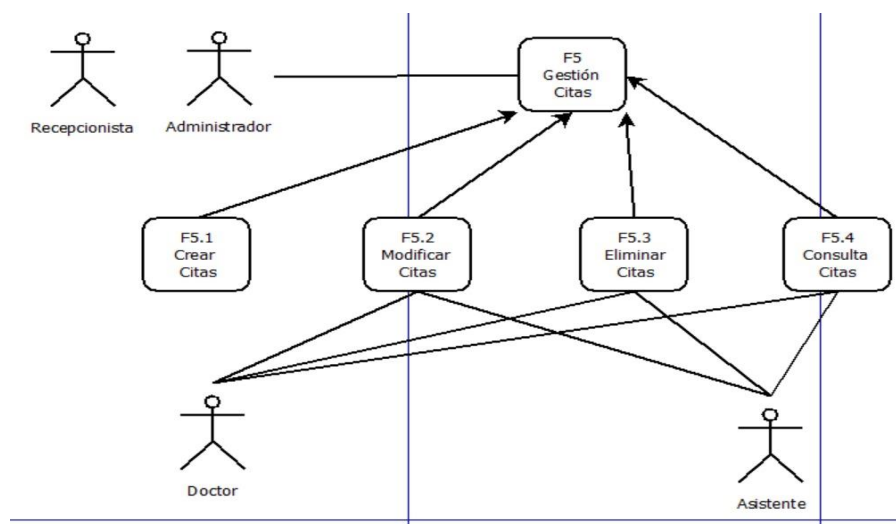
Flujo Alternativo:

- 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



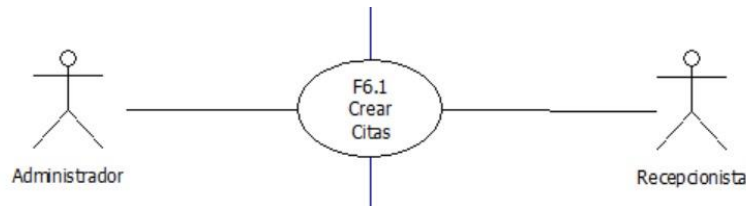
Nota. Diagrama detallado sobre la función de Gestión de Citas Médicas.

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.

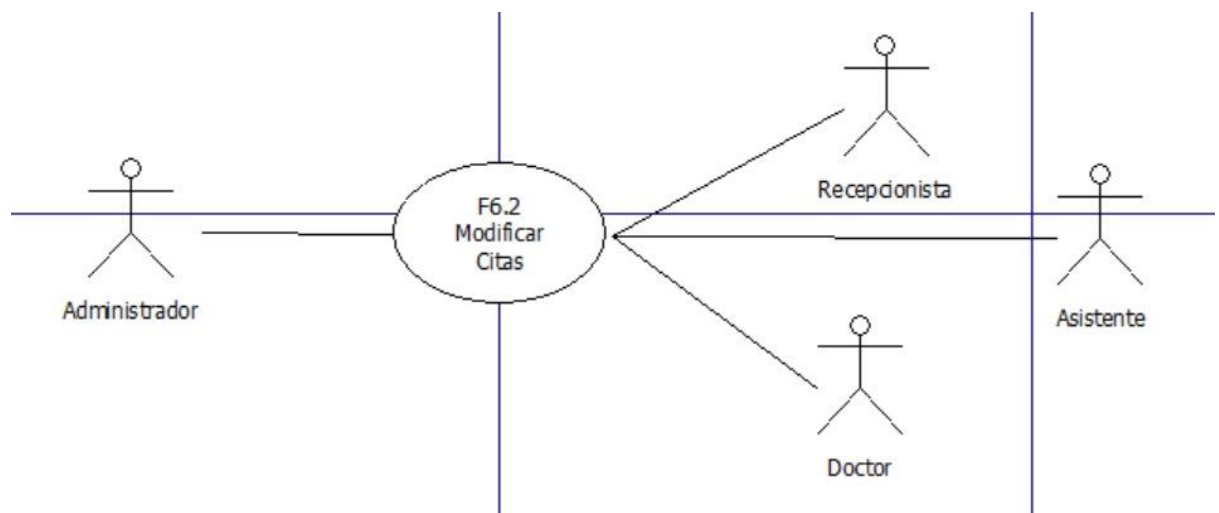
Excepciones:**Tabla 22***Excepciones en el Proceso de Programar Citas Médicas*

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 Alternativo:

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.

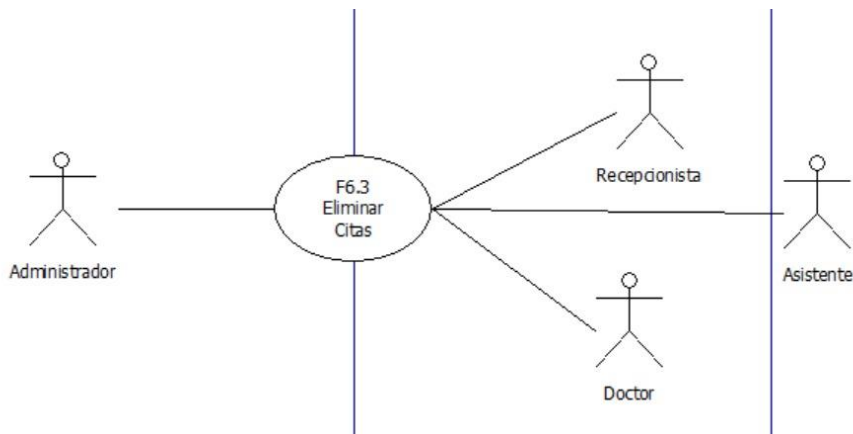
Flujo Alternativo:

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.

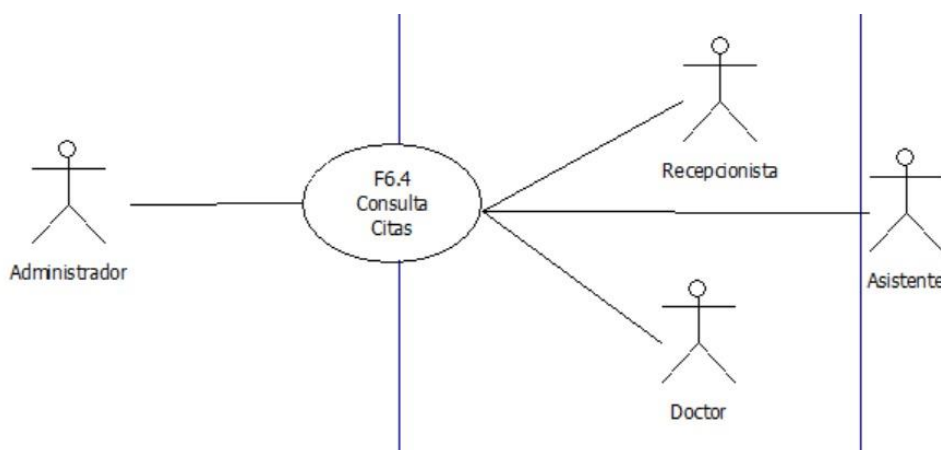
Flujo Alternativo:

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 Alternativo:

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

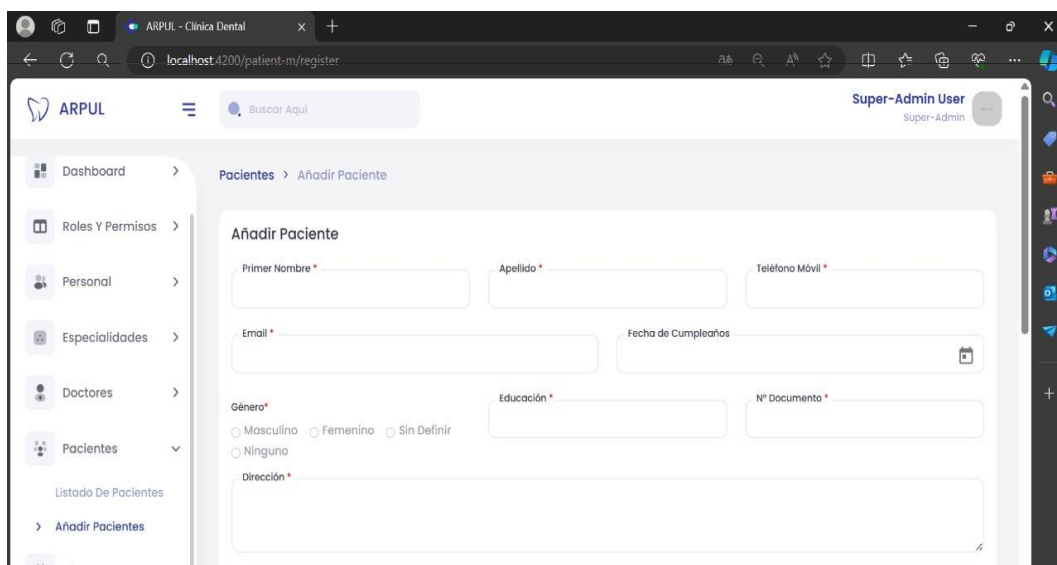
Las interfaces de usuario están diseñadas para facilitar la interacción eficiente con el sistema, proporcionando accesibilidad y seguridad en la gestión de datos y operaciones del sistema. Se describen a continuación las interfaces específicas para cada funcionalidad relevante del Sprint 1:

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



The screenshot displays the 'Añadir Paciente' (Add Patient) form within the ARPUL web application. The interface includes a sidebar menu with options like 'Dashboard', 'Roles Y Permisos', 'Personal', 'Especialidades', 'Doctores', and 'Pacientes'. The main content area is titled 'Pacientes > Añadir Paciente' and contains the following form fields:

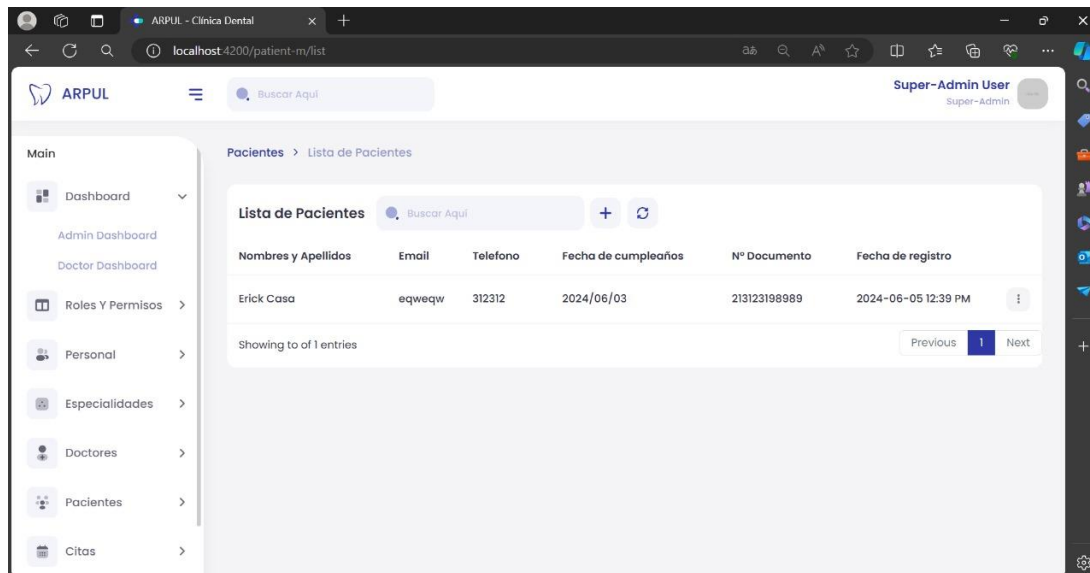
- Primer Nombre *
- Apellido *
- Teléfono Móvil *
- Email *
- Fecha de Cumpleaños (with a calendar icon)
- Género* (radio buttons for Masculino, Femenino, Sin Definir, and Ninguno)
- Educación *
- Nº Documento *
- Dirección *

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

The screenshot shows the 'Añadir Citas' form in the ARPUL application. The form is divided into several sections:

- Appointment Details:** Fields for 'Fecha de la Cita' (Appointment Date), 'Hora' (Time) with a dropdown menu, and 'Especialidades' (Specialties) with a dropdown menu. A 'Revisar' (Review) button is present.
- Doctor Availability:** A table with columns 'DOCTOR', 'DISPONIBILIDAD', and 'VER'.
- Patient Information:** A section titled 'Datos del paciente:' containing fields for 'Primer Nombre' (First Name), 'Apellido' (Last Name), 'Teléfono Móvil' (Mobile Phone), 'N° Documento' (ID Number), 'Nombre del Acompañante' (Accompanying Name), and 'Apellido del Acompañante' (Accompanying Last Name).

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

The screenshot shows the 'Listado de Citas' page in the ARPUL application. The page includes a search bar and a table of appointments. The table has the following columns: Doctor, Paciente, Fecha, Hora, Especialidad, Status, and Fecha de registro. A single appointment is listed with the status 'PENDIENTE'.

Doctor	Paciente	Fecha	Hora	Especialidad	Status	Fecha de registro
Doctor Medicina	Erick Casa	2024-06-05	11:00 AM - 11:15 AM	Cirugia General	PENDIENTE	2024-06-05 12:43 PM

Showing to of 1 entries

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

```
admin_clinica > src > app > medical > patient-m > add-patient-m > add-patient-m.component.ts > AddPatientMCompo
 9   export class AddPatientMComponent {
66   save(){
132     if(this.relationship_responsible){
133       formData.append("relationship_responsible",this.relationship_responsible);
134     }
135     if(this.current_disease){
136       formData.append("current_disease",this.current_disease);
137     }
138
139     formData.append("ta",this.ta+"");
140     formData.append("temperatura",this.temperatura+"");
141     formData.append("fc",this.fc+"");
142     formData.append("fr",this.fr+"");
143     formData.append("peso",this.peso+"");
144     this.patientService.registerPatient(formData).subscribe((resp:any) => {
145       console.log(resp);
146     });

```

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(.*)\][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 observar 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

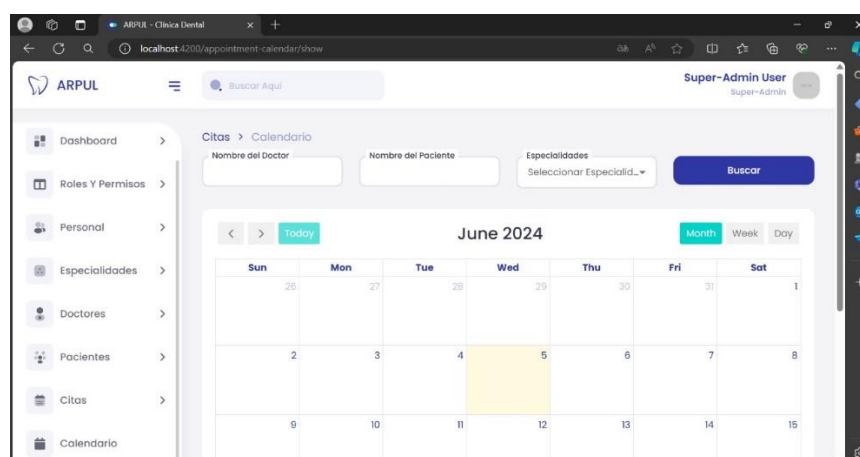
Las interfaces de usuario están diseñadas para facilitar la interacción eficiente con el sistema, proporcionando accesibilidad y seguridad en la gestión de datos y operaciones del sistema. A continuación, se describen las interfaces específicas para cada funcionalidad relevante del Sprint 3:

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.

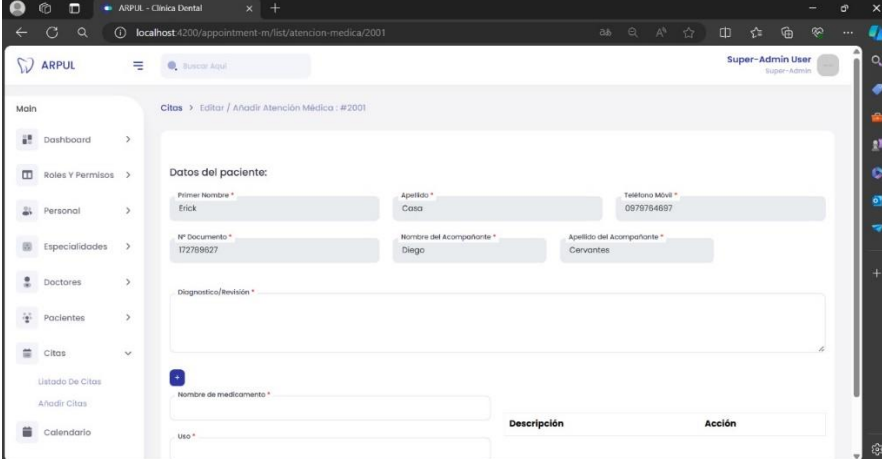
La **Figura 52** presenta la pestaña del calendario de citas médicas, que permite al personal de la clínica visualizar y gestionar todas las citas programadas de manera clara y accesible.

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



The screenshot displays the 'Atención de la Consulta Médica' form in the ARPUL application. The form is titled 'Citas > Editar / Añadir Atención Médica : #2001'. It features a sidebar menu on the left with options like Dashboard, Roles Y Permisos, Personal, Especialidades, Doctores, Pacientes, Citas, and Calendario. The main form area contains the following fields:

- Datos del paciente:**
 - Primer Nombre *: Erick
 - Apellido *: Casa
 - Teléfono Móvil *: 0979794697
 - Nº Documento *: 17279827
 - Nombre del Acompañante *: Diego
 - Apellido del Acompañante *: Cervantes
- Diagnóstico/Remisión ***: A large text area for recording the diagnosis or referral.
- Medicamentos:** A table with columns for 'Nombre de medicamento', 'Descripción', and 'Acción'. A '+' button is visible to add new entries.
- Uso ***: A field for recording the use of the medication.

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

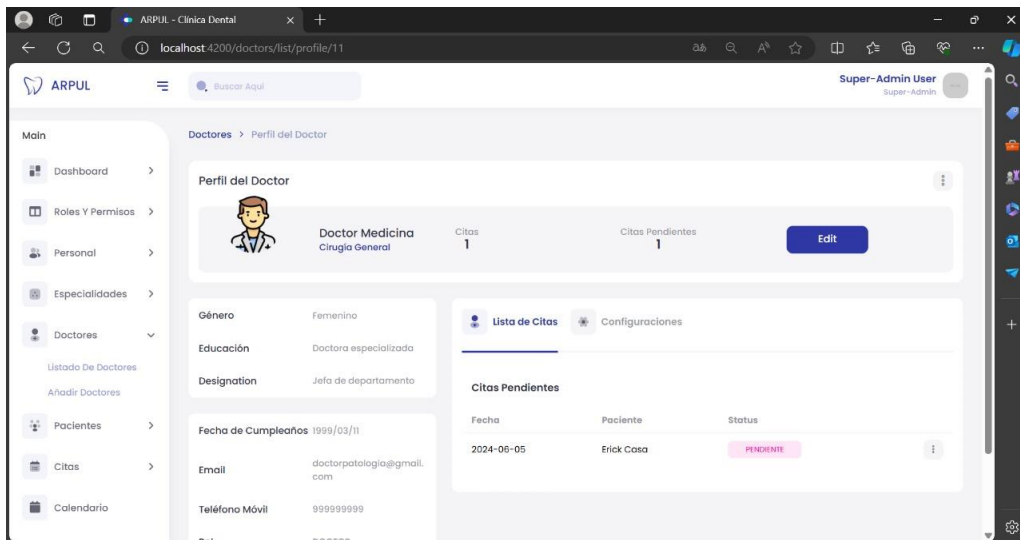
La **Figura 53** muestra la pestaña para la atención de la consulta médica, donde los doctores pueden registrar diagnósticos, tratamientos y otros detalles relevantes de cada consulta médica.

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

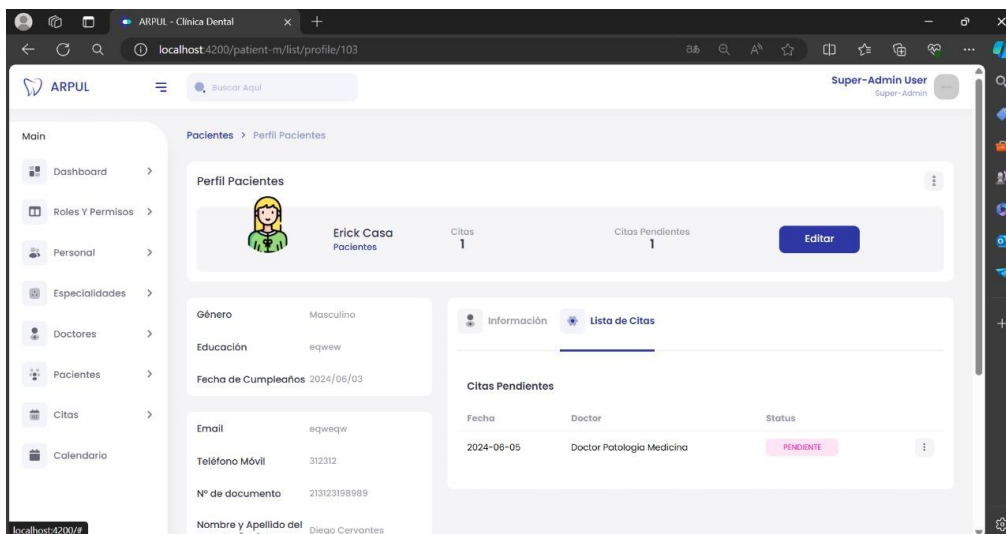


Nota. Pestaña que muestra el perfil detallado del doctor, incluyendo su especialidad y horarios.

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.

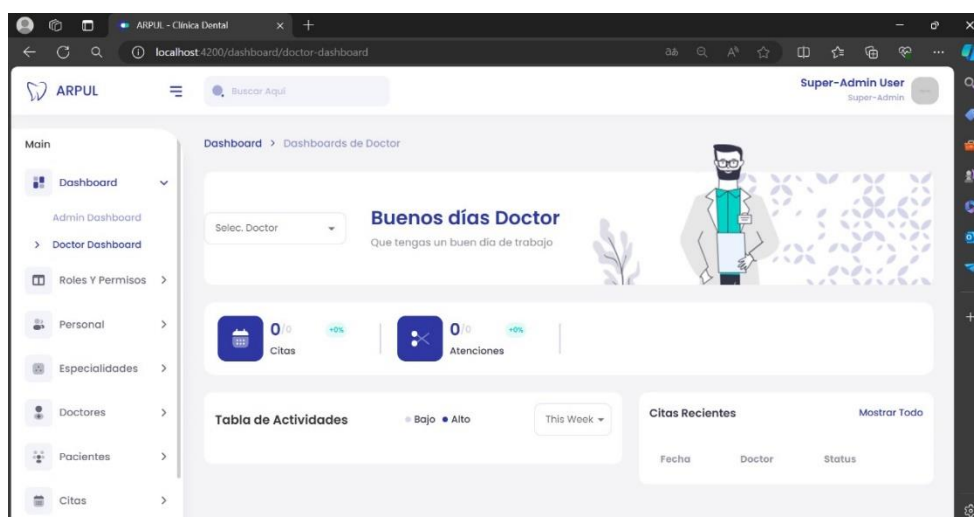
La **Figura 55** muestra la pestaña del perfil del paciente, donde se puede acceder a toda la información médica relevante y el historial de citas del paciente.

