TRABAJO DE FIN DE CICLO

DESARROLLO DE UNA APLICACIÓN WEB DE GESTIÓN MÉDICA

AUTORES: Rafael Romero Roibu y Alberto Martínez Pérez

TUTOR: Elkin Guadilla González

FECHA DE ENTREGA: 12 de junio de 2024

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

RESUMEN

ABSTRACT

INTRODUCCIÓN

En la actualidad, la atención sanitaria, así como la gestión de su información, el permitir a los pacientes no sólo llevar a cabo un seguimiento de su historial clínico, sino que sean participantes activos del mismo, es de alta importancia en un mundo digitalizado como el que tenemos hoy en día (por ejemplo, aportando información como mediciones regulares de tensión arterial o de glucosa).

Por desgracia, no existe ninguna aplicación de referencia en cuanto a nivel de gestión clínica debido a que la mayoría de las que existen cuentan con interfaces poco intuitivas y accesibles, además de ser ineficientes, ya que sólo permiten la entrada de datos por parte del personal médico y no de los pacientes.

A la vista de esto y considerando el desarrollo web como una de las ramas más importantes dentro del sector de la tecnología, hemos decidido desarrollar una aplicación web de gestión clínica que sea usable y accesible tanto para pacientes como profesionales, y que permita además llevar una gestión del centro clínico en tema de modificación de especialidades o de personal. Esperamos que con ella podamos demostrar los conocimientos que hemos ido adquiriendo a lo largo del ciclo formativo.

Por supuesto, somos conscientes de los desafíos que acompañan al desarrollo de una aplicación de este estilo, como es mantener la seguridad y confidencialidad de los datos médicos de los usuarios y profesionales que empleen la aplicación.

OBJETIVOS

El objetivo final de cara a la presentación proyecto será crear una aplicación web que permita a un paciente solicitar cita con su especialista, así como poder visualizar los informes resultantes de las citas. No obstante, el objetivo final de la aplicación avanza más allá de la presentación al estar previsto un desarrollo posterior que aporte a la aplicación nuevas funcionalidades tales como chat, registro de alergias conocidas, posibilidad de realizar consultas por videoconferencia, etc.

A continuación, se enumerarán los objetivos previstos de cara al desarrollo de este proyecto empezando por los objetivos más básicos relativos a la planificación y diseño de la aplicación web de gestión clínica y escalando hasta llegar a los puntos de programación y desarrollo de esta.

1. Desarrollar una aplicación web funcional e intuitiva.
2. Aprender la sintaxis y funcionamiento de Angular y Express.js.
3. Asegurar el correcto funcionamiento en los 3 estándares de pantalla actuales: monitor, Tablet y móvil.
4. Definir un proceso de registro de usuarios para que se puedan dar de alta en la clínica.
5. Permitir a un usuario administrador el poder gestionar a los usuarios y la información visible en las secciones de “Especialidades” y de “Especialistas”.
6. Definir un proceso de inicio de sesión que permita realizar diferentes funciones en relación con el rol del usuario.
7. Realizar un sistema que permita a un paciente poder subir sus mediciones de tensión arterial y glucosa permitiendo de esta manera al especialista tener un seguimiento sobre ello.
8. Realizar un sistema que permite a un paciente poder visualizar la medicación que tiene asignada, así como al especialista la posibilidad de editar, añadir o eliminar medicación a este listado.
9. Permitir a un especialista poder añadir nuevos medicamentos a la base de datos.

ALCANCE DEL PROYECTO

1. REQUISITOS FUNCIONALES Y NO FUNCIONALES

Funcionales:

1. Diseñar un formulario de registro, guardando los datos de forma segura.
2. Implementar un sistema de autenticación de usuarios basándose en el usuario y contraseña indicados en el formulario de registro.
3. Mostrar en la sección “Especialidades” todas las especialidades disponibles en la clínica.
4. Mostrar en la sección “Especialistas” todos los especialistas de la clínica organizados por especialidad.
5. Poder pedir citas con los especialistas sólo si se ha iniciado sesión previamente.
6. Permitir a un paciente el poder ver sus medicaciones y su pauta de toma.
7. Permitir a un especialista asignar medicamentos a sus pacientes.
8. Dar la posibilidad a un paciente de subir sus mediciones de tensión arterial y glucosa.
9. Permitir a un especialista llevar el seguimiento de las mediciones realizadas por los pacientes.
10. Conceder a un paciente el poder ver los informes médicos escritos por los especialistas.
11. Autorizar a un especialista el poder escribir el informe de su paciente.
12. Conceder privilegios de administración a un usuario para que pueda gestionar las cuentas de usuario, así como la información visible en las secciones principales de la web.