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

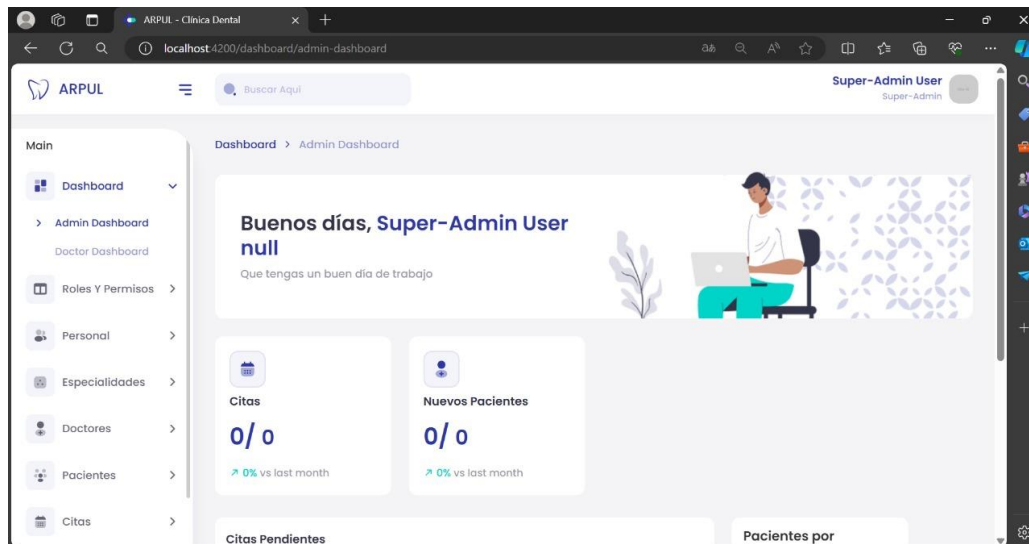


Nota. Dashboard que muestra los KPIs para el Administrador y datos relevantes para la administración de la clínica.

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*

```

$cachedRecord = Redis::get('profile_patient_#'.$id);
if(isset($cachedRecord)) {
    Redis::del('profile_patient_#'.$id);
}
// $request->request->add(["birth_date" => Carbon::parse
$patient->update($request->all());

if($patient->person){
    $patient->person->update($request->all());
}
return response()->json([
    "message" => 200
]);

```

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_attention_current = resp.num_appointments_attention_current;
      this.num_appointments_attention_before = resp.num_appointments_attention_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 relaciona 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

The screenshot shows a GitHub repository page for 'erickcasa1705 / ProyectodeTitulacionFinal'. The repository is private and has 1 commit on the main branch. The file structure is as follows:

File Name	Author	Commit Date
DrawerContainer	ManuealdePoda	5 months ago
MenuButton	ManuealdePoda	5 months ago
MenuImage	ManuealdePoda	5 months ago
assets	ManuealdePoda	5 months ago
img	ManuealdePoda	5 months ago
windows	ManuealdePoda	5 months ago
.gitignore	ManuealdePoda	5 months ago

The repository also shows 0 stars, 1 watcher, and 0 forks. There are no releases or packages published.

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?**
 - o **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?**
 - o **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?**
 - o **Respuesta:** Pensaría que no, esto que lo hagamos solo los administradores.
4. **¿Cómo debería ser el sistema de reserva de citas?**
 - o **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?**
 - o **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

```

1 import { Component, OnInit } from '@angular/core';
2 import { FormControl, FormGroup, Validators } from '@angular/forms';
3 import { Router } from '@angular/router';
4 import { AuthService } from 'src/app/shared/auth/auth.service';
5 import { routes } from 'src/app/shared/routes/routes';
6
7 @Component({
8   selector: 'app-login',
9   templateUrl: './login.component.html',
10  styleUrls: ['./login.component.scss'],
11 })
12 export class LoginComponent implements OnInit {
13   public routes = routes;
14   public passwordClass = false;
15   public ERROR = false;
16   form = new FormGroup({
17     email: new FormControl('josecode@gmail.com', [
18       Validators.required,
19       Validators.email,
20     ]),
21     password: new FormControl('12345678', [Validators.required]),
22   });
23
24   get f() {
25     return this.form.controls;
26   }
27
28   constructor(public auth: AuthService, public router: Router) {}
29   ngOnInit(): void {
30     if (localStorage.getItem('authenticated')) {
31       // localStorage.removeItem('authenticated');
32       this.router.navigate([routes.adminDashboard]);
33     }
34   }
35
36   loginFormSubmit() {
37     if (this.form.valid) {
38       this.ERROR = false;
39       this.auth.login(this.form.value.email ? this.form.value.email : '', this.form.value.password ? this.form.value.password : '')
40         .subscribe((resp:any) => {
41           console.log(resp);
42           if(resp){
43             // EL LOGIN ES EXITOSO
44             setTimeout(() => {
45               document.location.reload();
46             }, 50);
47           }else{
48             // EL LOGIN NO ES EXITOSO

```

```

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

```

1 import { Component } from '@angular/core';
2 import { Router } from '@angular/router';
3 import { AuthService } from 'src/app/shared/auth/auth.service';
4 import { routes } from 'src/app/shared/routes/routes';
5 import { SideBarService } from 'src/app/shared/side-bar/side-bar.service';
6
7 @Component({
8   selector: 'app-header',
9   templateUrl: './header.component.html',
10  styleUrls: ['./header.component.scss'],
11 })
12 export class HeaderComponent {
13   public routes = routes;
14   public openBox = false;
15   public miniSidebar = false;
16   public addClass = false;
17   public user:any;
18
19   constructor(public router: Router,private sideBar: SideBarService,public auth: AuthService) {
20     this.sideBar.toggleSideBar.subscribe((res: string) => {
21       if (res == 'true') {
22         this.miniSidebar = true;
23       } else {
24         this.miniSidebar = false;
25       }
26     });
27     let USER = localStorage.getItem("user");
28     this.user = JSON.parse(USER ? USER : '');
29     console.log(this.user);
30   }
31   getRole(){
32     let roleName = "";
33     this.user.roles.forEach((rol:any) => {
34       roleName = rol;
35     });
36     return roleName;
37   }
38   openBoxFunc() {
39     this.openBox = !this.openBox;
40     /* eslint no-var: off */
41     var mainWrapper = document.getElementsByClassName('main-wrapper')[0];
42     if (this.openBox) {
43       mainWrapper.classList.add('open-msg-box');
44     } else {
45       mainWrapper.classList.remove('open-msg-box');
46     }
47   }

```

```

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.components > ...
1  import { Component } from '@angular/core';
2  import { NavigationEnd, Router } from '@angular/router';
3  import { AuthService } from 'src/app/shared/auth/auth.service';
4  import { DataService } from 'src/app/shared/data/data.service';
5  import { MenuItem, SideBarData } from 'src/app/shared/models/models';
6  import { routes } from 'src/app/shared/routes/routes';
7  import { SideBarService } from 'src/app/shared/side-bar/side-bar.service';
8
9  @Component({
10   selector: 'app-sidebar',
11   templateUrl: './sidebar.component.html',
12   styleUrls: ['./sidebar.component.scss'],
13 })
14 export class SidebarComponent {
15   base = '';
16   page = '';
17   currentUrl = '';
18   public classAdd = false;
19
20   public multilevel: Array<boolean> = [false, false, false];
21
22   public routes = routes;
23   public sidebarData: Array<SideBarData> = [];
24   public user:any;
25   constructor(
26     private data: DataService,
27     private router: Router,
28     private sideBar: SideBarService,
29     public authService: AuthService,
30   ) {
31     // this.user = this.authService.user;
32     let USER = localStorage.getItem("user");
33     this.user = JSON.parse(USER ? USER : '');
34     // INICIO
35     if(this.user && this.user.roles.includes("Super-Admin")){
36       this.sidebarData = this.data.sideBar;
37     }else{
38       // VAMOS A FILTRAR Y VALIDAR QUE OPCIONES PUEDE VER ESE ROL
39       let permissions = this.user.permissions;
40

```

```

admin_clinica > src > app > common-component > sidebar > TS sidebar.components > ...
41   constructor(
42     let SIDE_BAR_G:any = [];
43     this.data.sideBar.forEach((side:any) => {
44       let SIDE_B:any = [];
45       side.menu.forEach((menu_s:any) => {
46         if(menu_s.subMenus.length > 0){
47           let SUB_MENU_S = menu_s.subMenus.filter((submenu:any) => permissions.includes(submenu.permission) && submenu.show_nav);
48           if(SUB_MENU_S.length > 0){
49             menu_s.subMenus = SUB_MENU_S;
50             SIDE_B.push(menu_s);
51           }
52         }else{
53           if(permissions.includes(menu_s.permission)){
54             menu_s.subMenus = [];
55             SIDE_B.push(menu_s);
56           }
57         }
58       });
59       if(SIDE_B.length > 0){
60         side.menu = SIDE_B;
61         SIDE_BAR_G.push(side);
62       }
63     });
64     this.sidebarData = SIDE_BAR_G;
65
66     // FIN
67     router.events.subscribe((event: object) => {
68       if (event instanceof NavigationEnd) {
69         this.getRoutes(event);
70       }
71     });
72     this.getRoutes(this.router);
73
74
75   public expandSubMenu(menu: MenuItem): void {
76     sessionStorage.setItem('menuValue', menu.menuValue);
77     this.sidebarData.map((mainMenu: SideBarData) => {
78       mainMenu.menu.map((resMenu: MenuItem) => {
79         if (resMenu.menuValue == menu.menuValue) {
80           menu.showSubRoute = !menu.showSubRoute;
81         } else {
82           resMenu.showSubRoute = false;
83         }
84       });
85     });
86

```

```

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 (final) > src > app > medical > appointment > add-appointments > AddAppointmentsComponent > AddAppointmentsComponent > save
1  import { Component } from '@angular/core';
2  import { AppointmentService } from '../service/appointment.service';
3
4  @Component({
5    selector: 'app-add-appointments',
6    templateUrl: './add-appointments.component.html',
7    styleUrls: ['./add-appointments.component.scss']
8  })
9  export class AddAppointmentsComponent {
10
11
12    hours:any = [];
13    specialities:any = [];
14    date_appointment:any;
15    hours:any;
16    specialitie_id:any;
17
18    name:string = '';
19    surname:string = '';
20    mobile:string = '';
21    n_document:number = 0;
22    name_companion:string = '';
23    surname_companion:string = '';
24
25    amount:number = 0;
26    amount_add:number = 0;
27    method_payment:string = '';
28
29    DOCTORS:any = [];
30    DOCTOR_SELECTED:any;
31    selected_segment_hour:any;
32
33    public text_success:string = '';
34    public text_validation:string = '';
35    constructor(
36      public appointmentService: AppointmentService,
37    ) {
38
39    }
40
41    ngOnInit(): void {
42      //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
43      //Add 'implements OnInit' to the class.
44      this.appointmentService.listenConfig().subscribe((resp:any) => {
45        this.hours = resp.hours;
46        this.specialities = resp.specialities;
47      })
48    }
49    save(){
50      this.text_validation = '';
51
52      if(!this.name || !this.surname || !this.mobile || !this.n_document || !this.name_companion || !this.surname_companion || !this.date_appointment
53         || !this.hours || !this.selected_segment_hour){
54        this.text_validation = "LOS CAMPOS SON NECESARIOS (SEGMENTO DE HORA, LA FECHA , LA ESPECIALIDAD, PACIENTE)";
55      }
56      return;
57    }
58
59

```

```

admin_clinica > src > app > medical > appointment > add-appointments > 78 add-appointments.components > 78 Add
9   export class AddAppointmentsComponent {
48   save(){
59
60     let data = {
61       "doctor_id": this.DOCTOR_SELECTED.doctor_id,
62       // "patient_id",
63       name: this.name,
64       surname: this.surname,
65       mobile: this.mobile,
66       n_document: this.n_document,
67       name_companion: this.name_companion,
68       surname_companion: this.surname_companion,
69       "date_appointment": this.date_appointment,
70       "specialitie_id": this.specialitie_id,
71       "doctor_schedule_join_hour_id": this.selected_segment_hour_id,
72       amount: this.amount,
73       amount_add: this.amount_add,
74       method_payment: this.method_payment,
75     }
76
77     this.appointmentService.registerAppointment(data).subscribe((resp:any) => {
78       console.log(resp);
79
80       this.text_success = "LA CITA MEDICA SE REGISTRO CON EXITO";
81     });
82   }
83
84   filtro(){
85     let data = {
86       date_appointment: this.date_appointment,
87       hour: this.hour,
88       specialitie_id: this.specialitie_id,
89     }
90     this.appointmentService.listfilter(data).subscribe((resp:any) => {
91       console.log(resp);
92       this.DOCTORS = resp.doctors;
93     })
94   }
95
96   countDisponibilidad(DOCTOR:any){
97     let SEGMENTS = [];
98     SEGMENTS = DOCTOR.segments.filter((item:any) => !item.is_appointment);
99     return SEGMENTS.length;
100  }
101
102   showSegment(DOCTOR:any){
103     this.DOCTOR_SELECTED = DOCTOR;
104   }
105
106   selectSegment(SEGMENT:any){
107     this.selected_segment_hour = SEGMENT;
108   }

```

```

110  filterPatient(){
111    this.appointmentService.listPatient(this.n_document+"").subscribe((resp:any) => {
112      console.log(resp);
113      if(resp.message == 403){
114        this.name = '';
115        this.surname = '';
116        this.mobile = '';
117        this.n_document = 0;
118      }else{
119        this.name = resp.name;
120        this.surname = resp.surname;
121        this.mobile = resp.mobile;
122        this.n_document = resp.n_document;
123      }
124    })
125  }
126
127  resetPatient(){
128    this.name = '';
129    this.surname = '';
130    this.mobile = '';
131    this.n_document = 0;
132  }
133 }
134

```

Anexo H

Funcionalidad de Atención Médica

```

adms@dimka ~$ cd > app > medical > appointment > atencion-medical > @ atencion-medical.component.ts > ...
1 import { Component } from '@angular/core';
2 import { AppointmentService } from '../service/appointment.service';
3 import { ActivatedRoute } from '@angular/router';
4
5 @Component({
6   selector: 'app-atencion-medical',
7   templateUrl: './atencion-medical.component.html',
8   styleUrls: ['./atencion-medical.component.scss']
9 })
10 export class AtencionMedicalComponent {
11
12   name:string = '';
13   surname:string = '';
14   mobile:string = '';
15   n_documento:number = 0;
16   name_companion:string = '';
17   surname_companion:string = '';
18
19   public text_success:string = '';
20   public text_validation:string = '';
21
22   public appointment_id:any;
23   public appointment_selected:any;
24
25   description:any;
26   name_medical:any;
27   uso:any;
28
29   public medical:any = [];
30   public appointment_attention_selected:any;
31   constructor(
32     public appointmentService: AppointmentService,
33     public activatedRoute: ActivatedRoute,
34   ) {
35
36   }
37
38   ngOnInit(): void {
39     //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
40     //Add 'implements OnInit' to the class.
41     this.activatedRoute.params.subscribe((resp:any) => {
42       console.log(resp);
43       this.appointment_id = resp.id;
44     });
45
46     this.appointmentService.showAppointment(this.appointment_id).subscribe((resp:any) => {
47       console.log(resp);
48       this.appointment_selected = resp.appointment;
49       // Datos del paciente
50       this.name = this.appointment_selected.patient.name;
51       this.surname = this.appointment_selected.patient.surname;
52       this.mobile = this.appointment_selected.patient.mobile;
53       this.n_documento = this.appointment_selected.patient.n_documento;
54       this.name_companion = this.appointment_selected.patient.name_companion;
55       this.surname_companion = this.appointment_selected.patient.surname_companion;
56     });
57
58     this.appointmentService.showAppointmentAttention(this.appointment_id).subscribe((resp:any) => {
59       console.log(resp);
60       this.appointment_attention_selected = resp.appointment_attention;
61       this.medical = this.appointment_attention_selected.receta_medica;
62       this.description = this.appointment_attention_selected.description;
63     });
64   }
65
66   addMedicamento(){
67     this.medical.push({
68       name_medical: this.name_medical,
69       uso: this.uso,
70     });
71     this.uso = '';
72     this.name_medical = '';
73   }
74
75   deleteMedical(i:any) {
76     this.medical.splice(i,1);
77   }
78
79   save(){
80
81     if(!this.description || this.medical.length == 0){
82       this.text_validation = "ES NECESARIO INGRESAR EL DIAGNOSTICO Y UNA RECETA MEDICA";
83       return;
84     }
85
86     let data = {
87       appointment_id: this.appointment_id,
88       patient_id: this.appointment_selected.patient_id,
89       description: this.description,
90       medical: this.medical,
91     };
92
93     this.appointmentService.registerAttention(data).subscribe((resp:any) => {
94       console.log(resp);
95       this.text_success = "SE GUARDO LA INFORMACIÓN DE LA ATENCIÓN MÉDICA";
96     });
97   }
98 }
99
100

```

Anexo I

Funcionalidad de Editar una Cita

```

admin_clinica > src > app > medical > appointment > edit-appointments > edit-appointments.component.ts > EditAppointmentsComponent > save
1  import { Component } from '@angular/core';
2  import { AppointmentService } from '../service/appointment.service';
3  import { ActivatedRoute } from '@angular/router';
4
5  @Component({
6    selector: 'app-edit-appointments',
7    templateUrl: './edit-appointments.component.html',
8    styleUrls: ['./edit-appointments.component.scss']
9  })
10 export class EditAppointmentsComponent {
11    hours:any = [];
12    specialities:any = [];
13    date_appointment:any;
14    hour:any;
15    specialitie_id:any;
16
17    name:string = '';
18    surname:string = '';
19    mobile:string = '';
20    n_document:number = 0;
21    name_companion:string = '';
22    surname_companion:string = '';
23
24    amount:number = 0;
25    amount_add:number = 0;
26    method_payment:string = '';
27
28    DOCTORS:any = [];
29    DOCTOR_SELECTED:any;
30    selected_segment_hour:any;
31
32    public text_success:string = '';
33    public text_validation:string = '';
34    public appointment_id:any;
35    public appointment_selected:any;
36    constructor(
37      public appointmentService: AppointmentService,
38      public activatedRoute: ActivatedRoute,
39    ) {
40    }
41  }
42
43  ngOnInit(): void {
44    //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
45    //Add 'implements OnInit' to the class.
46    this.activatedRoute.params.subscribe((resp:any) => {
47      console.log(resp);
48      this.appointment_id = resp.id;
49    })

```

```

admin_clinica > src > app > medical > appointment > edit-appointments > edit-appointments.component.ts > EditAppointmentsComponent > save
10 export class EditAppointmentsComponent {
43   ngOnInit(): void {
44
45     this.appointmentService.listConfig().subscribe((resp:any) => {
46       this.hours = resp.hours;
47       this.specialities = resp.specialities;
48
49     });
50
51     this.appointmentService.showAppointment(this.appointment_id).subscribe((resp:any) => {
52       console.log(resp);
53
54       this.appointment_selected = resp.appointment;
55       // Datos del paciente
56       this.name = this.appointment_selected.patient.name;
57       this.surname = this.appointment_selected.patient.surname;
58       this.mobile = this.appointment_selected.patient.mobile;
59       this.n_document = this.appointment_selected.patient.n_document;
60       this.name_companion = this.appointment_selected.patient.name_companion;
61       this.surname_companion = this.appointment_selected.patient.surname_companion;
62
63       this.date_appointment = new Date(this.appointment_selected.date_appointment).toISOString();
64
65       this.specialitie_id = this.appointment_selected.specialitie_id;
66       // this.selected_segment_hour = this.appointment_selected.selected_segment_hour;
67       this.amount = this.appointment_selected.amount;
68       this.hour = this.appointment_selected.segment_hour.format('segment:hour');
69       // this.filtro();
70     });
71   }
72
73   save(){
74     this.text_validation = '';
75
76     if(!this.date_appointment || !this.specialitie_id || !this.amount){
77       this.text_validation = "LOS CAMPOS SON NECESARIOS (LA FECHA , LA ESPECIALIDAD Y EL TOTAL DE PAGOS)";
78       return;
79     }
80
81     if(new Date(this.date_appointment).getTime() != new Date(this.appointment_selected.date_appointment).getTime()){
82       if(!this.selected_segment_hour){
83         this.text_validation = "NECESITAS SELECCIONAR UN SEGMENTO";
84         return;
85       }
86     }
87     // || !this.selected_segment_hour
88
89     let data = {
90       "doctor_id": this.DOCTOR_SELECTED.doctor_id,
91       "date_appointment": this.date_appointment,
92       "specialitie_id": this.specialitie_id,
93       "doctor_schedule_join_hour_id": this.selected_segment_hour ? this.selected_segment_hour.id : this.appointment_selected.doctor_schedule_join_hour_id,
94       amount: this.amount,
95     };
96
97     this.appointmentService.updateAppointment(this.appointment_id,data).subscribe((resp:any) => {
98       console.log(resp);
99       if(resp.message == 403){
100         this.text_validation = resp.message_text;

```

```

admin clinica > src > app > medical > appointment > edit-appointments > 78 edit-appointments.components.ts > 74 EditAppointmentsComponent > 70 save
10 export class EditAppointmentsComponent {
78 save(){
101
102   this.appointmentService.updateAppointment(this.appointment_id,data).subscribe((resp:any) => {
103     console.log(resp);
104     if(resp.message == 403){
105       this.text_validation = resp.message_text;
106     }else{
107       this.text_success = "LA CITA MEDICA SE EDITO CON EXITO";
108     }
109   });
110 }
111
112 onDataChange($event:any){
113   this.DOCTORS = [];
114   this.selected_segoment hour = null;
115   this.DOCTOR_SELECTED = null;
116 }
117 filtro(){
118   let data = {
119     date_appointment: this.date_appointment,
120     hour: this.hour,
121     specialitie_id : this.specialitie_id,
122   }
123   this.appointmentService.listfilter(data).subscribe((resp:any) => {
124     console.log(resp);
125     this.DOCTORS = resp.doctors;
126
127     // this.DOCTORS.forEach((doctor:any) => {
128     //   if(doctor.doctor_id == this.appointment_selected.doctor_id){
129     //     let INDEX = doctor.segments.findIndex((item:any) => item.id == this.appointment_selected.doctor_schedule_join_hour_id)
130     //     if(INDEX != -1){
131     //       this.showSegment(doctor);
132     //     }
133     //   }
134     // });
135   });
136 }
137
138 isDoctorSelected(DOCTOR:any){
139   if(DOCTOR.doctor_id == this.appointment_selected.doctor_id){
140     return true;
141   }
142   return false;
143 }
144 isSegmentSelected(SEGMENT:any){
145   if(SEGMENT.id == this.appointment_selected.doctor_schedule_join_hour_id){
146     return true;
147   }
148   return false;
149 }
150 countDisponibilidad(DOCTOR:any){
151   let SEGMENTS = [];
152   SEGMENTS = DOCTOR.segments.filter((item:any) => !item.is_appointment);
153   return SEGMENTS.length;
154 }

```

```

admin clinica > src > app > medical > appointment > edit-appointments > 78 edit-appointments.components.ts > 74 EditAppointmentsComponent > 70 save
10 export class EditAppointmentsComponent {
117 filtro(){
123   this.appointmentService.listfilter(data).subscribe((resp:any) => {
124     // });
125   }
126 }
127
128 isDoctorSelected(DOCTOR:any){
129   if(DOCTOR.doctor_id == this.appointment_selected.doctor_id){
130     return true;
131   }
132   return false;
133 }
134 isSegmentSelected(SEGMENT:any){
135   if(SEGMENT.id == this.appointment_selected.doctor_schedule_join_hour_id){
136     return true;
137   }
138   return false;
139 }
140 countDisponibilidad(DOCTOR:any){
141   let SEGMENTS = [];
142   SEGMENTS = DOCTOR.segments.filter((item:any) => !item.is_appointment);
143   return SEGMENTS.length;
144 }
145
146 showSegment(DOCTOR:any){
147   this.DOCTOR_SELECTED = DOCTOR;
148 }
149
150 selectSegment(SEGMENT:any){
151   this.selected_segoment_hour = SEGMENT;
152 }
153
154 filterPatient(){
155   this.appointmentService.listPatient(this.n_documento*).subscribe((resp:any) => {
156     console.log(resp);
157     if(resp.message == 403){
158       this.name = '';
159       this.surname = '';
160       this.mobile = '';
161       this.n_documento = 0;
162     }else{
163       this.name = resp.name;
164       this.surname = resp.surname;
165       this.mobile = resp.mobile;
166       this.n_documento = resp.n_documento;
167     }
168   });
169 }
170
171 resetPatient(){
172   this.name = '';
173   this.surname = '';
174   this.mobile = '';
175   this.n_documento = 0;
176 }
177 }
178
179
180
181
182
183
184
185
186
187
188

```

Anexo J

Funcionalidad de Listado de Citas

```

admin_clinica > src > app > medical > appointment > list-appointments > 78 list-appointments.component.ts > 74 ListAppointmentsComponent
1  import { Component } from '@angular/core';
2  import { AppointmentService } from '../service/appointment.service';
3  import { MatTableDataSource } from '@angular/material/table';
4
5
6  @Component({
7    selector: 'app-list-appointments',
8    templateUrl: './list-appointments.component.html',
9    styleUrls: ['./list-appointments.component.scss']
10 })
11 export class ListAppointmentsComponent {
12   public appointmentList: any = [];
13   dataSource: MatTableDataSource<any>;
14
15   public showFilter = false;
16   public searchDataValue = '';
17   public specialitie_id = '';
18   public date = null;
19   public lastIndex = 0;
20   public pageSize = 20;
21   public totalData = 0;
22   public skip = 0; //MIN
23   public limit: number = this.pageSize; //MAX
24   public pageIndex = 0;
25   public serialNumberArray: Array<number> = [];
26   public currentPage = 1;
27   public pageNumberArray: Array<number> = [];
28   public pageSelection: Array<any> = [];
29   public totalPages = 0;
30
31   public patient_generals: any = [];
32   public appointment_selected: any;
33
34   specialities: any = [];
35   public user: any;
36   constructor(
37     public appointmentService: AppointmentService,
38   ) {}
39
40   ngOnInit() {
41     this.getTableData();
42
43     this.appointmentService.listConfig().subscribe((resp: any) => {
44       this.specialities = resp.specialities;
45     });
46     this.user = this.appointmentService.authService.user;
47   }
48
49   isPermitted(){
50     let band = false;
51     this.user.roles.forEach((rol: any) => {
52       if(rol.toUpperCase().indexOf("DOCTOR") != -1){
53         band = true;
54       }
55     });
56     return band;
57   }
58

```

```

admin_clinica > src > app > medical > appointment > list-appointments > 78 list-appointments.component.ts > 74 ListAppointmentsComponent > 74 Page56
59 export class ListAppointmentsComponent {
60   isPermitted(permission: string) {
61     if(this.user.roles.includes('Super-Admin')){
62       return true;
63     }
64     if(this.user.permissions.includes(permission)){
65       return true;
66     }
67     return false;
68   }
69   private getTableData(page=1): void {
70     this.appointmentList = [];
71     this.serialNumberArray = [];
72     this.appointmentService.listAppointments(page, this.searchDataValue, this.specialitie_id, this.date).subscribe((resp: any) => {
73       console.log(resp);
74
75       this.totalData = resp.total;
76       this.appointmentList = resp.appointments.data;
77       // this.getTableDataGeneral();
78       this.dataSource = new MatTableDataSource<any>(this.appointmentList);
79       this.calculateTotalPages(this.totalData, this.pageSize);
80     });
81   }
82
83   }
84
85   }
86   getTableDataGeneral() {
87     this.appointmentList = [];
88     this.serialNumberArray = [];
89
90     this.patient_generals.map((res: any, index: number) => {
91       const serialNumber = index + 1;
92       if (index >= this.skip && serialNumber <= this.limit) {
93         this.appointmentList.push(res);
94         this.serialNumberArray.push(serialNumber);
95       }
96     });
97   }
98   this.dataSource = new MatTableDataSource<any>(this.appointmentList);
99   this.calculateTotalPages(this.totalData, this.pageSize);
100 }
101
102 selectUser(rol: any) {
103   this.appointment_selected = rol;
104 }
105
106 deleteAppointment() {
107
108   this.appointmentService.deleteAppointment(this.appointment_selected.id).subscribe((resp: any) => {
109     console.log(resp);
110     let INDEX = this.appointmentList.findIndex((item: any) => item.id == this.appointment_selected.id);
111     if (INDEX != -1) {
112       this.appointmentList.splice(INDEX, 1);
113     }
114     $('delete_patient').hide();
115     $('delete_patient').removeClass("show");

```

```

admin_clinica > src > app > medical > appointment > list-appointments > list-appointments.component.ts > ListAppointmentsComponent > g
10 export class ListAppointmentsComponent {
105
106   deleteAppointment(){
107
108     this.appointmentService.deleteAppointment(this.appointment_selected.id).subscribe((resp:any) => {
109       console.log(resp);
110       let INDEX = this.appointmentList.findIndex((item:any) => item.id == this.appointment_selected.id);
111       if(INDEX != -1){
112         this.appointmentList.splice(INDEX,1);
113
114         $('#delete_patient').hide();
115         $('#delete_patient').removeClass("show");
116         $('#modal_backdrop').remove();
117         $('body').removeClass();
118         $('body').removeAttr("style");
119
120         this.appointment_selected = null;
121       }
122     })
123
124     // eslint-disable-next-line @typescript-eslint/no-explicit-any
125     public searchData() {
126       // this.dataSource.filter = value.trim().toLowerCase();
127       // this.appointmentList = this.dataSource.filteredData;
128       this.pageSelection = [];
129       this.limit = this.pageSize;
130       this.skip = 0;
131       this.currentPage = 1;
132
133       this.getTableData();
134     }
135
136     public sortData(sort: any) {
137       const data = this.appointmentList.slice();
138
139       if (!sort.active || sort.direction === '') {
140         this.appointmentList = data;
141       } else {
142         this.appointmentList = data.sort((a:any, b:any) => {
143           // eslint-disable-next-line @typescript-eslint/no-explicit-any
144           const aValue = (a as any)[sort.active];
145           // eslint-disable-next-line @typescript-eslint/no-explicit-any
146           const bValue = (b as any)[sort.active];
147           return (aValue < bValue ? -1 : 1) * (sort.direction === 'asc' ? 1 : -1);
148         });
149       }
150     }
151
152     public getMoreData(event: string): void {
153       if (event == 'next') {
154         this.currentPage++;
155         this.pageIndex = this.currentPage - 1;
156         this.limit += this.pageSize;
157         this.skip = this.pageSize * this.pageIndex;
158         this.getTableData(this.currentPage);
159       } else if (event == 'previous') {
160         this.currentPage--;
161         this.pageIndex = this.currentPage - 1;

```

```

admin_clinica > src > app > medical > appointment > list-appointments > list-appointments.component.ts > ListAppointmentsCom
10 export class ListAppointmentsComponent {
111
112   public getMoreData(event: string): void {
113     if (event == 'next') {
114       this.currentPage++;
115       this.pageIndex = this.currentPage - 1;
116       this.limit += this.pageSize;
117       this.skip = this.pageSize * this.pageIndex;
118     } else if (event == 'previous') {
119       this.currentPage--;
120       this.pageIndex = this.currentPage - 1;
121       this.limit -= this.pageSize;
122       this.skip = this.pageSize * this.pageIndex;
123       this.getTableData(this.currentPage);
124     }
125   }
126
127   public moveToPage(pageNumber: number): void {
128     this.currentPage = pageNumber;
129     this.skip = this.pageSelection[pageNumber - 1].skip;
130     this.limit = this.pageSelection[pageNumber - 1].limit;
131     if (pageNumber > this.currentPage) {
132       this.pageIndex = pageNumber - 1;
133     } else if (pageNumber < this.currentPage) {
134       this.pageIndex = pageNumber + 1;
135     }
136     this.getTableData(this.currentPage);
137   }
138
139   public PageSize(): void {
140     this.pageSelection = [];
141     this.limit = this.pageSize;
142     this.skip = 0;
143     this.currentPage = 1;
144     this.searchDataValue = '';
145     this.specialitie_id = '';
146     this.date = null;
147     this.getTableData();
148   }
149
150   private calculateTotalPages(totalData: number, pageSize: number): void {
151     this.pageNumberArray = [];
152     this.totalPages = totalData / pageSize;
153     if (this.totalPages % 1 != 0) {
154       this.totalPages = Math.trunc(this.totalPages + 1) //10.6 o 10.9 11
155     }
156     /* eslint no-var: off */
157     for (var i = 1; i <= this.totalPages; i++) {
158       const limit = pageSize * i;
159       const skip = limit - pageSize;
160       this.pageNumberArray.push(i);
161       this.pageSelection.push({ skip: skip, limit: limit });
162     }
163     // 1
164     // 0 - 10
165     // 2
166     // 10 - 20
167     // 3

```

Anexo K

Funcionalidad del Calendario de Citas

```

admin_cliente > src > app > medical > calendar-appointment > appointment-calendar > appointment-calendar.components > ...
1 import { Component } from '@angular/core';
2 import { TimeGridPlugin } from '@fullcalendar/timegrid';
3 import { InteractionPlugin } from '@fullcalendar/interaction';
4 import { DayGridPlugin } from '@fullcalendar/daygrid';
5 import { AppointmentPayService } from '../../appointment-pay/service/appointment-pay.service';
6 import { CalendarAppointmentService } from '../../service/calendar-appointment.service';
7
8 @Component({
9   selector: 'app-appointment-calendar',
10  templateUrl: './appointment-calendar.component.html',
11  styleUrls: ['./appointment-calendar.component.scss']
12 })
13 export class AppointmentCalendarComponent {
14
15   // eslint-disable-next-line @typescript-eslint/no-explicit-any
16   options: any;
17   // eslint-disable-next-line @typescript-eslint/no-explicit-any
18   events: any[] = [];
19
20   public specialities: any = [];
21   public specialitie_id = '';
22   public search_doctor = '';
23   public search_patient = '';
24
25   public user: any;
26   constructor(
27     public appointmentPayService: AppointmentPayService,
28     public appointmentCalendarService: CalendarAppointmentService,
29   ) {
30     // eslint-disable-next-line @typescript-eslint/no-explicit-any
31     // this.data.getEvents().subscribe((events: any) => {
32     //   this.events = events;
33     //   this.options = { ...this.options, ...(events: events.data) };
34     // });
35     this.options = {
36       plugins: [DayGridPlugin, TimeGridPlugin, InteractionPlugin],
37       initialDate: new Date(),
38       headerToolbar: {
39         left: 'prev,next today',
40         center: 'title',
41         right: 'dayGridMonth,timeGridWeek,timeGridDay',
42       },
43       initialView: 'dayGridMonth',
44       editable: true,
45       selectable: true,
46       selectMirror: true,
47       dayMaxEvents: true,
48       events: [
49         { title: 'Meeting', start: new Date() }
50       ]
51     };
52   }
}

```

```

54   isPermitted(){
55     let band = false;
56     this.user.roles.forEach((rol: any) => {
57       if((rol).toUpperCase().indexOf("DOCTOR") != -1){
58         band = true;
59       }
60     });
61     return band;
62   }
63
64   ngOnInit(): void {
65     //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
66     //Add 'implements OnInit' to the class.
67     this.appointmentPayService.listConfig().subscribe((resp: any) => {
68       this.specialities = resp.specialities;
69     })
70     this.user = this.appointmentPayService.authService.user;
71     this.calendarAppointment();
72   }
73
74   calendarAppointment() {
75     let data = {
76       specialitie_id: this.specialitie_id,
77       search_doctor: this.search_doctor,
78       search_patient: this.search_patient,
79     }
80     this.appointmentCalendarService.calendarAppointment(data).subscribe((resp: any) => {
81       console.log(resp);
82       this.options = { ...this.options, ...(events: resp.appointments) };
83     })
84   }
85
86   searchData(){
87     this.calendarAppointment();
88   }
89 }
90

```

Anexo L

Funcionalidad de Añadir un Doctor

```

admin clinica > src > app > medical > doctors > add-doctor > add-doctor.component.ts > AddDoctorComponent > save
1 import { Component } from '@angular/core';
2 import { DoctorService } from '../service/doctor.service';
3
4 @Component({
5   selector: 'app-add-doctor',
6   templateUrl: './add-doctor.component.html',
7   styleUrls: ['./add-doctor.component.scss']
8 })
9 export class AddDoctorComponent {
10   public selectedValue!: string;
11   public name:string = '';
12   public surname:string = '';
13   public mobile:string = '';
14   public email:string = '';
15   public password:string = '';
16   public password_confirmation:string = '';
17
18   public birth_date:string = '';
19   public gender:number = 1;
20   public education:string = '';
21   public designation:string = '';
22   public address:string = '';
23
24   public roles:any = [];
25
26   public FILE_AVATAR:any;
27   public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';
28
29   public specialitie_id:any;
30   public specialities:any = [];
31
32   public text_success:string = '';
33   public text_validation:string = '';
34
35   public days_week = [
36     {
37       day: 'Lunes',
38       class: 'table-primary'
39     },
40     {
41       day: 'Martes',
42       class: 'table-secondary'
43     },
44     {
45       day: 'Miercoles',
46       class: 'table-success'
47     },
48     {
49       day: 'Jueves',
50       class: 'table-warning'
51     },
52     {
53       day: 'Viernes',
54       class: 'table-info'
55     }
56   ]
57   public hours_days:any = [];
58   public hours_selecteds:any = [];

```

```

admin clinica > src > app > medical > doctors > add-doctor > add-doctor.component.ts > AddDoctorComponent > save
9
10 export class AddDoctorComponent {
11
12   public hours_days:any = [];
13   public hours_selecteds:any = [];
14   constructor(
15     public doctorsService: DoctorService,
16   ) {
17   }
18
19   ngOnInit(): void {
20     //called after the constructor, initializing input properties, and the first call to ngOnChanges.
21     //Add implementations here to the class.
22     this.doctorsService.listConfig().subscribe((resp:any) => {
23       console.log(resp);
24       this.roles = resp.roles;
25       this.specialities = resp.specialities;
26       this.hours_days = resp.hours_days;
27     });
28   }
29
30   save(){
31     this.text_validation = '';
32     if(!this.name || !this.email || !this.surname || !this.mobile){
33       this.text_validation = "LOS CAMPOS SON NECESARIOS (Nombre, Apellido, Email, teléfono Móvil)";
34       return;
35     }
36     this.text_validation = '';
37     if(!this.password){
38       this.text_validation = "ES NECESARIO UNA CONTRASEÑA";
39       return;
40     }
41     this.text_validation = '';
42     if(!this.selectedValue || !this.specialitie_id){
43       this.text_validation = "LOS CAMPOS SON NECESARIOS (Rol, Especialidad)";
44       return;
45     }
46     if(this.password != this.password_confirmation){
47       this.text_validation = "LAS CONTRASÍAS DEBEN SER IGUALES";
48       return;
49     }
50     if(this.hours_selecteds.length == 0){
51       this.text_validation = "NECESITAS SELECCIONAR UN HORARIO AL MENOS";
52       return;
53     }
54     console.log(this.selectedValue);
55
56     let formData = new FormData();
57     formData.append("name",this.name);
58     formData.append("surname",this.surname);
59     formData.append("email",this.email);
60     formData.append("mobile",this.mobile);
61     formData.append("birth_date",this.birth_date);
62     formData.append("gender",this.gender+"");
63     formData.append("education",this.education);
64     formData.append("designation",this.designation);
65     formData.append("address",this.address);

```

```

admin clinica > src > app > medical > doctors > add-doctor > 18 add-doctor.component.ts > AddDoctorComponent > save
9 export class AddDoctorComponent {
75 save(){
183
184 let formData = new FormData();
185 formData.append("name",this.name);
186 formData.append("surname",this.surname);
187 formData.append("email",this.email);
188 formData.append("mobile",this.mobile);
189 formData.append("birth_date",this.birth_date);
110 formData.append("gender",this.gender);
111 formData.append("education",this.education);
112 formData.append("designation",this.designation);
113 formData.append("address",this.address);
114 formData.append("password",this.password);
115 formData.append("role_id",this.selectedValue);
116 formData.append("specialitie_id",this.specialitie_id);
117 formData.append("imagen",this.FILE_AVATAR);
118
119 let HOUR_SCHEDULES:any = [];
120
121 this.days_week.forEach(day:any) => {
122 let DAYS_HOURS = this.hours_selected.filter((hour_select:any) => hour_select.day_name == day.day);
123 HOUR_SCHEDULES.push({
124 day_name: day.day,
125 children: DAYS_HOURS,
126 });
127 }
128
129 formData.append("schedule_hours",JSON.stringify(HOUR_SCHEDULES));
130 this.doctorsService.registerDoctor(formData).subscribe((resp:any) => {
131 console.log(resp);
132
133 if(resp.message == 483){
134 this.text_validation = resp.message_text;
135 }else{
136 this.text_success = "El usuario ha sido registrado correctamente";
137
138 this.name = '';
139 this.surname = '';
140 this.email = '';
141 this.mobile = '';
142 this.birth_date = '';
143 this.gendor = 1;
144 this.education = '';
145 this.designation = '';
146 this.address = '';
147 this.password = '';
148 this.password_confirmation = '';
149 this.selectedValue = '';
150 this.specialitie_id = '';
151 this.FILE_AVATAR = null;
152 this.IMAGEN_PREVIZUALIZA = null;
153 this.hours_selecteds = [];
154 }
155
156 })
157 }
158

```

```

src C:\PROYECTO_FINAL\admin_clinica\src\app\medical\appointment\add-appointment\add-appointments.component.html Modified AddDoctorComponent > save
155
156 loadFile($event:any){
157 if($event.target.files[0].type.indexOf("image") < 0){
158 // alert("SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN");
159 this.text_validation = "SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN";
160 return;
161 }
162 this.text_validation = '';
163 this.FILE_AVATAR = $event.target.files[0];
164 let reader = new FileReader();
165 reader.readAsDataURL(this.FILE_AVATAR);
166 reader.onloadend = () => this.IMAGEN_PREVIZUALIZA = reader.result;
167 }
168
169 addHourItem(hours_day:any,day:any,item:any){
170
171 let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
172 // hour.hour == hours_day.hour
173 // hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
174
175 if(INDEX != -1){
176 this.hours_selecteds.splice(INDEX,1);
177 }else{
178 this.hours_selecteds.push({
179 "day": day,
180 "day_name": day.day,
181 "hours_day": hours_day,
182 "hour": hours_day.hour,
183 "grupo": "none",
184 "item": item,
185 });
186 }
187 console.log(this.hours_selecteds);
188 }
189
190 addHourAll(hours_day:any,day:any){
191 let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
192 // hour.hour == hours_day.hour && hour.grupo == "all");
193
194 let COUNT_SELECTED = this.hours_selecteds.filter((hour:any) => hour.day_name == day.day
195 // hour.hour == hours_day.hour).length;
196 if(INDEX != -1 && COUNT_SELECTED == hours_day.items.length){
197 hours_day.items.forEach((item:any) => {
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
202 if(INDEX != -1){
203 this.hours_selecteds.splice(INDEX,1);
204 }
205 });
206 }else{
207 hours_day.items.forEach((item:any) => {
208 let INDEX = this.hours_selecteds.findIndex((hour:any) => hour.day_name == day.day
209 // hour.hour == hours_day.hour
210 // hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
211
212 if(INDEX != -1){
213 this.hours_selecteds.splice(INDEX,1);
214 }

```