No funcionales:

1. El desarrollo se realizará utilizando los *frameworks* de Angular para toda la lógica relativa al front-end y de Express.js para la lógica relativa el *backend*.
2. La interfaz debe ser amigable y fácil de utilizar.
3. La interfaz debe ser responsiva pudiéndose adaptar a monitores, tablets y móviles.
4. El código deberá estar correctamente modularizado para poder mejorar su mantenimiento y escalabilidad.
5. El código estará correctamente documentado para facilitar su comprensión.
6. El sistema debe estar disponible en cualquier momento del día.

7. Los pacientes se podrán registrar de forma autónoma pero los especialistas necesitarán de un registro por parte del administrador.
8. Los datos asociados a registros, informes y mediciones serán almacenados de forma segura en la base de datos.

2. PROTOTIPO / MOCKUP

A. Navegación a través de la barra de navegación

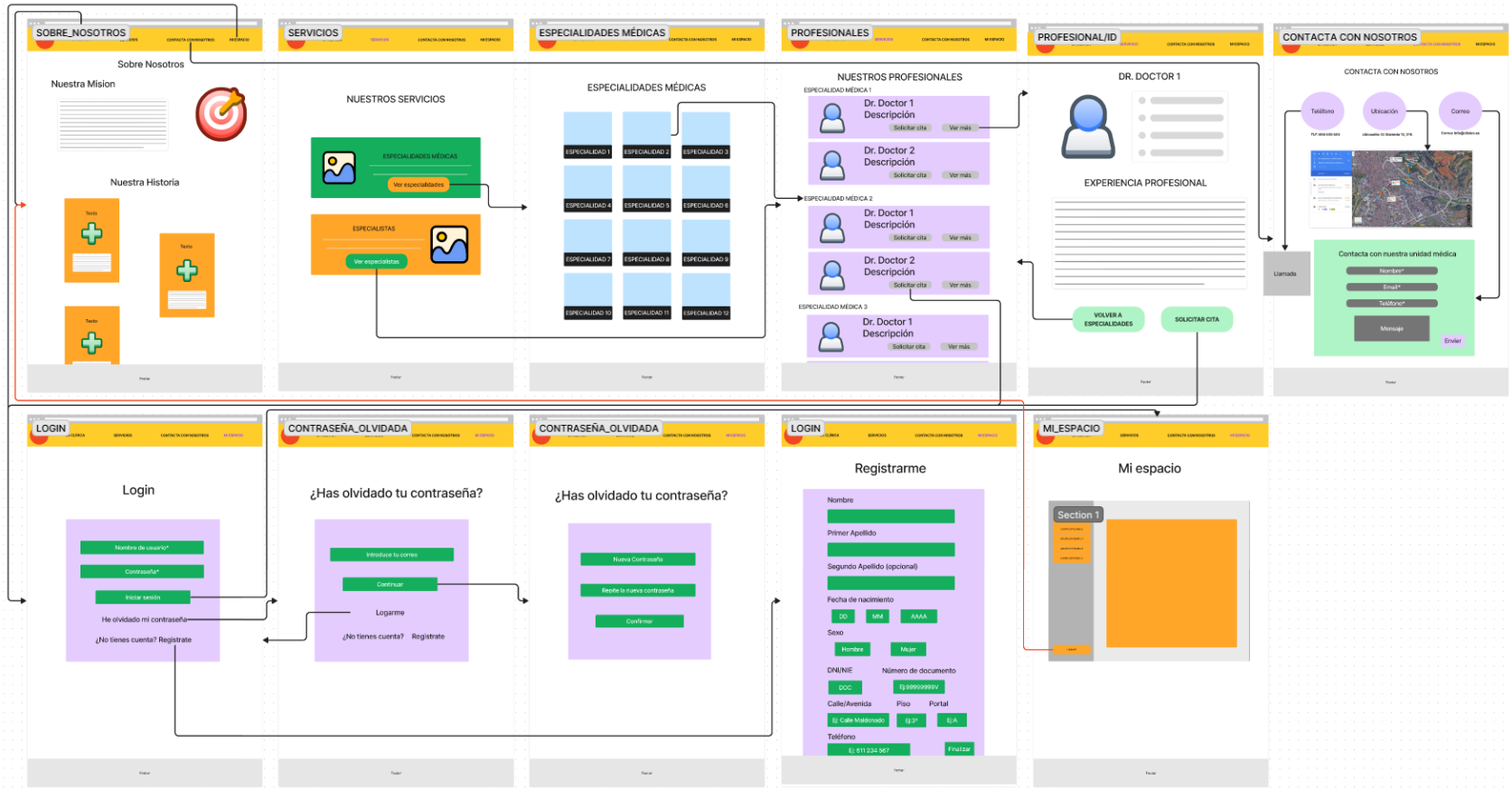

Fig 1 Prototipo de la navegación a través de la barra de navegación de la web.