```

admin:clinical > src > app > medical > doctors > add-doctor > add-doctor.components.ts > AddDoctorComponent > save
9
export class AddDoctorComponent {
193
  addHourAll(hours_day:any,day:any){
200
    hours_day.items.forEach((item:any) => {
207
    });
208
  }else{
209
    hours_day.items.forEach((item:any) => {
210
      let INDEX = this.hours_selectededs.findIndex((hour:any) => hour.day_name == day.day
211
        && hour.hour == hours_day.hour
212
        && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
213
      if(INDEX != -1){
214
        this.hours_selectededs.splice(INDEX,1);
215
      }
216
      this.hours_selectededs.push({
217
        "day": day,
218
        "day_name": day.day,
219
        "hours_day": hours_day,
220
        "hour": hours_day.hour,
221
        "grupo": "all",
222
        "item": item,
223
      });
224
    });
225
  }
226
  console.log(this.hours_selectededs);
227
}

addHourAllDay($event:any, hours_day:any){
228
229
  let INDEX = this.hours_selectededs.findIndex((hour:any) => hour.hour == hours_day.hour);
230
231
  if(INDEX != -1 && !$event.currentTarget.checked){
232
    this.days_week.forEach((day) => {
233
      hours_day.items.forEach((item:any) => {
234
        let INDEX = this.hours_selectededs.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_selectededs.splice(INDEX,1);
239
        }
240
      });
241
    });
242
  }
243
}
244
}else{
245
  this.days_week.forEach((day) => {
246
    hours_day.items.forEach((item:any) => {
247
      let INDEX = this.hours_selectededs.findIndex((hour:any) => hour.day_name == day.day
248
        && hour.hour == hours_day.hour
249
        && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
250
      if(INDEX != -1){
251
        this.hours_selectededs.splice(INDEX,1);
252
      }
253
    });
254
  }
255
  setTimeout(() => {
256
    this.days_week.forEach((day) => {
257
      this.addHourAll(hours_day,day);
258
    });
259
  }, 25);
260
}
}

```

```

}

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

  let COUNT_SELECTED = this.hours_selectededs.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;
  }
}

isCheckedHour(hours_day:any,day:any,item:any){
  let INDEX = this.hours_selectededs.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 > doctor-m-profile.component.ts | DoctorMProfileComponent > doctor_id
1 import { Component } from '@angular/core';
2 import { DoctorService } from '../service/doctor.service';
3 import { ActivatedRoute } from '@angular/router';
4
5 @Component({
6   selector: 'app-doctor-m-profile',
7   templateUrl: './doctor-m-profile.component.html',
8   styleUrls: ['./doctor-m-profile.component.scss']
9 })
10 export class DoctorMProfileComponent {
11   doctorProfile:any = [];
12   option_selected:number = 1;
13
14   doctor_id:string = '';
15
16   name:string = '';
17   surname:string = '';
18   mobile:string = '';
19   email:string = '';
20   address:string = '';
21   password:string = '';
22   password_repet:string = '';
23
24   num_appointment:number = 0;
25   money_of_appointments:number = 0;
26   num_appointment_pendings:number = 0;
27   doctor_selected:any;
28   appointment_pendings:any = [];
29   appointments:any = [];
30   public text_success:string = '';
31   public text_validation:string = '';
32   constructor(
33     public doctorService: DoctorService,
34     public activatedRoute: ActivatedRoute,
35   ) {
36
37   }
38
39   ngOnInit(): void {
40     //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
41     //Add 'implements OnInit' to the class.
42     this.activatedRoute.params.subscribe((resp:any) => {
43       console.log(resp);
44       this.doctor_id = resp.id;
45     });
46     this.doctorService.profileDoctor(this.doctor_id).subscribe((resp:any) => {
47       console.log(resp);
48       this.num_appointment = resp.num_appointment;
49       this.money_of_appointments = resp.money_of_appointments;
50       this.num_appointment_pendings = resp.num_appointment_pendings;
51       this.doctor_selected = resp.doctor;
52       this.appointment_pendings = resp.appointment_pendings.data;
53       this.appointments = resp.appointments;
54
55       this.name = this.doctor_selected.name;
56       this.surname = this.doctor_selected.surname;
57       this.mobile = this.doctor_selected.mobile;
58       this.email = this.doctor_selected.email;

```

```

admin clinica > src > app > medical > doctors > doctor-m-profile > doctor-m-profile.component.ts | DoctorMProfileComponent > doctor_id
18 export class DoctorMProfileComponent {
19   ngOnInit(): void {
20     this.doctorService.profileDoctor(this.doctor_id).subscribe((resp:any) => {
21       this.num_appointment = resp.num_appointment;
22       this.money_of_appointments = resp.money_of_appointments;
23       this.num_appointment_pendings = resp.num_appointment_pendings;
24       this.doctor_selected = resp.doctor;
25       this.appointment_pendings = resp.appointment_pendings.data;
26       this.appointments = resp.appointments;
27
28       this.name = this.doctor_selected.name;
29       this.surname = this.doctor_selected.surname;
30       this.mobile = this.doctor_selected.mobile;
31       this.email = this.doctor_selected.email;
32       this.address = this.doctor_selected.address;
33     });
34   }
35
36   optionSelected(value:number){
37     this.option_selected = value;
38   }
39
40   update(){
41     this.text_validation = '';
42     this.text_success = '';
43     if(!this.name || !this.email || !this.surname || !this.mobile){
44       this.text_validation = "LOS CAMPOS SON NECESARIOS (Nombre, Apellido Email, Teléfono Móvil)";
45       return;
46     }
47     if(this.password){
48       if(this.password != this.password_repet){
49         this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
50         return;
51       }
52     }
53     let data:any = {
54       name: this.name,
55       surname: this.surname,
56       mobile: this.mobile,
57       email: this.email,
58       address: this.address,
59     };
60     if(this.password){
61       data.password = this.password;
62     }
63     this.doctorService.updatedoctorProfile(this.doctor_id,data).subscribe((resp:any) => {
64       console.log(resp);
65
66       if(resp.message == 483){
67         this.text_validation = resp.message_text;
68         return;
69       }
70       this.text_success = 'El usuario ha sido editado correctamente';
71     });
72   }
73 }

```

Anexo N

Funcionalidad de Editar un Doctor

```

admin clinica > src > app > medical > doctors > edit-doctor > edit-doctor.component.ts > EditDoctorComponent >
1 import { Component } from '@angular/core';
2 import { DoctorService } from '../service/doctor.service';
3 import { ActivatedRoute } from '@angular/router';
4
5 @Component({
6   selector: 'app-edit-doctor',
7   templateUrl: './edit-doctor.component.html',
8   styleUrls: ['./edit-doctor.component.scss']
9 })
10 export class EditDoctorComponent {
11   public selectedValue!: string;
12   public name:string = '';
13   public surname:string = '';
14   public mobile:string = '';
15   public email:string = '';
16   public password:string = '';
17   public password_confirmation:string = '';
18
19   public birth_date:string = '';
20   public gender:number = 1;
21   public education:string = '';
22   public designation:string = '';
23   public address:string = '';
24
25   public roles:any = [];
26
27   public FILE_AVATAR:any;
28   public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';
29
30   public specialitie_id:any;
31   public specialities:any = [];
32
33   public text_success:string = '';
34   public text_validation:string = '';
35
36   public days_week = [
37     {
38       day: 'Lunes',
39       class: 'table-primary'
40     },
41     {
42       day: 'Martes',
43       class: 'table-secondary'
44     },
45     {
46       day: 'Miercoles',
47       class: 'table-success'
48     },
49     {
50       day: 'Jueves',
51       class: 'table-warning'
52     },
53     {
54       day: 'Viernes',
55       class: 'table-info'
56     }
57 ]

```

```

admin clinica > src > app > medical > doctors > edit-doctor > edit-doctor.component.ts > EditDoctorComponent > save
10 export class EditDoctorComponent {
11   public hours_days:any = [];
12   public hours_selected:any = [];
13   public doctor_id:any;
14   public doctor_selected:any;
15   constructor(
16     public doctorsService: DoctorService,
17     public activatedRoute: ActivatedRoute,
18   ) {
19   }
20   ngOnInit(): void {
21     //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
22     //Add 'implements OnInit' to the class.
23     this.activatedRoute.params.subscribe((resp:any) => {
24       console.log(resp);
25       this.doctor_id = resp.id;
26     });
27
28     this.doctorsService.listConfig().subscribe((resp:any) => {
29       console.log(resp);
30       this.roles = resp.roles;
31       this.specialities = resp.specialities;
32       this.hours_days = resp.hours_days;
33
34       this.doctorsService.showDoctor(this.doctor_id).subscribe((resp:any) => {
35         console.log(resp);
36         this.doctor_selected = resp.doctor;
37
38         this.selectedValue = this.doctor_selected.role.id;
39         this.specialitie_id = this.doctor_selected.specialitie.id;
40         this.name = this.doctor_selected.name;
41         this.surname = this.doctor_selected.surname;
42         this.mobile = this.doctor_selected.mobile;
43         this.email = this.doctor_selected.email;
44         this.birth_date = new Date(this.doctor_selected.birth_date).toLocaleString();
45         this.gender = this.doctor_selected.gender;
46         this.education = this.doctor_selected.education;
47         this.designation = this.doctor_selected.designation;
48         this.address = this.doctor_selected.address;
49         this.IMAGEN_PREVIZUALIZA = this.doctor_selected.avatar;
50         this.hours_selecteds = this.doctor_selected.schedule_selecteds;
51       });
52     });
53   }
54   save(){
55     this.text_validation = '';
56     if(!this.name || !this.email || !this.surname || !this.mobile){
57       this.text_validation = "LOS CAMPOS SON NECESARIOS (Nombre, Apellido, Email, Telefono Movil)";
58       return;
59     }
60     this.text_validation = '';
61     if(!this.password){
62       this.text_validation = "ES NECESARIO UNA CONTRASEÑA";
63       return;
64     }
65   }

```

```

admin_clinica > src > app > medical > doctors > edit-doctor > edit-doctor.components.ts > EditDoctorComponent > save
18  export class EditDoctorComponent {
184  save(){
185
186      this.text_validation = '';
187      if(!this.selectedValue || !this.specialitie_id){
188          this.text_validation = "LOS CAMPOS SON NECESARIOS (Rol, Especialidad)";
189          return;
190      }
191
192      if(this.password != this.password_confirmation){
193          this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
194          return;
195      }
196
197      if(this.hours_selecteds.length == 0){
198          this.text_validation = "NECESITAS SELECCIONAR UN HORARIO AL MENOS";
199          return;
200      }
201      console.log(this.selectedValue);
202
203      let formData = new FormData();
204      formData.append("name",this.name);
205      formData.append("surname",this.surname);
206      formData.append("email",this.email);
207      formData.append("mobile",this.mobile);
208      formData.append("birth_date",this.birth_date);
209      formData.append("gender",this.gendere);
210
211      formData.append("role_id",this.selectedValue);
212      formData.append("specialitie_id",this.specialitie_id);
213
214      if(this.education){
215          formData.append("education",this.education);
216      }
217      if(this.designation){
218          formData.append("designation",this.designation);
219      }
220      if(this.address){
221          formData.append("address",this.address);
222      }
223      if(this.password){
224          formData.append("password",this.password);
225      }
226      if(this.FILE_AVATAR){
227          formData.append("imagen",this.FILE_AVATAR);
228      }
229
230      let HOUR_SCHEDULES:any = [];
231
232      this.days_week.forEach( (day:any) => {
233          let DAYS_HOURS = this.hours_selecteds.filter((hour_select:any) => hour_select.day_name == day.day);
234          HOUR_SCHEDULES.push({
235              day_name: day.day,
236              children: DAYS_HOURS,
237          });
238      })
239  }

```

```

admin_clinica > src > app > medical > doctors > edit-doctor > edit-doctor.components.ts > EditDoctorComponent > save
18  export class EditDoctorComponent {
184  save(){
185
186      this.text_validation = '';
187      if(!this.selectedValue || !this.specialitie_id){
188          this.text_validation = "LOS CAMPOS SON NECESARIOS (Rol, Especialidad)";
189          return;
190      }
191
192      if(this.password != this.password_confirmation){
193          this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
194          return;
195      }
196
197      if(this.hours_selecteds.length == 0){
198          this.text_validation = "NECESITAS SELECCIONAR UN HORARIO AL MENOS";
199          return;
200      }
201      console.log(this.selectedValue);
202
203      let formData = new FormData();
204      formData.append("name",this.name);
205      formData.append("surname",this.surname);
206      formData.append("email",this.email);
207      formData.append("mobile",this.mobile);
208      formData.append("birth_date",this.birth_date);
209      formData.append("gender",this.gendere);
210
211      formData.append("role_id",this.selectedValue);
212      formData.append("specialitie_id",this.specialitie_id);
213
214      if(this.education){
215          formData.append("education",this.education);
216      }
217      if(this.designation){
218          formData.append("designation",this.designation);
219      }
220      if(this.address){
221          formData.append("address",this.address);
222      }
223      if(this.password){
224          formData.append("password",this.password);
225      }
226      if(this.FILE_AVATAR){
227          formData.append("imagen",this.FILE_AVATAR);
228      }
229
230      let HOUR_SCHEDULES:any = [];
231
232      this.days_week.forEach( (day:any) => {
233          let DAYS_HOURS = this.hours_selecteds.filter((hour_select:any) => hour_select.day_name == day.day);
234          HOUR_SCHEDULES.push({
235              day_name: day.day,
236              children: DAYS_HOURS,
237          });
238      })
239  }

```

```

admin clinica > src > app > medical > doctors > edit-doctor > 18 edit-doctor.component.ts > 14 EditDoctorComponent > 6 save
18 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_selectedts.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
202 if(INDEX != -1){
203 this.hours_selectedts.splice(INDEX,1);
204 }else{
205 this.hours_selectedts.push({
206 "day": day,
207 "day_name": day.day,
208 "hours_day": hours_day,
209 "hour": hours_day.hour,
210 "grupo": "none",
211 "item": item,
212 });
213 }
214 console.log(this.hours_selectedts);
215 }
216
217 addHourAll(hours_day:any,day:any){
218 let INDEX = this.hours_selectedts.findIndex((hour:any) => hour.day_name == day.day
219 && hour.hour == hours_day.hour && hour.grupo == "all");
220
221 let COUNT_SELECTED = this.hours_selectedts.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_selectedts.findIndex((hour:any) => hour.day_name == day.day

```

```

admin clinica > src > app > medical > doctors > edit-doctor > 18 edit-doctor.component.ts > 14 EditDoctorComponent > 6 save
18 export class EditDoctorComponent {
217 addHourAll(hours_day:any,day:any){
218 let INDEX = this.hours_selectedts.findIndex((hour:any) => hour.day_name == day.day
219 && hour.hour == hours_day.hour && hour.grupo == "all");
220
221 let COUNT_SELECTED = this.hours_selectedts.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_selectedts.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
229 if(INDEX != -1){
230 this.hours_selectedts.splice(INDEX,1);
231 }
232 });
233 }else{
234 hours_day.items.forEach((item:any) => {
235 let INDEX = this.hours_selectedts.findIndex((hour:any) => hour.day_name == day.day
236 && hour.hour == hours_day.hour
237 && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
238
239 if(INDEX != -1){
240 this.hours_selectedts.splice(INDEX,1);
241 }
242 this.hours_selectedts.push({
243 "day": day,
244 "day_name": day.day,
245 "hours_day": hours_day,
246 "hour": hours_day.hour,
247 "grupo": "all",
248 "item": item,
249 });
250 });
251 console.log(this.hours_selectedts);
252 }
253 }
254
255 addHourAllDay($event:any,hours_day:any){
256
257 let INDEX = this.hours_selectedts.findIndex((hour:any) => hour.hour == hours_day.hour);
258
259 if(INDEX != -1 && !$event.currentTarget.checked){
260 this.days_week.forEach((day) => {
261 this.hours_selectedts.forEach((item:any) => {
262 let INDEX = this.hours_selectedts.findIndex((hour:any) => hour.day_name == day.day
263 && hour.hour == hours_day.hour
264 && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
265
266 if(INDEX != -1){
267 this.hours_selectedts.splice(INDEX,1);
268 }
269 });
270 }
271 }else{
272 this.days_week.forEach((day) => {
273 hours_day.items.forEach((item:any) => {
274 let INDEX = this.hours_selectedts.findIndex((hour:any) => hour.day_name == day.day
275 && hour.hour == hours_day.hour
276 && 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
10 export class EditDoctorComponent {
11   addHourAllDay(sequelize:any, hours_day:any){
12     this.days_week.forEach( day => {
13       hours_day.items.forEach( item:any) => {
14         if(INDEX != -1){
15           this.hours_selected.splice(INDEX,1);
16         }
17       }
18     });
19   }
20   }else{
21     this.days_week.forEach(day => {
22       hours_day.items.forEach( item:any) => {
23         let INDEX = this.hours_selected.findIndex((hour:any) => hour.day_name == day.day
24           && hour.hour == hours_day.hour
25           && hour.item.hour_start == item.hour_start && hour.item.hour_end == item.hour_end);
26         if(INDEX != -1){
27           this.hours_selected.splice(INDEX,1);
28         }
29       }
30     });
31   }
32   }
33   }
34   }
35   }
36   }
37   }
38   }
39   }
40   }
41   }
42   }
43   }
44   }
45   }
46   }
47   }
48   }
49   }
50   }
51   }
52   }
53   }
54   }
55   }
56   }
57   }
58   }
59   }
60   }
61   }
62   }
63   }
64   }
65   }
66   }
67   }
68   }
69   }
70   }
71   }
72   }
73   }
74   }
75   }
76   }
77   }
78   }
79   }
80   }
81   }
82   }
83   }
84   }
85   }
86   }
87   }
88   }
89   }
90   }
91   }
92   }
93   }
94   }
95   }
96   }
97   }
98   }
99   }
100  }
101  }
102  }
103  }
104  }
105  }
106  }
107  }
108  }
109  }
110  }
111  }
112  }
113  }
114  }
115  }
116  }
117  }
118  }
119  }
120  }
121  }
122  }
123  }
124  }
125  }
126  }
127  }
128  }
129  }
130  }
131  }
132  }
133  }
134  }
135  }
136  }
137  }
138  }
139  }
140  }
141  }
142  }
143  }
144  }
145  }
146  }
147  }
148  }
149  }
150  }
151  }
152  }
153  }
154  }
155  }
156  }
157  }
158  }
159  }
160  }
161  }
162  }
163  }
164  }
165  }
166  }
167  }
168  }
169  }
170  }
171  }
172  }
173  }
174  }
175  }
176  }
177  }
178  }
179  }
180  }
181  }
182  }
183  }
184  }
185  }
186  }
187  }
188  }
189  }
190  }
191  }
192  }
193  }
194  }
195  }
196  }
197  }
198  }
199  }
200  }
201  }
202  }
203  }
204  }
205  }
206  }
207  }
208  }
209  }
210  }
211  }
212  }
213  }
214  }
215  }
216  }
217  }
218  }
219  }
220  }
221  }
222  }
223  }
224  }
225  }
226  }
227  }
228  }
229  }
230  }
231  }
232  }
233  }
234  }
235  }
236  }
237  }
238  }
239  }
240  }
241  }
242  }
243  }
244  }
245  }
246  }
247  }
248  }
249  }
250  }
251  }
252  }
253  }
254  }
255  }
256  }
257  }
258  }
259  }
260  }
261  }
262  }
263  }
264  }
265  }
266  }
267  }
268  }
269  }
270  }
271  }
272  }
273  }
274  }
275  }
276  }
277  }
278  }
279  }
280  }
281  }
282  }
283  }
284  }
285  }
286  }
287  }
288  }
289  }
290  }
291  }
292  }
293  }
294  }
295  }
296  }
297  }
298  }
299  }
300  }
301  }
302  }
303  }
304  }
305  }
306  }
307  }
308  }
309  }
310  }
311  }
312  }
313  }
314  }
315  }
316  }
317  }
318  }
319  }
320  }
321  }
322  }
323  }
324  }
325  }
326  }
327  }
328  }
329  }
330  }
331  }
332  }
333  }
334  }
335  }
336  }
337  }
338  }
339  }
340  }
341  }
342  }
343  }
344  }
345  }
346  }
347  }
348  }
349  }
350  }
351  }
352  }
353  }
354  }
355  }
356  }
357  }
358  }
359  }
360  }
361  }
362  }
363  }
364  }
365  }
366  }
367  }
368  }
369  }
370  }
371  }
372  }
373  }
374  }
375  }
376  }
377  }
378  }
379  }
380  }
381  }
382  }
383  }
384  }
385  }
386  }
387  }
388  }
389  }
390  }
391  }
392  }
393  }
394  }
395  }
396  }
397  }
398  }
399  }
400  }
401  }
402  }
403  }
404  }
405  }
406  }
407  }
408  }
409  }
410  }
411  }
412  }
413  }
414  }
415  }
416  }
417  }
418  }
419  }
420  }
421  }
422  }
423  }
424  }
425  }
426  }
427  }
428  }
429  }
430  }
431  }
432  }
433  }
434  }
435  }
436  }
437  }
438  }
439  }
440  }
441  }
442  }
443  }
444  }
445  }
446  }
447  }
448  }
449  }
450  }
451  }
452  }
453  }
454  }
455  }
456  }
457  }
458  }
459  }
460  }
461  }
462  }
463  }
464  }
465  }
466  }
467  }
468  }
469  }
470  }
471  }
472  }
473  }
474  }
475  }
476  }
477  }
478  }
479  }
480  }
481  }
482  }
483  }
484  }
485  }
486  }
487  }
488  }
489  }
490  }
491  }
492  }
493  }
494  }
495  }
496  }
497  }
498  }
499  }
500  }
501  }
502  }
503  }
504  }
505  }
506  }
507  }
508  }
509  }
510  }
511  }
512  }
513  }
514  }
515  }
516  }
517  }
518  }
519  }
520  }
521  }
522  }
523  }
524  }
525  }
526  }
527  }
528  }
529  }
530  }
531  }
532  }
533  }
534  }
535  }
536  }
537  }
538  }
539  }
540  }
541  }
542  }
543  }
544  }
545  }
546  }
547  }
548  }
549  }
550  }
551  }
552  }
553  }
554  }
555  }
556  }
557  }
558  }
559  }
560  }
561  }
562  }
563  }
564  }
565  }
566  }
567  }
568  }
569  }
570  }
571  }
572  }
573  }
574  }
575  }
576  }
577  }
578  }
579  }
580  }
581  }
582  }
583  }
584  }
585  }
586  }
587  }
588  }
589  }
590  }
591  }
592  }
593  }
594  }
595  }
596  }
597  }
598  }
599  }
600  }
601  }
602  }
603  }
604  }
605  }
606  }
607  }
608  }
609  }
610  }
611  }
612  }
613  }
614  }
615  }
616  }
617  }
618  }
619  }
620  }
621  }
622  }
623  }
624  }
625  }
626  }
627  }
628  }
629  }
630  }
631  }
632  }
633  }
634  }
635  }
636  }
637  }
638  }
639  }
640  }
641  }
642  }
643  }
644  }
645  }
646  }
647  }
648  }
649  }
650  }
651  }
652  }
653  }
654  }
655  }
656  }
657  }
658  }
659  }
660  }
661  }
662  }
663  }
664  }
665  }
666  }
667  }
668  }
669  }
670  }
671  }
672  }
673  }
674  }
675  }
676  }
677  }
678  }
679  }
680  }
681  }
682  }
683  }
684  }
685  }
686  }
687  }
688  }
689  }
690  }
691  }
692  }
693  }
694  }
695  }
696  }
697  }
698  }
699  }
700  }
701  }
702  }
703  }
704  }
705  }
706  }
707  }
708  }
709  }
710  }
711  }
712  }
713  }
714  }
715  }
716  }
717  }
718  }
719  }
720  }
721  }
722  }
723  }
724  }
725  }
726  }
727  }
728  }
729  }
730  }
731  }
732  }
733  }
734  }
735  }
736  }
737  }
738  }
739  }
740  }
741  }
742  }
743  }
744  }
745  }
746  }
747  }
748  }
749  }
750  }
751  }
752  }
753  }
754  }
755  }
756  }
757  }
758  }
759  }
760  }
761  }
762  }
763  }
764  }
765  }
766  }
767  }
768  }
769  }
770  }
771  }
772  }
773  }
774  }
775  }
776  }
777  }
778  }
779  }
780  }
781  }
782  }
783  }
784  }
785  }
786  }
787  }
788  }
789  }
790  }
791  }
792  }
793  }
794  }
795  }
796  }
797  }
798  }
799  }
800  }
801  }
802  }
803  }
804  }
805  }
806  }
807  }
808  }
809  }
810  }
811  }
812  }
813  }
814  }
815  }
816  }
817  }
818  }
819  }
820  }
821  }
822  }
823  }
824  }
825  }
826  }
827  }
828  }
829  }
830  }
831  }
832  }
833  }
834  }
835  }
836  }
837  }
838  }
839  }
840  }
841  }
842  }
843  }
844  }
845  }
846  }
847  }
848  }
849  }
850  }
851  }
852  }
853  }
854  }
855  }
856  }
857  }
858  }
859  }
860  }
861  }
862  }
863  }
864  }
865  }
866  }
867  }
868  }
869  }
870  }
871  }
872  }
873  }
874  }
875  }
876  }
877  }
878  }
879  }
880  }
881  }
882  }
883  }
884  }
885  }
886  }
887  }
888  }
889  }
890  }
891  }
892  }
893  }
894  }
895  }
896  }
897  }
898  }
899  }
900  }
901  }
902  }
903  }
904  }
905  }
906  }
907  }
908  }
909  }
910  }
911  }
912  }
913  }
914  }
915  }
916  }
917  }
918  }
919  }
920  }
921  }
922  }
923  }
924  }
925  }
926  }
927  }
928  }
929  }
930  }
931  }
932  }
933  }
934  }
935  }
936  }
937  }
938  }
939  }
940  }
941  }
942  }
943  }
944  }
945  }
946  }
947  }
948  }
949  }
950  }
951  }
952  }
953  }
954  }
955  }
956  }
957  }
958  }
959  }
960  }
961  }
962  }
963  }
964  }
965  }
966  }
967  }
968  }
969  }
970  }
971  }
972  }
973  }
974  }
975  }
976  }
977  }
978  }
979  }
980  }
981  }
982  }
983  }
984  }
985  }
986  }
987  }
988  }
989  }
990  }
991  }
992  }
993  }
994  }
995  }
996  }
997  }
998  }
999  }
1000 }

```

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 isPermission(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     console.log(resp);
56     this.totalData = resp.users.data.length;

```

```

admin.clinica > src > app > medical > doctors > list-doctor > 18 list-doctor.component.ts > ...
18 export class ListDoctorComponent {
19   private getTableData(): void {
20     this.doctorService.listDoctors().subscribe((res:any) => {
21       this.totalData = res.users.data.length;
22       this.role_generals = res.users.data;
23       this.getTableDataGeneral();
24     });
25   }
26
27   getTableDataGeneral() {
28     this.userslist = [];
29     this.serialNumberArray = [];
30
31     this.role_generals.map((res: any, index: number) => {
32       const serialNumber = index + 1;
33       if (index >= this.skip && serialNumber <= this.limit) {
34         this.userslist.push(res);
35         this.serialNumberArray.push(serialNumber);
36       }
37     });
38     this.dataSource = new MatTableDataSource<any>(this.userslist);
39     this.calculateTotalPages(this.totalData, this.pageSize);
40   }
41
42   selectUser(role:any){
43     this.doctor_selected = role;
44   }
45
46   deleteUser(){
47     this.doctorService.deleteDoctor(this.doctor_selected.id).subscribe((res:any) => {
48       console.log(res);
49       let INDEX = this.userslist.findIndex((item:any) => item.id == this.doctor_selected.id);
50       if(INDEX != -1){
51         this.userslist.splice(INDEX,1);
52
53         $('#delete_patient').hide();
54         $('#delete_patient').removeClass('show');
55         $('#modal-backdrop').remove();
56         $('#body').removeClass();
57         $('#body').removeAttr('style');
58
59         this.doctor_selected = null;
60       }
61     });
62
63     // eslint-disable-next-line @typescript-eslint/no-explicit-any
64     public searchData(value: any): void {
65       this.dataSource.filter = value.trim().toLowerCase();
66       this.userslist = this.dataSource.filteredData;
67     }
68
69     public sortData(sort: any) {
70       const data = this.userslist.slice();
71
72       if (!sort.active || sort.direction === '') {
73         this.userslist = data;
74       } else {
75         this.userslist = data.sort((a:any, b:any) => {
76           // eslint-disable-next-line @typescript-eslint/no-explicit-any
77           const aValue = (a as any)[sort.active];
78           // eslint-disable-next-line @typescript-eslint/no-explicit-any
79           const bValue = (b as any)[sort.active];
80           return (aValue < bValue ? -1 : 1) * (sort.direction === 'asc' ? 1 : -1);
81         });
82       }
83     }
84
85     public getMoreData(event: string): void {
86       if (event == 'next') {
87         this.currentPage++;
88         this.pageIndex = this.currentPage - 1;
89         this.limit += this.pageSize;
90         this.skip = this.pageSize * this.pageIndex;
91         this.getTableDataGeneral();
92       } else if (event == 'previous') {
93         this.currentPage--;
94         this.pageIndex = this.currentPage - 1;
95         this.limit -= this.pageSize;
96         this.skip = this.pageSize * this.pageIndex;
97         this.getTableDataGeneral();
98       }
99     }
100
101     public moveToPage(pageNumber: number): void {
102       this.currentPage = pageNumber;
103       this.skip = this.pageSelection[pageNumber - 1].skip;
104       this.limit = this.pageSelection[pageNumber - 1].limit;
105       if (pageNumber > this.currentPage) {
106         this.pageIndex = pageNumber - 1;
107       } else if (pageNumber < this.currentPage) {
108         this.pageIndex = pageNumber + 1;
109       }
110       this.getTableDataGeneral();
111     }
112
113     public PageSize(): void {
114       this.pageSelection = [];
115       this.limit = this.pageSize;
116       this.skip = 0;
117       this.currentPage = 1;
118       this.searchDataValue = '';
119       this.getTableDataGeneral();
120     }
121
122     private calculateTotalPages(totalData: number, pageSize: number): void {
123       this.pageNumberArray = [];
124       this.totalPages = totalData / pageSize;
125     }

```

```

admin.clinica > src > app > medical > doctors > list-doctor > 18 list-doctor.component.ts > ...
18 export class ListDoctorComponent {
19
20   public sortData(sort: any) {
21     const data = this.userslist.slice();
22
23     if (!sort.active || sort.direction === '') {
24       this.userslist = data;
25     } else {
26       this.userslist = data.sort((a:any, b:any) => {
27         // eslint-disable-next-line @typescript-eslint/no-explicit-any
28         const aValue = (a as any)[sort.active];
29         // eslint-disable-next-line @typescript-eslint/no-explicit-any
30         const bValue = (b as any)[sort.active];
31         return (aValue < bValue ? -1 : 1) * (sort.direction === 'asc' ? 1 : -1);
32       });
33     }
34   }
35
36   public getMoreData(event: string): void {
37     if (event == 'next') {
38       this.currentPage++;
39       this.pageIndex = this.currentPage - 1;
40       this.limit += this.pageSize;
41       this.skip = this.pageSize * this.pageIndex;
42       this.getTableDataGeneral();
43     } else if (event == 'previous') {
44       this.currentPage--;
45       this.pageIndex = this.currentPage - 1;
46       this.limit -= this.pageSize;
47       this.skip = this.pageSize * this.pageIndex;
48       this.getTableDataGeneral();
49     }
50   }
51
52   public moveToPage(pageNumber: number): void {
53     this.currentPage = pageNumber;
54     this.skip = this.pageSelection[pageNumber - 1].skip;
55     this.limit = this.pageSelection[pageNumber - 1].limit;
56     if (pageNumber > this.currentPage) {
57       this.pageIndex = pageNumber - 1;
58     } else if (pageNumber < this.currentPage) {
59       this.pageIndex = pageNumber + 1;
60     }
61     this.getTableDataGeneral();
62   }
63
64   public PageSize(): void {
65     this.pageSelection = [];
66     this.limit = this.pageSize;
67     this.skip = 0;
68     this.currentPage = 1;
69     this.searchDataValue = '';
70     this.getTableDataGeneral();
71   }
72
73   private calculateTotalPages(totalData: number, pageSize: number): void {
74     this.pageNumberArray = [];
75     this.totalPages = totalData / pageSize;
76   }

```

```

admin_clinica > src > app > medical > doctors > list-doctor > 18 list-doctor.component.ts > ...
10 export class ListDoctorComponent {
126 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 }
142 public moveToPage(pageNumber: number): void {
143     this.currentPage = pageNumber;
144     this.skip = this.pageSelection[pageNumber - 1].skip;
145     this.limit = this.pageSelection[pageNumber - 1].limit;
146     if (pageNumber > this.currentPage) {
147         this.pageIndex = pageNumber - 1;
148     } else if (pageNumber < this.currentPage) {
149         this.pageIndex = pageNumber + 1;
150     }
151     this.getTableDataGeneral();
152 }
153 }
154 public PageSize(): void {
155     this.pageSelection = [];
156     this.limit = this.pageSize;
157     this.skip = 0;
158     this.currentPage = 1;
159     this.searchDataValue = '';
160     this.getTableDataGeneral();
161 }
162 }
163 private calculateTotalPages(totalData: number, pageSize: number): void {
164     this.pageNumberArray = [];
165     this.totalPages = totalData / pageSize;
166     if (this.totalPages % 1 != 0) {
167         this.totalPages = Math.trunc(this.totalPages + 1);
168     }
169     /* eslint no-var: off */
170     for (var i = 1; i <= this.totalPages; i++) {
171         const limit = pageSize * i;
172         const skip = limit - pageSize;
173         this.pageNumberArray.push(i);
174         this.pageSelection.push({ skip: skip, limit: limit });
175     }
176     // 1
177     // 8 - 18
178     // 2
179     // 18 - 20
180 }
181 }
182 }

```

Anexo P

Funcionalidad de Añadir Paciente

```

admin_clinica > src > app > medical > patient-m > add-patient-m > 18 add-patient-m.component.ts > 18 AddPatientMComponent > 18 save
1 import { Component } from '@angular/core';
2 import { PatientMService } from '../service/patient-m.service';
3
4 @Component({
5     selector: 'app-add-patient-m',
6     templateUrl: './add-patient-m.component.html',
7     styleUrls: ['./add-patient-m.component.scss']
8 })
9 export class AddPatientMComponent {
10     public selectedValue!: string;
11     public name:string = '';
12     public surname:string = '';
13     public mobile:string = '';
14     public email:string = '';
15
16     public birth_date:string = '';
17     public gender:number = 1;
18     public education:string = '';
19     public address:string = '';
20
21     public antecedent_family:string = '';
22     public antecedent_personal:string = '';
23     public antecedent_allergic:string = '';
24
25     public name_companion:string = '';
26     public surname_companion:string = '';
27     public mobile_companion:string = '';
28     public relationship_companion:string = '';
29
30     public name_responsible:string = '';
31     public surname_responsible:string = '';
32     public mobile_responsible:string = '';
33     public relationship_responsible:string = '';
34
35     public current_disease:string = '';
36
37     public ta :number = 0;
38     public temperatura :number = 0;
39     public fc :number = 0;
40     public fr :number = 0;
41     public peso :number = 0;
42
43     public n_document:any = null;
44
45     public roles:any = [];
46
47     public FILE_AVATAR:any;
48     public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';
49
50     public text_success:string = '';
51     public text_validation:string = '';
52     constructor(
53         private patientService: PatientMService,
54     ) {
55     }
56 }
57 ngOnInit(): void {
58     //called after the constructor, initializing input properties, and the first call to ngOnChanges.

```

```

admin clinica > src > app > medical > patient-m > add-patient-m > add-patient-m.component.ts > AddPatientMComponent > save
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.listonfig().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);
    }
  }
}

```

```

admin clinica > src > app > medical > patient-m > add-patient-m > add-patient-m.component.ts > AddPatientMComponent > save
export class AddPatientMComponent {
  save(){
    113
    114
    115     formData.append("name_companion",this.name_companion);
    116     formData.append("surname_companion",this.surname_companion);
    117     if(this.mobile_companion){
    118       formData.append("mobile_companion",this.mobile_companion);
    119     }
    120     if(this.relationship_companion){
    121       formData.append("relationship_companion",this.relationship_companion);
    122     }
    123     if(this.name_responsible){
    124       formData.append("name_responsible",this.name_responsible);
    125     }
    126     if(this.surname_responsible){
    127       formData.append("surname_responsible",this.surname_responsible);
    128     }
    129     if(this.mobile_responsible){
    130       formData.append("mobile_responsible",this.mobile_responsible);
    131     }
    132     if(this.relationship_responsible){
    133       formData.append("relationship_responsible",this.relationship_responsible);
    134     }
    135     if(this.current_disease){
    136       formData.append("current_disease",this.current_disease);
    137     }
    138
    139     formData.append("ta",this.ta);
    140     formData.append("temperatura",this.temperatura);
    141     formData.append("fc",this.fc);
    142     formData.append("fr",this.fr);
    143     formData.append("peso",this.peso);
    144     this.patientService.registerPatient(formData).subscribe((resp:any) => {
    145       console.log(resp);
    146
    147       if(resp.message == 403){
    148         this.text_validation = resp.message_text;
    149       }else{
    150         this.text_success = 'El paciente ha sido registrado correctamente';
    151
    152         this.name = '';
    153         this.surname = '';
    154         this.email = '';
    155         this.mobile = '';
    156         this.birth_date = '';
    157         this.gender = 1;
    158         this.education = '';
    159
    160         this.n_document = '';
    161         this.antecedent_family = '';
    162         this.antecedent_personal = '';
    163         this.antecedent_allergic = '';
    164         this.name_companion = '';
    165         this.surname_companion = '';
    166         this.mobile_companion = '';
    167         this.relationship_companion = '';
    168         this.name_responsible = '';

```

```

admin_clinica > src > app > medical > patient-m > add-patient-m > 18 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 > 18 edit-patient-m.component.ts > ...
1 import { Component } from '@angular/core';
2 import { PatientMService } from '../service/patient-m.service';
3 import { ActivatedRoute } from '@angular/router';
4
5 @Component({
6   selector: 'app-edit-patient-m',
7   templateUrl: './edit-patient-m.component.html',
8   styleUrls: ['./edit-patient-m.component.scss']
9 })
10 export class EditPatientMComponent {
11   public selectedValue !: string ;
12   public name:string = '';
13   public surname:string = '';
14   public mobile:string = '';
15   public email:string = '';
16
17   public birth_date:string = '';
18   public gender:number = 1;
19   public education:string = '';
20   public address:string = '';
21
22   public antecedent_family:string = '';
23   public antecedent_personal:string = '';
24   public antecedent_allergic:string = '';
25
26   public name_companion:string = '';
27   public surname_companion:string = '';
28   public mobile_companion:string = '';
29   public relationship_companion:string = '';
30
31   public name_responsible:string = '';
32   public surname_responsible:string = '';
33   public mobile_responsible:string = '';
34   public relationship_responsible:string = '';
35
36   public current_disease:string = '';
37
38   public ta :number = 0;
39   public temperatura :number = 0;
40   public fc :number = 0;
41   public fr :number = 0;
42   public peso :number = 0;
43
44   public n_document:any = null;
45
46   public roles:any = [];
47
48   public FILE_AVATAR:any;
49   public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';
50
51   public text_success:string = '';
52   public text_validation:string = '';
53
54   public patient_id:any;
55   constructor(
56     public patientService: PatientMService,
57     public activatedRoute: ActivatedRoute,
58   ) {

```

```

admin clínica > src > app > medical > patient-m > edit-patient-m > 78 edit-patient-m-components > ...
18  export class EditPatientMComponent {
19  constructor() {
20  }
21  }
22  ngOnInit(): void {
23  //called after the constructor, initializing input properties, and the first call to ngOnChanges.
24  //Add 'implements OnInit' to the class.
25  // this.patientService.listConfig().subscribe((resp:any) => {
26  //   console.log(resp);
27  //   this.roles = resp.roles;
28  // });
29  this.activatedRoute.params.subscribe((resp:any) => {
30  this.patient_id = resp.id;
31  });
32  this.patientService.showPatient(this.patient_id).subscribe((resp:any) => {
33  console.log(resp);
34  });
35  this.name = resp.patient.name;
36  this.surname = resp.patient.surname;
37  this.email = resp.patient.email;
38  this.mobile = resp.patient.mobile;
39  this.n_document = resp.patient.n_document;
40  this.birth_date = resp.patient.birth_date ? new Date(resp.patient.birth_date).toISOString() : '';
41  this.gender = resp.patient.gender;
42  this.education = resp.patient.education;
43  this.address = resp.patient.address;
44  this.antecedent_family = resp.patient.antecedent_family;
45  this.antecedent_personal = resp.patient.antecedent_personal;
46  this.antecedent_allergic = resp.patient.antecedent_allergic;
47  };
48  this.name_companion = resp.patient.person.name_companion;
49  this.surname_companion = resp.patient.person.surname_companion;
50  this.mobile_companion = resp.patient.person.mobile_companion;
51  this.relationship_companion = resp.patient.person.relationship_companion;
52  this.name_responsible = resp.patient.person.name_responsible;
53  this.surname_responsible = resp.patient.person.surname_responsible;
54  this.mobile_responsible = resp.patient.person.mobile_responsible;
55  this.relationship_responsible = resp.patient.person.relationship_responsible;
56  };
57  this.current_disease = resp.patient.current_disease;
58  this.ta = resp.patient.ta;
59  this.temperatura = resp.patient.temperatura;
60  this.fc = resp.patient.fc;
61  this.fr = resp.patient.fr;
62  this.peso = resp.patient.peso;
63  this.IMAGEN_PREVIZUALIZA = resp.patient.avatar;
64  });
65  }
66  save(){
67  this.text_validation = '';
68  if([this.name || this.n_document || this.surname || this.mobile || this.email]{
69  this.text_validation = "LOS CAMPOS SON NECESARIOS (Nombre, Apellido, Nº de documento, Teléfono Móvil)";
70  return;
71  }
72  if([this.name_companion || this.surname_companion || this.mobile_companion || this.relationship_companion]{
73  this.text_validation = "LOS DATOS DEL ACOMPAÑANTE SON OBLIGATORIOS";
74  }

```

```

admin clínica > src > app > medical > patient-m > edit-patient-m > 78 edit-patient-m-components > ...
18  export class EditPatientMComponent {
19  constructor() {
20  }
21  }
22  ngOnInit(): void {
23  //called after the constructor, initializing input properties, and the first call to ngOnChanges.
24  //Add 'implements OnInit' to the class.
25  // this.patientService.listConfig().subscribe((resp:any) => {
26  //   console.log(resp);
27  //   this.roles = resp.roles;
28  // });
29  this.activatedRoute.params.subscribe((resp:any) => {
30  this.patient_id = resp.id;
31  });
32  this.patientService.showPatient(this.patient_id).subscribe((resp:any) => {
33  console.log(resp);
34  });
35  this.name = resp.patient.name;
36  this.surname = resp.patient.surname;
37  this.email = resp.patient.email;
38  this.mobile = resp.patient.mobile;
39  this.n_document = resp.patient.n_document;
40  this.birth_date = resp.patient.birth_date ? new Date(resp.patient.birth_date).toISOString() : '';
41  this.gender = resp.patient.gender;
42  this.education = resp.patient.education;
43  this.address = resp.patient.address;
44  this.antecedent_family = resp.patient.antecedent_family;
45  this.antecedent_personal = resp.patient.antecedent_personal;
46  this.antecedent_allergic = resp.patient.antecedent_allergic;
47  };
48  this.name_companion = resp.patient.person.name_companion;
49  this.surname_companion = resp.patient.person.surname_companion;
50  this.mobile_companion = resp.patient.person.mobile_companion;
51  this.relationship_companion = resp.patient.person.relationship_companion;
52  this.name_responsible = resp.patient.person.name_responsible;
53  this.surname_responsible = resp.patient.person.surname_responsible;
54  this.mobile_responsible = resp.patient.person.mobile_responsible;
55  this.relationship_responsible = resp.patient.person.relationship_responsible;
56  };
57  this.current_disease = resp.patient.current_disease;
58  this.ta = resp.patient.ta;
59  this.temperatura = resp.patient.temperatura;
60  this.fc = resp.patient.fc;
61  this.fr = resp.patient.fr;
62  this.peso = resp.patient.peso;
63  this.IMAGEN_PREVIZUALIZA = resp.patient.avatar;
64  });
65  }
66  save(){
67  this.text_validation = "LOS DATOS DEL ACOMPAÑANTE SON OBLIGATORIOS";
68  return;
69  }
70  if([this.ta || this.temperatura || this.fc || this.fr || this.peso]{
71  this.text_validation = "LOS SIGNOS VITALES SON OBLIGATORIO";
72  return;
73  }
74  }
75  console.log(this.selectedValue);
76  let formData = new FormData();
77  formData.append("name",this.name);
78  formData.append("surname",this.surname);
79  if(this.email){
80  formData.append("email",this.email);
81  }
82  formData.append("mobile",this.mobile);
83  formData.append("n_document",this.n_document);
84  if(this.birth_date){
85  formData.append("birth_date",this.birth_date);
86  }
87  formData.append("gender",this.gender+"");
88  if(this.education){
89  formData.append("education",this.education);
90  }
91  if(this.address){
92  formData.append("address",this.address);
93  }
94  if(this.FILE_AVATAR){
95  formData.append("imagen",this.FILE_AVATAR);
96  }
97  if(this.antecedent_family){
98  formData.append("antecedent_family",this.antecedent_family);
99  }
100  if(this.antecedent_personal){
101  formData.append("antecedent_personal",this.antecedent_personal);
102  }
103  if(this.antecedent_allergic){
104  formData.append("antecedent_allergic",this.antecedent_allergic);
105  }
106  }
107  formData.append("name_companion",this.name_companion);
108  formData.append("surname_companion",this.surname_companion);
109  if(this.mobile_companion){
110  formData.append("mobile_companion",this.mobile_companion);
111  }
112  if(this.relationship_companion){
113  formData.append("relationship_companion",this.relationship_companion);
114  }
115  if(this.name_responsible){
116  formData.append("name_responsible",this.name_responsible);
117  }
118  if(this.surname_responsible){
119  formData.append("surname_responsible",this.surname_responsible);
120  }
121  }

```

```

admin_clinica > src > app > medical > patient-m > edit-patient-m > 78 edit-patient-m.component.ts > ...
18
export class EditPatientMComponent {
186
  save(){
184
185     formData.append("name_companion",this.name_companion);
186     formData.append("surname_companion",this.surname_companion);
187     if(this.mobile_companion){
188       formData.append("mobile_companion",this.mobile_companion);
189     }
190     if(this.relationship_companion){
191       formData.append("relationship_companion",this.relationship_companion);
192     }
193     if(this.name_responsible){
194       formData.append("name_responsible",this.name_responsible);
195     }
196     if(this.surname_responsible){
197       formData.append("surname_responsible",this.surname_responsible);
198     }
199     if(this.mobile_responsible){
200       formData.append("mobile_responsible",this.mobile_responsible);
201     }
202     if(this.relationship_responsible){
203       formData.append("relationship_responsible",this.relationship_responsible);
204     }
205     if(this.current_disease){
206       formData.append("current_disease",this.current_disease);
207     }
208
209     formData.append("ta",this.ta+"");
210     formData.append("temperatura",this.temperatura+"");
211     formData.append("fc",this.fc+"");
212     formData.append("fr",this.fr+"");
213     formData.append("peso",this.peso+"");
214     this.patientService.updatePatient(this.patient_id,formData).subscribe((resp:any) => {
215       console.log(resp);
216
217       if(resp.message == 403){
218         this.text_validation = resp.message_text;
219       }else{
220         this.text_success = 'El paciente ha sido editado correctamente';
221       }
222     })
223   }
224
225   loadFile($event:any){
226     if($event.target.files[0].type.indexOf("image") < 0){
227       // alert("SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN");
228       this.text_validation = "SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN";
229       return;
230     }
231     this.text_validation = '';
232     this.FILE_AVATAR = $event.target.files[0];
233     let reader = new FileReader();
234     reader.readAsDataURL(this.FILE_AVATAR);
235     reader.onloadend = () => this.IMAGEN_PREVIZUALIZA = reader.result;
236   }
237
238 }
239

```

Anexo R

Funcionalidad de Listado de Pacientes

```

admin_clinica > src > app > medical > patient-m > list-patient-m > 76 list-patient-m.component.ts > 76 ListPatientMComponent > ngOnInit
1
import { Component } from '@angular/core';
2
import { PatientService } from '../services/patient-m.service';
3
import { MatTableDataSource } from '@angular/material/table';
4
5
@Component({
6
  selector: 'app-list-patient-m',
7
  templateUrl: './list-patient-m.component.html',
8
  styleUrls: ['./list-patient-m.component.scss']
9
})
10
export class ListPatientMComponent {
11
12   public patientsList:any = [];
13   dataSource: MatTableDataSource<any>;
14
15   public showFilter = false;
16   public searchDataValue = '';
17   public lastIndex = 0;
18   public pageSize = 20;
19   public totalRows = 0;
20   public skip = 0; //MAX
21   public limit: number = this.pageSize; //MAX
22   public pageIndex = 0;
23   public serialNumberArray: Array<number> = [];
24   public currentPage = 1;
25   public pageNumberArray: Array<number> = [];
26   public pageSelection: Array<any> = [];
27   public totalPages = 0;
28
29   public patient_generals:any = [];
30   public patient_selected:any;
31   public user:any;
32   constructor() {
33     this.patientService = PatientMService;
34   }
35
36
37   ngOnInit() {
38     this.getTableData();
39     this.user = this.patientService.authService.user;
40   }
41
42   isPermission(permission:string){
43     if(this.user.roles.includes('Super-Admin')){
44       return true;
45     }
46     if(this.user.permissions.includes(permission)){
47       return true;
48     }
49     return false;
50   }
51   private getTableData(page=1): void {
52     this.patientsList = [];
53     this.serialNumberArray = [];
54
55     this.patientService.listPatients(page,this.searchDataValue).subscribe((resp:any) => {
56       console.log(resp);
57

```

```

admin_clinica > src > app > medical > patient-m > list-patient-m > 18 list-patient-m.component.ts > 18 ListPatientMComponent > ngOnOnInit
18 export class ListPatientMComponent {
19   private getTableData(page=1): void {
20     this.serialNumberArray = [];
21
22     this.patientService.listPatients(page, this.searchDataValue).subscribe((resp:any) => {
23
24       console.log(resp);
25
26       this.totalData = resp.total;
27       this.patientsList = resp.patients.data;
28       // this.getTableDataGeneral();
29       this.dataSource = new MatTableDataSource<any>(this.patientsList);
30       this.calculateTotalPages(this.totalData, this.pageSize);
31     });
32   }
33
34   getTableDataGeneral() {
35     this.patientsList = [];
36     this.serialNumberArray = [];
37
38     this.patient_generals.map((res: any, index: number) => {
39       const serialNumber = index + 1;
40       if (index >= this.skip && serialNumber <= this.limit) {
41
42         this.patientsList.push(res);
43         this.serialNumberArray.push(serialNumber);
44       }
45     });
46     this.dataSource = new MatTableDataSource<any>(this.patientsList);
47     this.calculateTotalPages(this.totalData, this.pageSize);
48   }
49
50   selectUser(rol:any){
51     this.patient_selected = rol;
52   }
53
54   deletePatient(){
55
56     this.patientService.deletePatient(this.patient_selected.id).subscribe((resp:any) => {
57       console.log(resp);
58       let INDEX = this.patientsList.findIndex((item:any) => item.id == this.patient_selected.id);
59       if(INDEX != -1){
60         this.patientsList.splice(INDEX,1);
61
62         $('#delete_patient').hide();
63         $('#delete_patient').removeClass("show");
64         $('#modal-backdrop').remove();
65         $('#body').removeClass();
66         $('#body').removeAttr("style");
67
68         this.patient_selected = null;
69       }
70     });
71   }
72 }
73 // eslint-disable-next-line @typescript-eslint/no-explicit-any

```

```

admin_clinica > src > app > medical > patient-m > list-patient-m > 18 list-patient-m.component.ts > 18 ListPatientMComponent > ngOnOnInit
18 export class ListPatientMComponent {
19   deletePatient(){
20     this.patientService.deletePatient(this.patient_selected.id).subscribe((resp:any) => {
21
22     })
23   }
24   // eslint-disable-next-line @typescript-eslint/no-explicit-any
25   public searchData() {
26     // this.dataSource.filter = value.trim().toLowerCase();
27     // this.patientsList = this.dataSource.filteredData;
28     this.pageSelection = [];
29     this.limit = this.pageSize;
30     this.skip = 0;
31     this.currentPage = 1;
32
33     this.getTableData();
34   }
35
36   public sortData(sort: any) {
37     const data = this.patientsList.slice();
38
39     if (!sort.active || sort.direction === '') {
40       this.patientsList = data;
41     } else {
42       this.patientsList = data.sort((a:any, b:any) => {
43         // eslint-disable-next-line @typescript-eslint/no-explicit-any
44         const aValue = (a as any)[sort.active];
45         // eslint-disable-next-line @typescript-eslint/no-explicit-any
46         const bValue = (b as any)[sort.active];
47         return (aValue < bValue ? -1 : 1) * (sort.direction === 'asc' ? 1 : -1);
48       });
49     }
50   }
51
52   public getMoreData(event: string): void {
53     if (event == 'next') {
54       this.currentPage++;
55       this.pageIndex = this.currentPage - 1;
56       this.limit += this.pageSize;
57       this.skip = this.pageSize * this.pageIndex;
58       this.getTableData(this.currentPage);
59     } else if (event == 'previous') {
60       this.currentPage--;
61       this.pageIndex = this.currentPage - 1;
62       this.limit -= this.pageSize;
63       this.skip = this.pageSize * this.pageIndex;
64       this.getTableData(this.currentPage);
65     }
66   }
67
68   public moveToPage(pageNumber: number): void {
69     this.currentPage = pageNumber;
70     this.skip = this.pageSize * (pageNumber - 1);
71     this.limit = this.pageSize * (pageNumber - 1);
72     if (pageNumber > this.currentPage) {
73       this.pageIndex = pageNumber - 1;
74     } else if (pageNumber < this.currentPage) {
75       this.pageIndex = pageNumber + 1;
76     }
77   }
78 }

```

```

admin clinica > src > app > medical > patient-m > list-patient-m > 78 list-patient-m.component.ts > 4 ListPatientMComponent > 6
10 export class ListPatientMComponent {
115 public getMoreData(event: string): void {
140   this.skip = this.pageSize * this.pageIndex;
141   this.getTableData(this.currentPage);
142 } else if (event == 'previous') {
143   this.currentPage--;
144   this.pageIndex = this.currentPage - 1;
145   this.limit = this.pageSize;
146   this.skip = this.pageSize * this.pageIndex;
147   this.getTableData(this.currentPage);
148 }
149 }
150
151 public moveToPage(pageNumber: number): void {
152   this.currentPage = pageNumber;
153   this.skip = this.pageSelection[pageNumber - 1].skip;
154   this.limit = this.pageSelection[pageNumber - 1].limit;
155   if (pageNumber > this.currentPage) {
156     this.pageIndex = pageNumber - 1;
157   } else if (pageNumber < this.currentPage) {
158     this.pageIndex = pageNumber + 1;
159   }
160   this.getTableData(this.currentPage);
161 }
162
163 public PageSize(): void {
164   this.pageSelection = [];
165   this.limit = this.pageSize;
166   this.skip = 0;
167   this.currentPage = 1;
168   this.searchDataValue = '';
169   this.getTableData();
170 }
171
172 private calculateTotalPages(totalData: number, pageSize: number): void {
173   this.pageNumberArray = [];
174   this.totalPages = totalData / pageSize;
175   if (this.totalPages % 1 != 0) {
176     this.totalPages = Math.trunc(this.totalPages + 1); //10.6 o 10.9 11
177   }
178   /* eslint no-var: off */
179   for (var i = 1; i <= this.totalPages; i++) {
180     const limit = pageSize * i;
181     const skip = limit - pageSize;
182     this.pageNumberArray.push(i);
183     this.pageSelection.push({ skip: skip, limit: limit });
184     // 1
185     // 0 - 10
186     // 2
187     // 10 - 20
188     // 3
189     // 20 - 30
190   }
191 }
192
193 }
194

```

Anexo S

Funcionalidad del Perfil del Paciente

```

admin clinica > src > app > medical > patient-m > patient-m-profile > 78 patient-m-profile.component.ts > 4 PatientMProfileComponent
1 import { Component } from '@angular/core';
2 import { ActivatedRoute } from '@angular/router';
3 import { PatientMService } from '../service/patient-m.service';
4
5 @Component({
6   selector: 'app-patient-m-profile',
7   templateUrl: './patient-m-profile.component.html',
8   styleUrls: ['./patient-m-profile.component.scss']
9 })
10 export class PatientMProfileComponent {
11   patientProfile:any = [];
12   option_selected:number = 1;
13
14   patient_id:string = '';
15   num_appointment:number = 0;
16   money_of_appointments:number = 0;
17   num_appointment_pendings:number = 0;
18   patient_selected:any;
19   appointment_pendings:any = [];
20   appointments:any = [];
21
22   constructor(
23     public patientService: PatientMService,
24     public activatedRoute: ActivatedRoute,
25   ) {
26   }
27
28 }
29
30 ngOnInit(): void {
31   this.activatedRoute.params.subscribe((resp: any) => {
32     console.log(resp);
33     this.patient_id = resp.id;
34     // Asegurate de llamar al servicio después de obtener el patient_id
35     this.patientService.profilePatient(this.patient_id).subscribe((resp: any) => {
36       console.log(resp);
37       this.num_appointment = resp.num_appointment;
38       this.money_of_appointments = resp.money_of_appointments;
39       this.num_appointment_pendings = resp.num_appointment_pendings;
40       this.patient_selected = resp.patient;
41       this.appointment_pendings = resp.appointment_pendings.data;
42       this.appointments = resp.appointments;
43     }, error => {
44       console.error('ERROR:', error); // Manejo de errores de la llamada
45     });
46   });
47
48   optionSelected(value:number){
49     this.option_selected = value;
50   }
51 }
52
53

```

Anexo T

Funcionalidad de Añadir un Rol

```

admin clinica > src > app > medical > roles > add-role-user > 78 add-role-user.components.ts > 79 AddRoleUserComponent > save > 80 subscribe() ca
1  import { Component } from '@angular/core';
2  import { DataService } from 'src/app/shared/data/data.service';
3  import { RolesService } from '../service/roles.service';
4
5  @Component({
6    selector: 'app-add-role-user',
7    templateUrl: './add-role-user.component.html',
8    styleUrls: ['./add-role-user.component.scss']
9  })
10 export class AddRoleUserComponent {
11
12   sidebar:any = [];
13   name:string = '';
14   permissions:any = [];
15   valid_form: boolean = false;
16   valid_form_success: boolean = false;
17   text_validation:any = null;
18   constructor(
19     public DataService: DataService,
20     public RolesService: RolesService,
21   ) {
22
23   }
24   ngOnInit(): void {
25     //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
26     //Add 'implements OnInit' to the class.
27     this.sidebar = this.DataService.sidebar[0].menu;
28   }
29
30   addPermission(subMenu:any){
31     if(subMenu.permission){
32       let INDEX = this.permissions.findIndex((item:any) => item == subMenu.permission);
33       if(INDEX != -1){
34         this.permissions.splice(INDEX,1);
35       }else{
36         this.permissions.push(subMenu.permission);
37       }
38       console.log(this.permissions);
39     }
40   }
41

```

```

41
42   save(){
43     this.valid_form = false;
44     if(!this.name || this.permissions.length == 0){
45       this.valid_form = true;
46       return;
47     }
48     let data = {
49       name: this.name,
50       permissions:this.permissions,
51     };
52     this.valid_form_success = false;
53     this.text_validation = null;
54     this.RolesService.storeRoles(data).subscribe((resp:any) => {
55       console.log(resp);
56       if(resp.message == 403){
57         this.text_validation = resp.message_text;
58       }else{
59         this.name = '';
60         this.permissions = [];
61         this.valid_form_success = true;
62
63         let SIDE_BAR = this.sidebar;
64         this.sidebar = [];
65         setTimeout(() => {
66           this.sidebar =SIDE_BAR;
67         }, 50);
68       }
69     });
70   }
71 }
72 }
73

```

Anexo U

Funcionalidad de Editar un Rol

```

admin_clinica > src > app > medical > roles > edit-role-user > 78 edit-role-user.component.ts > ...
1  import { Component } from '@angular/core';
2  import { RoleService } from '../service/roles.service';
3  import { DataService } from 'src/app/shared/data/data.service';
4  import { ActivatedRoute } from '@angular/router';
5
6
7  @Component({
8    selector: 'app-edit-role-user',
9    templateUrl: './edit-role-user.component.html',
10   styleUrls: ['./edit-role-user.component.scss']
11 })
12 export class EditRoleUserComponent {
13
14   sidebar:any = [];
15   name:string = '';
16   permissions:any = [];
17   valid_form: boolean = false;
18   valid_form_success: boolean = false;
19   text_validation:any = null;
20
21   role_id:any;
22   constructor(
23     public DataService: DataService ,
24     public RoleService: RoleService,
25     public activatedRoute: ActivatedRoute,
26   ) {
27   }
28   ngOnInit(): void {
29     //called after the constructor, initializing input properties, and the first call to ngOnChanges.
30     //Add 'implements OnInit' to the class.
31     this.sidebar = this.DataService.sidebar[0].menu;
32     this.activatedRoute.params.subscribe((resp:any) => {
33       this.role_id = resp.id;
34     })
35     this.showRole();
36   }
37
38   showRole(){
39     this.RoleService.showRoles(this.role_id).subscribe((resp:any) => {
40       console.log(resp);
41       this.name = resp.name;
42       this.permissions = resp.permission_pluck;
43     })
44   }
45
46   addPermission(subMenu:any){
47     if(subMenu.permission){
48       let INDEX = this.permissions.findIndex((item:any) => item == subMenu.permission);
49       if(INDEX != -1){
50         this.permissions.splice(INDEX,1);
51       }else{
52         this.permissions.push(subMenu.permission);
53       }
54       console.log(this.permissions);
55     }
56   }
57 }

```

```

58 save(){
59   this.valid_form = false;
60   if(!this.name || this.permissions.length == 0){
61     this.valid_form = true;
62     return;
63   }
64   let data = {
65     name: this.name,
66     permissions:this.permissions,
67   };
68   this.valid_form_success = false;
69   this.text_validation = null;
70   this.RoleService.editRoles(data,this.role_id).subscribe((resp:any) => {
71     console.log(resp);
72     if(resp.message == 403){
73       this.text_validation = resp.message_text;
74       return ;
75     }
76     this.valid_form_success = true;
77   })
78 }
79
80 }
81

```

Anexo V

Funcionalidad de Añadir Especialidad

```

admin_clinica > src > app > medical > specialitie > add-specialitie > 78 add-specialitie.components > ...
Click to add a breakpoint
  3 import { Component } from '@angular/core';
  4 import { SpecialitieService } from '../service/specialitie.service';
  5
  6 @Component({
  7   selector: 'app-add-specialitie',
  8   templateUrl: './add-specialitie.component.html',
  9   styleUrls: ['./add-specialitie.component.scss']
 10 })
 11 export class AddSpecialitieComponent {
 12
 13   name:string = '';
 14   valid_form: boolean = false;
 15   valid_form_success: boolean = false;
 16   text_validation:any = null;
 17
 18   constructor(
 19     public specialitieService: SpecialitieService,
 20   ) {
 21
 22   }
 23
 24   ngOnInit(): void {
 25
 26   }
 27
 28   save(){
 29     this.valid_form = false;
 30     if(!this.name){
 31       this.valid_form = true;
 32       return;
 33     }
 34     let data = {
 35       name: this.name,
 36     };
 37     this.valid_form_success = false;
 38     this.text_validation = null;
 39     this.specialitieService.storeSpecialities(data).subscribe((resp:any) => {
 40       console.log(resp);
 41       if(resp.message == 403){
 42         this.text_validation = resp.message_text;
 43       }else{
 44         this.name = '';
 45         this.valid_form_success = true;
 46       }
 47     })
 48   }
 49 }
 50

```

Anexo W

Funcionalidad de Editar Especialidad

```

admin_clinica > src > app > medical > specialitie > edit-specialitie > 78 edit-specialitie.components > ...
  1 import { Component } from '@angular/core';
  2 import { ActivatedRoute } from '@angular/router';
  3 import { SpecialitieService } from '../service/specialitie.service';
  4
  5 @Component({
  6   selector: 'app-edit-specialitie',
  7   templateUrl: './edit-specialitie.component.html',
  8   styleUrls: ['./edit-specialitie.component.scss']
  9 })
 10 export class EditSpecialitieComponent {
 11
 12   name:string = '';
 13   state:number = 1;
 14   valid_form: boolean = false;
 15   valid_form_success: boolean = false;
 16   text_validation:any = null;
 17
 18   specialitie_id:any;
 19
 20   constructor(
 21     public specialitieService: SpecialitieService,
 22     public activatedRoute: ActivatedRoute,
 23   ) {
 24
 25   }
 26
 27   ngOnInit(): void {
 28     this.activatedRoute.params.subscribe((resp:any) => {
 29       this.specialitie_id = resp.id;
 30     })
 31     this.showSpecialitie();
 32   }
 33
 34   showSpecialitie(){
 35     (property) EditSpecialitieComponent.specialitie_id: any
 36     this.specialitieService.showSpecialities(this.specialitie_id).subscribe((resp:any) => {
 37       console.log(resp);
 38       this.name = resp.name;
 39       this.state = resp.state;
 40     })
 41   }
 42 }
 43

```

Anexo X

Funcionalidad de Listado de Especialidades

```

admin_clinica > src > app > medical > specialitie > list-specialitie > 78 list-specialitie.component.ts > ...
1 import { Component } from '@angular/core';
2 import { SpecialitiesService } from '../service/specialitie.service';
3 import { MatTableDataSource } from '@angular/material/table';
4
5 @Component({
6   selector: 'app-list-specialitie',
7   templateUrl: './list-specialitie.component.html',
8   styleUrls: ['./list-specialitie.component.scss']
9 })
10 export class ListSpecialitieComponent {
11   public specialitiesList: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;//MAX
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 specialitie_generals:any = [];
29   public specialitie_selected:any;
30   public user:any;
31
32   constructor(
33     public specialitiesService: SpecialitiesService,
34   ){
35
36   }
37   ngOnInit() {
38     this.getTableData();
39     this.user = this.specialitiesService.authService.user;
40   }
41   isPermission(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.specialitiesList = [];
52     this.serialNumberArray = [];
53
54     this.specialitiesService.listSpecialities().subscribe((resp:any) => {
55       console.log(resp);
56
57       this.totalData = resp.specialities.length;
58

```

```

admin_clinica > src > app > medical > specialitie > list-specialitie > 78 list-specialitie.component.ts > ...
10 export class ListSpecialitieComponent {
11   isPermission(permission:string){
12
13   }
14   private getTableData(): void {
15     this.specialitiesList = [];
16     this.serialNumberArray = [];
17
18     this.specialitiesService.listSpecialities().subscribe((resp:any) => {
19       console.log(resp);
20
21       this.totalData = resp.specialities.length;
22       this.specialitie_generals = resp.specialities;
23       this.getTableDataGeneral();
24     })
25   }
26
27   }
28   getTableDataGeneral() {
29     this.specialitiesList = [];
30     this.serialNumberArray = [];
31
32     this.specialitie_generals.map((res: any, index: number) => {
33       const serialNumber = index + 1;
34       if (index > this.skip && serialNumber <= this.limit) {
35
36         this.specialitiesList.push(res);
37         this.serialNumberArray.push(serialNumber);
38       }
39     });
40     this.dataSource = new MatTableDataSource<any>(this.specialitiesList);
41     this.calculateTotalPages(this.totalData, this.pageSize);
42   }
43
44   selectSpecialitie(rol:any){
45     this.specialitie_selected = rol;
46   }
47
48   deleteSpecialitie(){
49
50   }
51   this.specialitiesService.deleteSpecialities(this.specialitie_selected.id).subscribe((resp:any) => {
52     console.log(resp);
53     let INDEX = this.specialitiesList.findIndex((item:any) => item.id == this.specialitie_selected.id);
54     if (INDEX != -1){
55       this.specialitiesList.splice (INDEX,1);
56     }
57     $(' #delete_patient' ).hide();
58     $(' #delete_patient' ).removeClass("show");
59     $(' #modal-backdrop' ).remove();
60     $(' #body' ).removeClass();
61     $(' #body' ).removeAttr("style");
62
63     this.specialitie_selected = null;
64   }
65 }
66 // eslint-disable-next-line @typescript-eslint/no-explicit-any

```

```

admin_clinica > src > app > medical > specialitie > list-specialitie > 18 list-specialitie.component.ts > ...
18 export class ListSpecialitieComponent {
19
203 }
204 // eslint-disable-next-line @typescript-eslint/no-explicit-any
205 public searchData(value: any): void {
206   this.dataSource.filter = value.trim().toLowerCase();
207   this.specialitiesList = this.dataSource.filteredData;
208 }
209
210 public sortData(sort: any) {
211   const data = this.specialitiesList.slice();
212
213   if (!sort.active || sort.direction === '') {
214     this.specialitiesList = data;
215   } else {
216     this.specialitiesList = data.sort((a: any, b: any) => {
217       // eslint-disable-next-line @typescript-eslint/no-explicit-any
218       const aValue = (a as any)[sort.active];
219       // eslint-disable-next-line @typescript-eslint/no-explicit-any
220       const bValue = (b as any)[sort.active];
221       return (aValue < bValue ? -1 : 1) * (sort.direction === 'asc' ? 1 : -1);
222     });
223   }
224 }
225
226 public getMoreData(event: string): void {
227   if (event == 'next') {
228     this.currentPage++;
229     this.pageIndex = this.currentPage - 1;
230     this.limit += this.pageSize;
231     this.skip = this.pageSize * this.pageIndex;
232     this.getTableDataGeneral();
233   } else if (event == 'previous') {
234     this.currentPage--;
235     this.pageIndex = this.currentPage - 1;
236     this.limit -= this.pageSize;
237     this.skip = this.pageSize * this.pageIndex;
238     this.getTableDataGeneral();
239   }
240 }
241
242 public moveToPage(pageNumber: number): void {
243   this.currentPage = pageNumber;
244   this.skip = this.pageSelection[pageNumber - 1].skip;
245   this.limit = this.pageSelection[pageNumber - 1].limit;
246   if (pageNumber > this.currentPage) {
247     this.pageIndex = pageNumber - 1;
248   } else if (pageNumber < this.currentPage) {
249     this.pageIndex = pageNumber + 1;
250   }
251   this.getTableDataGeneral();
252 }
253
254 public PageSize(): void {
255   this.pageSize = [];
256   this.limit = this.pageSize;
257   this.skip = 0;
258   this.currentPage = 1;
259   this.searchDataValue = '';

```

```

admin_clinica > src > app > medical > specialitie > list-specialitie > 18 list-specialitie.component.ts > ...
126 public getMoreData(event: string): void {
127   if (event == 'next') {
128     this.currentPage++;
129     this.pageIndex = this.currentPage - 1;
130     this.limit += this.pageSize;
131     this.skip = this.pageSize * this.pageIndex;
132     this.getTableDataGeneral();
133   } else if (event == 'previous') {
134     this.currentPage--;
135     this.pageIndex = this.currentPage - 1;
136     this.limit -= this.pageSize;
137     this.skip = this.pageSize * this.pageIndex;
138     this.getTableDataGeneral();
139   }
140 }
141
142 public moveToPage(pageNumber: number): void {
143   this.currentPage = pageNumber;
144   this.skip = this.pageSelection[pageNumber - 1].skip;
145   this.limit = this.pageSelection[pageNumber - 1].limit;
146   if (pageNumber > this.currentPage) {
147     this.pageIndex = pageNumber - 1;
148   } else if (pageNumber < this.currentPage) {
149     this.pageIndex = pageNumber + 1;
150   }
151   this.getTableDataGeneral();
152 }
153
154 public PageSize(): void {
155   this.pageSize = [];
156   this.limit = this.pageSize;
157   this.skip = 0;
158   this.currentPage = 1;
159   this.searchDataValue = '';
160 }
161
162 public getTotalPages(totalData: number, pageSize: number): void {
163   this.totalPages = totalData / pageSize;
164   let totalPages = Math.ceil(this.totalPages);
165   let currentPage = 1;
166   let skip = 0;
167   let limit = 0;
168   let pageSelection = [];
169   for (var i = 1; i <= this.totalPages; i++) {
170     let limit = pageSize * i;
171     let skip = (i - 1) * pageSize;
172     let pageSelection.push({ skip: skip, limit: limit });
173     let pageSelection.push({ skip: skip, limit: limit });
174   }
175   this.pageSelection = pageSelection;
176 }
177
178 public getTotalPages(totalData: number, pageSize: number): void {
179   this.totalPages = totalData / pageSize;
180   let totalPages = Math.ceil(this.totalPages);
181   let currentPage = 1;
182   let skip = 0;
183   let limit = 0;
184   let pageSelection = [];
185   for (var i = 1; i <= this.totalPages; i++) {
186     let limit = pageSize * i;
187     let skip = (i - 1) * pageSize;
188     let pageSelection.push({ skip: skip, limit: limit });
189     let pageSelection.push({ skip: skip, limit: limit });
190   }
191   this.pageSelection = pageSelection;

```

Anexo Y

Funcionalidad de Añadir Personal

```

admin_clinica > src > app > medical > staff > add-staff-n > TS add-staff-ncomponents > ...
1 import { Component } from '@angular/core';
2 import { StaffService } from '../service/staff.service';
3
4 @Component({
5   selector: 'app-add-staff-n',
6   templateUrl: './add-staff-n.component.html',
7   styleUrls: ['./add-staff-n.component.scss']
8 })
9 export class AddStaffComponent {
10
11   public selectedValue!: string;
12   public name:string = '';
13   public surname:string = '';
14   public mobile:string = '';
15   public email:string = '';
16   public password:string = '';
17   public password_confirmation:string = '';
18
19   public birth_date:string = '';
20   public gender:number = 1;
21   public education:string = '';
22   public designation:string = '';
23   public address:string = '';
24
25   public roles:any = [];
26
27   public FILE_AVATAR:any;
28   public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';
29
30   public text_success:string = '';
31   public text_validation:string = '';
32   constructor(
33     public staffService: StaffService,
34   ) {
35   }
36
37   ngOnInit(): void {
38     //called after the constructor, initializing input properties, and the first call to ngOnChanges.
39     //Add 'implements OnInit' to the class.
40     this.staffService.listConfig().subscribe((resp:any) => {
41       console.log(resp);
42       this.roles = resp.roles;
43     });
44   }
45
46   save(){
47     this.text_validation = '';
48     if(!this.name || !this.email || !this.surname || !this.FILE_AVATAR || !this.selectedValue){
49       this.text_validation = "LOS CAMPOS SON NECESARIOS (Nombre, Apellido, Email , Avatar, Rol)";
50       return;
51     }
52
53     if(this.password != this.password_confirmation){
54       this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
55       return;
56     }
57     console.log(this.selectedValue);

```

```

admin_clinica > src > app > medical > staff > add-staff-n > TS add-staff-ncomponents > ...
9 export class AddStaffComponent {
10
11   save(){
12
13     let formData = new FormData();
14     formData.append("name",this.name);
15     formData.append("surname",this.surname);
16     formData.append("email",this.email);
17     formData.append("mobile",this.mobile);
18     formData.append("birth_date",this.birth_date);
19     formData.append("gender",this.gender+"");
20     formData.append("education",this.education);
21     formData.append("designation",this.designation);
22     formData.append("address",this.address);
23     formData.append("role_id",this.selectedValue);
24     formData.append("imagen",this.FILE_AVATAR);
25
26     this.staffService.registerUser(formData).subscribe((resp:any) => {
27       console.log(resp);
28
29       if(resp.message == 403){
30         this.text_validation = resp.message_text;
31       }else{
32         this.text_success = 'El usuario ha sido registrado correctamente';
33
34         this.name = '';
35         this.surname = '';
36         this.email = '';
37         this.mobile = '';
38         this.birth_date = '';
39         this.gender = 1;
40         this.education = '';
41         this.designation = '';
42         this.address = '';
43         this.password = '';
44         this.password_confirmation = '';
45         this.selectedValue = '';
46         this.FILE_AVATAR = null;
47         this.IMAGEN_PREVIZUALIZA = null;
48       }
49     });
50   }
51
52   loadfile(event:any){
53     if(event.target.files[0].type.indexOf("image") < 0){
54       // alert("SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN");
55       this.text_validation = "SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN";
56       return;
57     }
58     this.text_validation = '';
59     this.FILE_AVATAR = event.target.files[0];
60     let reader = new FileReader();
61     reader.readAsDataURL(this.FILE_AVATAR);
62     reader.onloadend = () => this.IMAGEN_PREVIZUALIZA = reader.result;
63   }
64 }

```

Anexo Z

Funcionalidad de Editar Personal

```

admin_clinica > src > app > medical > staff > edit-staff-n > % edit-staff-n.component.ts > ...
1 import { Component } from '@angular/core';
2 import { ActivatedRoute } from '@angular/router';
3 import { StaffService } from '../service/staff.service';
4
5 @Component({
6   selector: 'app-edit-staff-n',
7   templateUrl: './edit-staff-n.component.html',
8   styleUrls: ['./edit-staff-n.component.scss']
9 })
10 export class EditStaffComponent {
11
12   public selectedValue!: string;
13   public name:string = '';
14   public surname:string = '';
15   public mobile:string = '';
16   public email:string = '';
17   public password:string = '';
18   public password_confirmation:string = '';
19
20   public birth_date:string = '';
21   public gender:number = 1;
22   public education:string = '';
23   public designation:string = '';
24   public address:string = '';
25
26   public roles:any = [];
27
28   public FILE_AVATAR:any;
29   public IMAGEN_PREVIZUALIZA:any = 'assets/img/user-06.jpg';
30
31   public text_success:string = '';
32   public text_validation:string = '';
33
34   public staff_id:any;
35   public staff_selected:any;
36   constructor(
37     public staffService: StaffService,
38     public activatedRoute: ActivatedRoute
39   ) {
40   }
41
42   ngOnInit(): void {
43     //Called after the constructor, initializing input properties, and the first call to ngOnChanges.
44     //Add 'implements OnInit' to the class.
45     this.activatedRoute.params.subscribe((resp:any) => {
46       console.log(resp);
47       this.staff_id = resp.id;
48     })
49
50     this.staffService.showUser(this.staff_id).subscribe((resp:any) => {
51       console.log(resp);
52       this.staff_selected = resp.user;
53
54       this.selectedValue = this.staff_selected.role.id;
55       this.name = this.staff_selected.name;
56       this.surname = this.staff_selected.surname;
57       this.mobile = this.staff_selected.mobile;
58       this.email = this.staff_selected.email;
59     })

```

```

admin_clinica > src > app > medical > staff > edit-staff-n > % edit-staff-n.component.ts > ...
10 export class EditStaffComponent {
11   ngOnInit(): void {
12     this.staffService.showUser(this.staff_id).subscribe((resp:any) => {
13       console.log(resp);
14       this.staff_selected = resp.user;
15       this.name = this.staff_selected.name;
16       this.surname = this.staff_selected.surname;
17       this.mobile = this.staff_selected.mobile;
18       this.email = this.staff_selected.email;
19       this.birth_date = new Date(this.staff_selected.birth_date).toISOString();
20       this.gender = this.staff_selected.gender;
21       this.education = this.staff_selected.education;
22       this.designation = this.staff_selected.designation;
23       this.address = this.staff_selected.address;
24       this.IMAGEN_PREVIZUALIZA = this.staff_selected.avatar;
25     })
26
27     this.staffService.listConfig().subscribe((resp:any) => {
28       console.log(resp);
29       this.roles = resp.roles;
30     })
31   }
32
33   save(){
34     this.text_validation = '';
35     if(!this.name || !this.email || !this.surname){
36       this.text_validation = "LOS CAMPOS SON NECESARIOS (name,surname,email)";
37       return;
38     }
39     if(this.password){
40       if(this.password != this.password_confirmation){
41         this.text_validation = "LAS CONTRASENAS DEBEN SER IGUALES";
42         return;
43       }
44     }
45     console.log(this.selectedValue);
46
47     let formData = new FormData();
48     formData.append("name",this.name);
49     formData.append("surname",this.surname);
50     formData.append("email",this.email);
51     formData.append("mobile",this.mobile);
52     formData.append("birth_date",this.birth_date);
53     formData.append("gender",this.gender+"");
54     if(this.education){
55       formData.append("education",this.education);
56     }
57     if(this.designation){
58       formData.append("designation",this.designation);
59     }
60     if(this.address){
61       formData.append("address",this.address);
62     }
63     if(this.password){
64       formData.append("password",this.password);
65     }
66     formData.append("role_id",this.selectedValue);
67     if(this.FILE_AVATAR){
68       formData.append("imagen",this.FILE_AVATAR);
69     }

```

```

admin_clinica > src > app > medical > staff > edit-staff-n > ? edit-staff-n.component.ts > ...
10 export class EditStaffComponent {
73   save(){
81     this.text_validation = "LAS CONTRASEÑA DEBEN SER IGUALES";
82     return;
83   }
84 }
85 console.log(this.selectedValue);
86
87 let formData = new FormData();
88 formData.append("name",this.name);
89 formData.append("surname",this.surname);
90 formData.append("email",this.email);
91 formData.append("mobile",this.mobile);
92 formData.append("birth_date",this.birth_date);
93 formData.append("gender",this.genders);
94 if(this.education){
95   formData.append("education",this.education);
96 }
97 if(this.designation){
98   formData.append("designation",this.designation);
99 }
100 if(this.address){
101   formData.append("address",this.address);
102 }
103 if(this.password){
104   formData.append("password",this.password);
105 }
106 formData.append("role_id",this.selectedValue);
107 if(this.FILE_AVATAR){
108   formData.append("imagen",this.FILE_AVATAR);
109 }
110
111 this.staffService.updateUser(this.staff_id,formData).subscribe((resp:any) => {
112   console.log(resp);
113
114   if(resp.message == 403){
115     this.text_validation = resp.message_text;
116   }else{
117     this.text_success = 'El usuario ha editado correctamente';
118   }
119
120 })
121 }
122
123 loadFile($event:any){
124   if($event.target.files[0].type.indexOf('image') < 0){
125     // alert("SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN");
126     this.text_validation = "SOLAMENTE PUEDEN SER ARCHIVOS DE TIPO IMAGEN";
127     return;
128   }
129   this.text_validation = '';
130   this.FILE_AVATAR = $event.target.files[0];
131   let reader = new FileReader();
132   reader.readAsDataURL(this.FILE_AVATAR);
133   reader.onloadend = () => this.IMAGEN_PREVIZUALIZA = reader.result;
134 }
135
136 }

```

Anexo AA

Funcionalidad de Listado de Personal

```

admin_clinica > src > app > medical > staff > list-staff-n > ? list-staff-n.component.ts > ...
1 import { Component } from '@angular/core';
2 import { StaffService } from '../service/staff.service';
3 import { MatTableDataSource } from '@angular/material/table';
4
5 @Component({
6   selector: 'app-list-staff-n',
7   templateUrl: './list-staff-n.component.html',
8   styleUrls: ['./list-staff-n.component.scss']
9 })
10 export class ListStaffComponent {
11
12   public userslist:any = [];
13   dataSource: MatTableDataSource<any>;
14
15   public showFilter = false;
16   public searchDataValue = '';
17   public lastIndex = 0;
18   public pageSize = 10;
19   public totalData = 0;
20   public skip = 0;//MIN
21   public limit: number = this.pageSize;//MAX
22   public pageIndex = 0;
23   public serialNumberArray: Array<number> = [];
24   public currentPage = 1;
25   public pageNumberArray: Array<number> = [];
26   public pageSelection: Array<any> = [];
27   public totalPages = 0;
28
29   public role_generals:any = [];
30   public staff_selected:any;
31   public user:any;
32   constructor(
33     public staffService: StaffService,
34   ){
35   }
36
37   ngOnInit() {
38     this.getTableData();
39     this.user = this.staffService.authService.user;
40   }
41   private getTableData(): void {
42     this.userslist = [];
43     this.serialNumberArray = [];
44
45     this.staffService.listUsers().subscribe((resp:any) => {
46
47       console.log(resp);
48
49       this.totalData = resp.users.data.length;
50       this.role_generals = resp.users.data;
51       this.getTableDataGeneral();
52     })
53   }

```

```

admin_clinica > src > app > medical > staff > list-staff-n > 18 list-staff-n.component.ts > ...
10 export class ListStaffNComponent {
11   private getTableData(): void {
12     this.staffService.listUsers().subscribe((resp:any) => {
13       this.role_generals = resp.users.data;
14       this.getTableDataGeneral();
15     });
16   }
17   isPermission(permission:string){
18     if(this.user.roles.includes('Super-Admin')){
19       return true;
20     }
21     if(this.user.permissions.includes(permission)){
22       return true;
23     }
24     return false;
25   }
26   getTableDataGeneral() {
27     this.userlist = [];
28     this.serialNumberArray = [];
29     this.role_generals.map((res: any, index: number) => {
30       const serialNumber = index + 1;
31       if (index >= this.skip && serialNumber <= this.limit) {
32         this.userlist.push(res);
33         this.serialNumberArray.push(serialNumber);
34       }
35     });
36     this.dataSource = new MatTableDataSource<any>(this.userlist);
37     this.calculateTotalPages(this.totalData, this.pageSize);
38   }
39   selectUser(rol:any){
40     this.staff_selected = rol;
41   }
42   deleteUser(){
43     this.staffService.deleteUser(this.staff_selected.id).subscribe((resp:any) => {
44       console.log(resp);
45       let INDEX = this.userlist.findIndex((item:any) => item.id == this.staff_selected.id);
46       if(INDEX != -1){
47         this.userlist.splice(INDEX,1);
48         $('#delete_patient').hide();
49         $('#delete_patient').removeClass("show");
50         $('#modal-backdrop').remove();
51         $('#body').removeClass();
52         $('#body').removeAttr("style");
53       }
54       this.staff_selected = null;
55     });
56   }
57   // eslint-disable-next-line @typescript-eslint/no-explicit-any
58   public searchData(value: any): void {

```

```

admin_clinica > src > app > medical > staff > list-staff-n > 18 list-staff-n.component.ts > ...
10 export class ListStaffNComponent {
11   private getTableData(): void {
12     this.staffService.listUsers().subscribe((resp:any) => {
13       this.role_generals = resp.users.data;
14       this.getTableDataGeneral();
15     });
16   }
17   isPermission(permission:string){
18     if(this.user.roles.includes('Super-Admin')){
19       return true;
20     }
21     if(this.user.permissions.includes(permission)){
22       return true;
23     }
24     return false;
25   }
26   getTableDataGeneral() {
27     this.userlist = [];
28     this.serialNumberArray = [];
29     this.role_generals.map((res: any, index: number) => {
30       const serialNumber = index + 1;
31       if (index >= this.skip && serialNumber <= this.limit) {
32         this.userlist.push(res);
33         this.serialNumberArray.push(serialNumber);
34       }
35     });
36     this.dataSource = new MatTableDataSource<any>(this.userlist);
37     this.calculateTotalPages(this.totalData, this.pageSize);
38   }
39   selectUser(rol:any){
40     this.staff_selected = rol;
41   }
42   deleteUser(){
43     this.staffService.deleteUser(this.staff_selected.id).subscribe((resp:any) => {
44       console.log(resp);
45       let INDEX = this.userlist.findIndex((item:any) => item.id == this.staff_selected.id);
46       if(INDEX != -1){
47         this.userlist.splice(INDEX,1);
48         $('#delete_patient').hide();
49         $('#delete_patient').removeClass("show");
50         $('#modal-backdrop').remove();
51         $('#body').removeClass();
52         $('#body').removeAttr("style");
53       }
54       this.staff_selected = null;
55     });
56   }
57   // eslint-disable-next-line @typescript-eslint/no-explicit-any
58   public searchData(value: any): void {
59     this.dataSource.filter = value.trim().toLowerCase();
60     this.userlist = this.dataSource.filteredData;
61   }
62   public sortData(sort: any) {
63     const data = this.userlist.slice();
64     if (!sort.active || sort.direction === '') {
65       this.userlist = data;
66     } else {
67       this.userlist = data.sort((a: any, b: any) => {
68         // eslint-disable-next-line @typescript-eslint/no-explicit-any
69         const aValue = (a as any)[sort.active];
70         // eslint-disable-next-line @typescript-eslint/no-explicit-any
71         const bValue = (b as any)[sort.active];
72         return (aValue < bValue ? -1 : 1) * (sort.direction === 'asc' ? 1 : -1);
73       });
74     }
75   }
76   public getMoreData(event: string): void {
77     if (event == 'next') {
78       this.currentPage++;
79       this.pageIndex = this.currentPage - 1;
80       this.limit = this.pageSize;
81       this.skip = this.pageSize * this.pageIndex;
82       this.getTableDataGeneral();
83     } else if (event == 'previous') {
84       this.currentPage--;
85       this.pageIndex = this.currentPage - 1;
86       this.limit = this.pageSize;
87       this.skip = this.pageSize * this.pageIndex;
88       this.getTableDataGeneral();
89     }
90   }
91   public moveToPage(pageNumber: number): void {
92     this.currentPage = pageNumber;
93     this.skip = this.pageSelection[pageNumber - 1].skip;
94     this.limit = this.pageSelection[pageNumber - 1].limit;
95     if (pageNumber > this.currentPage) {
96       this.pageIndex = pageNumber - 1;
97     } else if (pageNumber < this.currentPage) {
98       this.pageIndex = pageNumber + 1;
99     }
100     this.getTableDataGeneral();
101   }
102   public PageSize(): void {
103     this.pageSelection = [];
104     // ...

```

```

admin_clinica > src > app > medical > staff > list-staff-n > list-staff-n.component.ts >
10 export class ListStaffComponent {
11   public getMoreData(event: string): void {
12     this.currentPage++;
13     this.pageIndex = this.currentPage - 1;
14     this.limit += this.pageSize;
15     this.skip = this.pageSize * this.pageIndex;
16     this.getTableDataGeneral();
17   } else if (event == 'previous') {
18     this.currentPage--;
19     this.pageIndex = this.currentPage - 1;
20     this.limit -= this.pageSize;
21     this.skip = this.pageSize * this.pageIndex;
22     this.getTableDataGeneral();
23   }
24 }
25
26 public moveOfPage(pageNumber: number): void {
27   this.currentPage = pageNumber;
28   this.skip = this.pageSelection[pageNumber - 1].skip;
29   this.limit = this.pageSelection[pageNumber - 1].limit;
30   if (pageNumber > this.currentPage) {
31     this.pageIndex = pageNumber - 1;
32   } else if (pageNumber < this.currentPage) {
33     this.pageIndex = pageNumber + 1;
34   }
35   this.getTableDataGeneral();
36 }
37
38 public PageSize(): void {
39   this.pageSelection = [];
40   this.limit = this.pageSize;
41   this.skip = 0;
42   this.currentPage = 1;
43   this.searchDataValue = '';
44   this.getTableDataGeneral();
45 }
46
47 private calculateTotalPages(totalData: number, pageSize: number): void {
48   this.pageNumberArray = [];
49   this.totalPages = totalData / pageSize;
50   if (this.totalPages % 1 != 0) {
51     this.totalPages = Math.trunc(this.totalPages + 1);
52   }
53   /* eslint no-var: off */
54   for (var i = 1; i <= this.totalPages; i++) {
55     const limit = pageSize * i;
56     const skip = limit - pageSize;
57     this.pageNumberArray.push(i);
58     this.pageSelection.push({ skip: skip, limit: limit });
59   }
60   // 1
61   // 0 - 10
62   // 2
63   // 10 - 20
64 }
65 }

```

Anexo BB

Gestión de Autenticación con Angular: auth.service.ts

```

admin_clinica > src > app > shared > auth > 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   // Como esto vamos a poder realizar peticiones HTTP, hacia los endpoint que definamos dentro de nuestro
17   constructor(
18     private router: Router,
19     public http: HttpClient,
20   ) {
21     this.getLocalStorage();
22   }
23
24   //Hacemos esta función para poder llamar los usuarios 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 la 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 almacenar 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 }

```

```

56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75

```

```

savelocalStorage(auth:any){
  if(auth && auth.access_token){
    localStorage.setItem("token",auth.access_token);
    localStorage.setItem("user",JSON.stringify(auth.user));
    //Aqui 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

```

```

1  import { Injectable } from '@angular/core';
2  import { routes } from '../routes/routes';
3  import { map, Observable } from 'rxjs';
4  import { HttpClient } from '@angular/common/http';
5  import { apiResultFormat } from '../models/models';
6
7
8  @Injectable({
9    providedIn: 'root',
10 })
11 export class DataService {
12   constructor(private http: HttpClient) {}
13
14   public getDoctorslist(): Observable<apiResultFormat> {
15     return this.http.get<apiResultFormat>('assets/json/doctors-list.json').pipe(
16       map((res: apiResultFormat) => {
17         return res;
18       })
19     );
20   }
21   public getPatientslist(): Observable<apiResultFormat> {
22     return this.http.get<apiResultFormat>('assets/json/doctors-list.json').pipe(
23       map((res: apiResultFormat) => {
24         return res;
25       })
26     );
27   }
28   public getStafflist(): Observable<apiResultFormat> {
29     return this.http.get<apiResultFormat>('assets/json/staff-list.json').pipe(
30       map((res: apiResultFormat) => {
31         return res;
32       })
33     );
34   }
35   public getAppointmentlist(): Observable<apiResultFormat> {
36     return this.http.get<apiResultFormat>('assets/json/appointment-list.json').pipe(
37       map((res: apiResultFormat) => {
38         return res;
39       })
40     );
41   }
42   public getStaffHoliday(): Observable<apiResultFormat> {
43     return this.http.get<apiResultFormat>('assets/json/staff-holiday.json').pipe(
44       map((res: apiResultFormat) => {
45         return res;
46       })
47     );
48   }
49   public getSchedule(): Observable<apiResultFormat> {
50     return this.http.get<apiResultFormat>('assets/json/schedule.json').pipe(
51       map((res: apiResultFormat) => {
52         return res;
53       })
54     );
55   }
56 }

```

```

admin@clinica > src > app > shared > data > TS dataservices > DataService > sidebar > menu
11 export class DataService {
12   public getSchedule(): Observable<ApiResponseFormat> {
13     //
14   }
15 }
16 public getInvoices(): Observable<ApiResponseFormat> {
17   return this.http.get<ApiResponseFormat>('assets/json/invoices.json').pipe(
18     map((res: ApiResponseFormat) => {
19       return res;
20     })
21   );
22 }
23 public getPayments(): Observable<ApiResponseFormat> {
24   return this.http.get<ApiResponseFormat>('assets/json/payments.json').pipe(
25     map((res: ApiResponseFormat) => {
26       return res;
27     })
28   );
29 }
30 public getExpenses(): Observable<ApiResponseFormat> {
31   return this.http.get<ApiResponseFormat>('assets/json/expenses.json').pipe(
32     map((res: ApiResponseFormat) => {
33       return res;
34     })
35   );
36 }
37 public getTaxes(): Observable<ApiResponseFormat> {
38   return this.http.get<ApiResponseFormat>('assets/json/taxes.json').pipe(
39     map((res: ApiResponseFormat) => {
40       return res;
41     })
42   );
43 }
44 public getProvidentFund(): Observable<ApiResponseFormat> {
45   return this.http.get<ApiResponseFormat>('assets/json/provident-fund.json').pipe(
46     map((res: ApiResponseFormat) => {
47       return res;
48     })
49   );
50 }
51 public getDepartmentist(): Observable<ApiResponseFormat> {
52   return this.http.get<ApiResponseFormat>('assets/json/department-list.json').pipe(
53     map((res: ApiResponseFormat) => {
54       return res;
55     })
56   );
57 }
58 public getSalary(): Observable<ApiResponseFormat> {
59   return this.http.get<ApiResponseFormat>('assets/json/salary.json').pipe(
60     map((res: ApiResponseFormat) => {
61       return res;
62     })
63   );
64 }
65 public getAssetslist(): Observable<ApiResponseFormat> {
66   return this.http.get<ApiResponseFormat>('assets/json/assets-list.json').pipe(
67     map((res: ApiResponseFormat) => {
68       return res;
69     })
70   );
71 }

```

```

admin@clinica > src > app > shared > data > TS dataservices > DataService > sidebar > menu
11 export class DataService {
98   public getSalary(): Observable<ApiResponseFormat> {
104   }
105 }
106 public getAssetslist(): Observable<ApiResponseFormat> {
107   return this.http.get<ApiResponseFormat>('assets/json/assets-list.json').pipe(
108     map((res: ApiResponseFormat) => {
109       return res;
110     })
111   );
112 }
113 public getExpenseReports(): Observable<ApiResponseFormat> {
114   return this.http.get<ApiResponseFormat>('assets/json/expense-reports.json').pipe(
115     map((res: ApiResponseFormat) => {
116       return res;
117     })
118   );
119 }
120 public getInvoiceReports(): Observable<ApiResponseFormat> {
121   return this.http.get<ApiResponseFormat>('assets/json/invoice-reports.json').pipe(
122     map((res: ApiResponseFormat) => {
123       return res;
124     })
125   );
126 }
127 public getAllInvoice(): Observable<ApiResponseFormat> {
128   return this.http.get<ApiResponseFormat>('assets/json/all-invoice.json').pipe(
129     map((res: ApiResponseFormat) => {
130       return res;
131     })
132   );
133 }
134 public getPatientDashboard(): Observable<ApiResponseFormat> {
135   return this.http.get<ApiResponseFormat>('assets/json/patient-dashboard.json').pipe(
136     map((res: ApiResponseFormat) => {
137       return res;
138     })
139   );
140 }
141 public getInvoicesPaid(): Observable<ApiResponseFormat> {
142   return this.http.get<ApiResponseFormat>('assets/json/invoices-paid.json').pipe(
143     map((res: ApiResponseFormat) => {
144       return res;
145     })
146   );
147 }
148 public getInvoicesOverdue(): Observable<ApiResponseFormat> {
149   return this.http.get<ApiResponseFormat>('assets/json/invoices-overdue.json').pipe(
150     map((res: ApiResponseFormat) => {
151       return res;
152     })
153   );
154 }
155 public getInvoicesDraft(): Observable<ApiResponseFormat> {
156   return this.http.get<ApiResponseFormat>('assets/json/invoices-draft.json').pipe(
157     map((res: ApiResponseFormat) => {
158       return res;
159     })
160   );

```



```

admin_clinica > src > app > shared > data > data.services > DataService > sidebar > menu
11 export class DataService {
196   public sidebar = [
201     menu: [
320       subMenus: [
339         permission: 'edit_specialty',
340         show_nav: false,
341       ],
342       {
343         menuValue: 'Eliminar Especialidades',
344         route: '',
345         base: '',
346         permission: 'delete_specialty',
347         show_nav: false,
348       },
349     ],
350   },
351   {
352     menuValue: 'Doctores',
353     hasSubRoute: true,
354     showSubRoute: false,
355     base: 'doctor',
356     img: 'assets/img/icons/menu-icon-02.svg',
357     subMenus: [
358       {
359         menuValue: 'Listado de Doctores',
360         route: routes.doctorslist,
361         base: routes.doctorslist,
362         permission: 'list_doctor',
363         show_nav: true,
364       },
365       {
366         menuValue: 'Añadir Doctores',
367         route: routes.addDoctor,
368         base: routes.addDoctor,
369         permission: 'register_doctor',
370         show_nav: true,
371       },
372       {
373         menuValue: 'editar Doctores',
374         route: '',
375         base: '',
376         permission: 'edit_doctor',
377         show_nav: false,
378       },
379     ],
380     {
381       menuValue: 'Eliminar Doctores',
382       route: '',
383       base: '',
384       permission: 'delete_doctor',
385       show_nav: false,
386     },
387     {
388       menuValue: 'Perfil del Doctores',
389       route: '',
390       base: '',
391       permission: 'profile_doctor',
392       show_nav: false,
393     }
394   ]
395 }

```

```

admin_clinica > src > app > shared > data > data.services > DataService > sidebar > menu
11 export class DataService {
196   public sidebar = [
201     menu: [
357     subMenus: [
385       show_nav: false,
386     ],
387     {
388       menuValue: 'Perfil del Doctores',
389       route: '',
390       base: '',
391       permission: 'profile_doctor',
392       show_nav: false,
393     },
394   ],
395 },
396 {
397   menuValue: 'Pacientes',
398   hasSubRoute: true,
399   showSubRoute: false,
400   base: 'patient',
401   img: 'assets/img/icons/menu-icon-03.svg',
402   subMenus: [
403     {
404       menuValue: 'Listado de Pacientes',
405       route: routes.patientslist,
406       base: routes.patientslist,
407       permission: 'list_patient',
408       show_nav: true,
409     },
410     {
411       menuValue: 'Añadir Pacientes',
412       route: routes.addPatient,
413       base: routes.addPatient,
414       permission: 'register_patient',
415       show_nav: true,
416     },
417     {
418       menuValue: 'editar Pacientes',
419       route: '',
420       base: '',
421       permission: 'edit_patient',
422       show_nav: false,
423     },
424     {
425       menuValue: 'Eliminar Pacientes',
426       route: '',
427       base: '',
428       permission: 'delete_patient',
429       show_nav: false,
430     },
431     {
432       menuValue: 'Perfil del Paciente',
433       route: '',
434       base: '',
435       permission: 'profile_patient',
436       show_nav: false,
437     }
438   ]
439 }

```

```

admin_clinica > src > app > shared > data > 78 dataservice > DataService > sidebar > menu
11 export class DataService {
196   public sidebar = [
201     menu: [
402       subMenus: [
431         {
432           menuValue: 'Perfil del Paciente',
433           route: '',
434           base: '',
435           permission: 'profile_patient',
436           show_nav: false,
437         },
438       ],
439     },
440     {
441       menuValue: 'Citas',
442       hasSubRoute: true,
443       showSubRoute: false,
444       base: 'appointments',
445       img: 'assets/img/icons/menu-icon-04.svg',
446       subMenus: [
447         {
448           menuValue: 'Listado de Citas',
449           route: routes.appointmentlist,
450           base: routes.appointmentlist,
451           permission: 'list_appointment',
452           show_nav: true,
453         },
454         {
455           menuValue: 'Aadir Citas',
456           route: routes.addAppointment,
457           base: routes.addAppointment,
458           permission: 'register_appointment',
459           show_nav: true,
460         },
461         {
462           menuValue: 'Editar Citas',
463           route: '',
464           base: '',
465           permission: 'edit_appointment',
466           show_nav: false,
467         },
468         {
469           menuValue: 'Eliminar Citas',
470           route: '',
471           base: '',
472           permission: 'delete_appointment',
473           show_nav: false,
474         },
475         {
476           menuValue: 'Atenci3n de Citas',
477           route: '',
478           base: '',
479           permission: 'attention_appointment',
480           show_nav: false,
481         },
482       ],
483     },
484   ];

```

```

// },
{
  menuValue: 'Calendario',
  route: routes.calendar,
  hasSubRoute: false,
  showSubRoute: false,
  icon: 'fa-calendar',
  faIcon: true,
  base: 'calendar',
  permission: 'calendar',
  show_nav: true,
  subMenus: [],
},
];
// public sidebarList = [

```

Anexo DD

Punto de Entrada para API en Laravel: api.php

```

1  api_clinica > routes > api.php
2  </php>
3  use Illuminate\Http\Request;
4  use Illuminate\Support\Facades\Route;
5  use App\Http\Controllers\AuthController;
6  use App\Http\Controllers\Admin\Rol\RolesController;
7  use App\Http\Controllers\Patient\PatientController;
8  use App\Http\Controllers\Admin\Staff\StaffsController;
9  use App\Http\Controllers\Admin\Doctor\DoctorsController;
10 use App\Http\Controllers\Dashboard\DashboardApiController;
11 use App\Http\Controllers\Admin\Doctor\SpecialityController;
12 use App\Http\Controllers\Appointment\AppointmentController;
13 use App\Http\Controllers\Appointment\AppointmentPayController;
14 use App\Http\Controllers\Appointment\AppointmentAttentionController;
15
16 /*
17 |-----|
18 | API Routes
19 |-----|
20 |
21 | Here is where you can register API routes for your application. These
22 | routes are loaded by the RouteServiceProvider and all of them will
23 | be assigned to the "api" middleware group. Make something great!
24 |
25 |*/
26
27 Route::middleware("auth:sanctum")->get('/user', function (Request $request) {
28     return $request->user();
29 });
30
31 Route::group([
32     // se comenta la siguiente línea de código ,porque estas rutas no es necesario mandar un token, ya que tenemos
33     // el login y tenemos el registro que por lo general son accesos publicos
34     // El unico publico es el login, ya que las demás rutas deben ser autenticadas
35     // 'middleware' => 'auth:api',
36     'prefix' => 'auth',
37     // 'middleware' => ['role:admin','permission:publish articles'],
38 ], function ($router) {
39     Route::post('/register', [AuthController::class, 'register'])->name('register');
40     Route::post('/login', [AuthController::class, 'login'])->name('login');
41     Route::post('/logout', [AuthController::class, 'logout'])->name('logout');
42     Route::post('/refresh', [AuthController::class, 'refresh'])->name('refresh');
43     Route::post('/me', [AuthController::class, 'me'])->name('me');
44     Route::post('/list', [AuthController::class, 'list']);
45     Route::post('/reg', [AuthController::class, 'reg']);
46 });
47
48 Route::group([
49     'middleware' => 'auth:api',
50 ], function ($router) {
51     Route::resource("roles",RolesController::class);
52
53     Route::get("staffs/config",[StaffsController::class,"config"]);
54     Route::post("staffs/{id}",[StaffsController::class,"update"]);
55     Route::resource("staffs",StaffsController::class);
56     //
57     Route::resource("specialities",SpecialityController::class);
58

```

```

api_clinica > routes > api.php
1  </php>
2  Route::group([
3      // se comenta la siguiente línea de código ,porque estas rutas no es necesario mandar un token, ya que tenemos
4      // 'middleware' => 'auth:api',
5      'prefix' => 'auth',
6      // 'middleware' => ['role:admin','permission:publish articles'],
7  ], function ($router) {
8      Route::post('/register', [AuthController::class, 'register'])->name('register');
9      Route::post('/login', [AuthController::class, 'login'])->name('login');
10     Route::post('/logout', [AuthController::class, 'logout'])->name('logout');
11     Route::post('/refresh', [AuthController::class, 'refresh'])->name('refresh');
12     Route::post('/me', [AuthController::class, 'me'])->name('me');
13     Route::post('/list', [AuthController::class, 'list']);
14     Route::post('/reg', [AuthController::class, 'reg']);
15 });
16
17 Route::group([
18     'middleware' => 'auth:api',
19 ], function ($router) {
20     Route::resource("roles",RolesController::class);
21
22     Route::get("staffs/config",[StaffsController::class,"config"]);
23     Route::post("staffs/{id}",[StaffsController::class,"update"]);
24     Route::resource("staffs",StaffsController::class);
25     //
26     Route::resource("specialities",SpecialityController::class);
27     //
28     Route::get("doctors/profile/{id}",[DoctorsController::class,"profile"]);
29     Route::get("doctors/config",[DoctorsController::class,"config"]);
30     Route::post("doctors/{id}",[DoctorsController::class,"update"]);
31     Route::resource("doctors",DoctorsController::class);
32     //
33     Route::get("patients/profile/{id}",[PatientController::class,"profile"]);
34     Route::post("patients/{id}",[PatientController::class,"update"]);
35     Route::resource("patients",PatientController::class);
36     //
37     Route::get("appointment/config",[AppointmentController::class,"config"]);
38     Route::get("appointment/patient",[AppointmentController::class,"query_patient"]);
39     Route::post("appointment/filter",[AppointmentController::class,"filter"]);
40     Route::post("appointment/calendar",[AppointmentController::class,"calendar"]);
41     Route::resource("appointment",AppointmentController::class);
42
43     Route::resource("appointment-pay",AppointmentPayController::class);
44     Route::resource("appointment-attention",AppointmentAttentionController::class);
45
46     Route::post("dashboard/admin",[DashboardApiController::class,"dashboard_admin"]);
47     Route::post("dashboard/admin-year",[DashboardApiController::class,"dashboard_admin_year"]);
48
49     Route::post("dashboard/doctor",[DashboardApiController::class,"dashboard_doctor"]);
50     Route::get("dashboard/config",[DashboardApiController::class,"config"]);
51     Route::post("dashboard/doctor-year",[DashboardApiController::class,"dashboard_doctor_year"]);
52 });

```

Anexo EE

Controlador de Autenticación en Laravel: AuthController.php

```

api_clinica > app > Http > Controllers > AuthController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Support\Facades\Auth;
6  use App\Http\Controllers\Controller;
7  use App\Models\User;
8  use Validator;
9
10
11 8 references|0 implementations
12 class AuthController extends Controller
13 {
14     /**
15      * Create a new AuthController instance.
16      *
17      * @return void
18      */
19     0 references|0 overloads
20     public function __construct()
21     {
22         $this->middleware('auth:api', ['except' => ['login', 'register']]);
23     }
24
25     /**
26      * Register a User.
27      *
28      * @return \Illuminate\Http\JsonResponse
29      */
30     1 reference|0 overloads
31     public function register() {
32
33         $validator = Validator::make(request()->all(), [
34             'name' => 'required',
35             'email' => 'required|email|unique:users',
36             'password' => 'required|min:8',
37         ]);
38
39         if ($validator->fails()) {
40             return response()->json($validator->errors()->toJson(), 400);
41         }
42
43         $user = new User;
44         $user->name = request()->name;
45         $user->email = request()->email;
46         $user->password = bcrypt(request()->password);
47         $user->save();
48
49         return response()->json($user, 201);
50     }
51
52     1 reference|0 overloads
53     public function reg() {
54         $this->authorize('create', User::class);
55
56         $validator = Validator::make(request()->all(), [
57             'name' => 'required',
58             'email' => 'required|email|unique:users',
59

```

```

api_clinica > app > Http > Controllers > AuthController.php > ...
13 class AuthController extends Controller
14 {
15     1 reference|0 overloads
16     public function reg() {
17         $this->authorize('create', User::class);
18
19         $validator = Validator::make(request()->all(), [
20             'name' => 'required',
21             'email' => 'required|email|unique:users',
22             'password' => 'required|min:8',
23         ]);
24
25         if ($validator->fails()) {
26             return response()->json($validator->errors()->toJson(), 400);
27         }
28
29         $user = new User;
30         $user->name = request()->name;
31         $user->email = request()->email;
32         $user->password = bcrypt(request()->password);
33         $user->save();
34
35         return response()->json($user, 201);
36     }
37
38     /**
39      * Get a JWT via given credentials.
40      *
41      * @return \Illuminate\Http\JsonResponse
42      */
43     1 reference|0 overloads
44     public function login()
45     {
46         $credentials = request(['email', 'password']);
47
48         if (! $token = auth('api')->attempt($credentials)) {
49             return response()->json(['error' => 'Unauthorized'], 401);
50         }
51
52         return $this->respondWithToken($token);
53     }
54
55     /**
56      * Get the authenticated User.
57      *
58      * @return \Illuminate\Http\JsonResponse
59      */
60     1 reference|0 overloads
61     public function me()
62     {
63         return response()->json(auth('api')->user());
64     }
65
66     1 reference|0 overloads
67     function list() {
68         $users = User::all();
69         return response()->json([
70             'users' => $users,
71

```

```

api_clinica > app > Http > Controllers > AuthController.php > ...
11 class AuthController extends Controller
12 {
13     /**
14      * reference | @overrides
15      */
16     public function list() {
17         $users = User::all();
18         return response()->json([
19             "users" => $users,
20         ]);
21     }
22     /**
23      * Log the user out (Invalidate the token).
24      *
25      * @return \Illuminate\Http\JsonResponse
26      */
27     public function logout()
28     {
29         auth()->logout();
30
31         return response()->json(['message' => 'Successfully logged out!']);
32     }
33     /**
34      * Refresh a token.
35      *
36      * @return \Illuminate\Http\JsonResponse
37      */
38     public function refresh()
39     {
40         return $this->respondWithToken(auth()->refresh());
41     }
42     /**
43      * Get the token array structure.
44      *
45      * @param string $token
46      *
47      * @return \Illuminate\Http\JsonResponse
48      */
49     protected function respondWithToken($token)
50     {
51         $permissions = auth('api')->user()->getAllPermissions()->map(function($perm) {
52             return $perm->name;
53         });
54         return response()->json([
55             'access_token' => $token,
56             'token_type' => 'bearer',
57             'expires_in' => auth('api')->factory()->getTTL() * 60,
58             "user" => [
59                 "name" => auth('api')->user()->name,
60                 "surname" => auth('api')->user()->surname,
61                 // "avatar" => auth('api')->user()->avatar,
62                 "email" => auth('api')->user()->email,
63                 "roles" => auth('api')->user()->getRoleNames(),
64                 "permissions" => $permissions,
65             ],
66         ]);
67     }
68 }

```

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>.