B. Mi Espacio - Rol Administrador

1) Gestión de usuarios

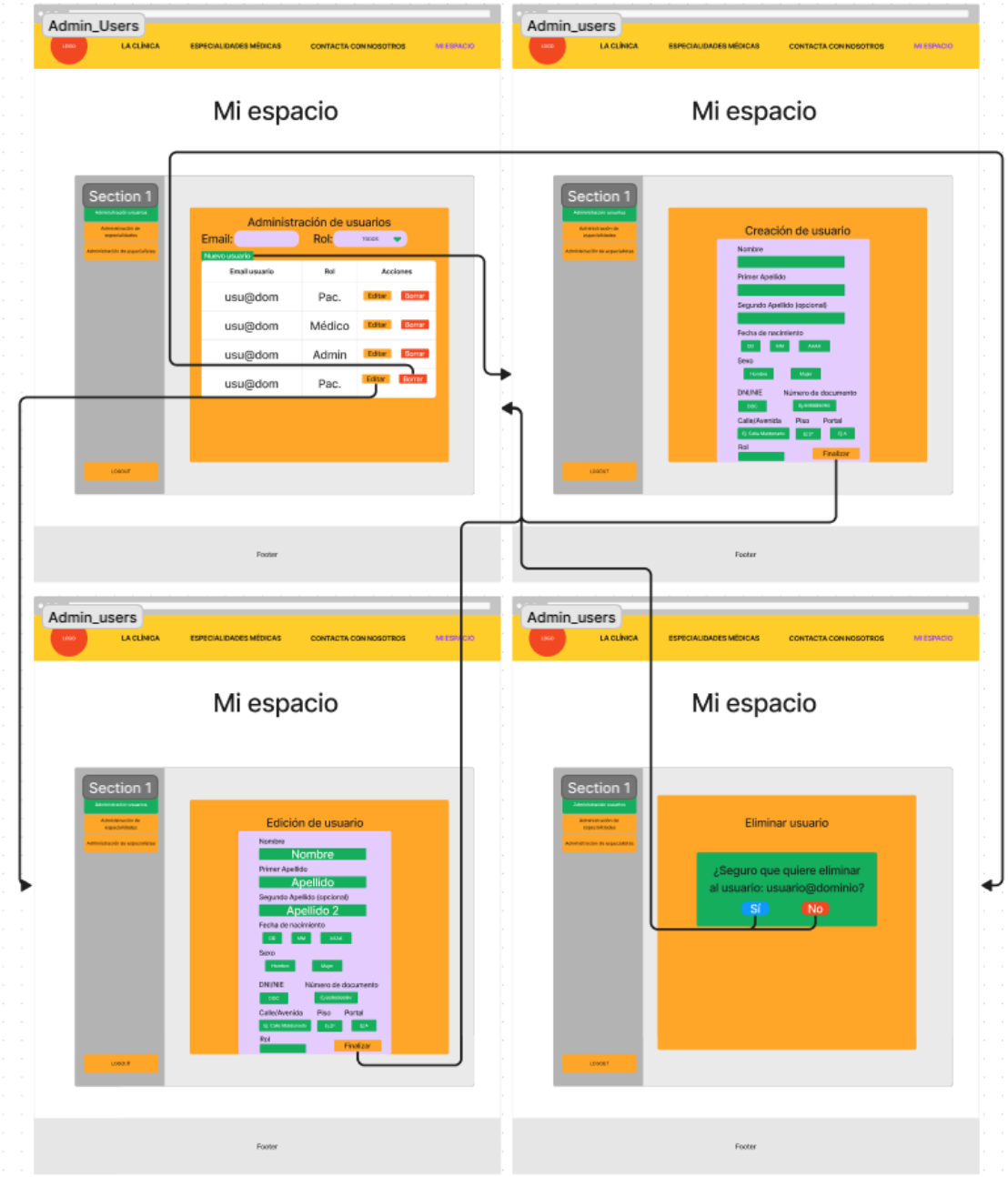

Fig 2 Prototipo para la gestión de usuarios por parte del administrador.

2) Gestión de especialidades

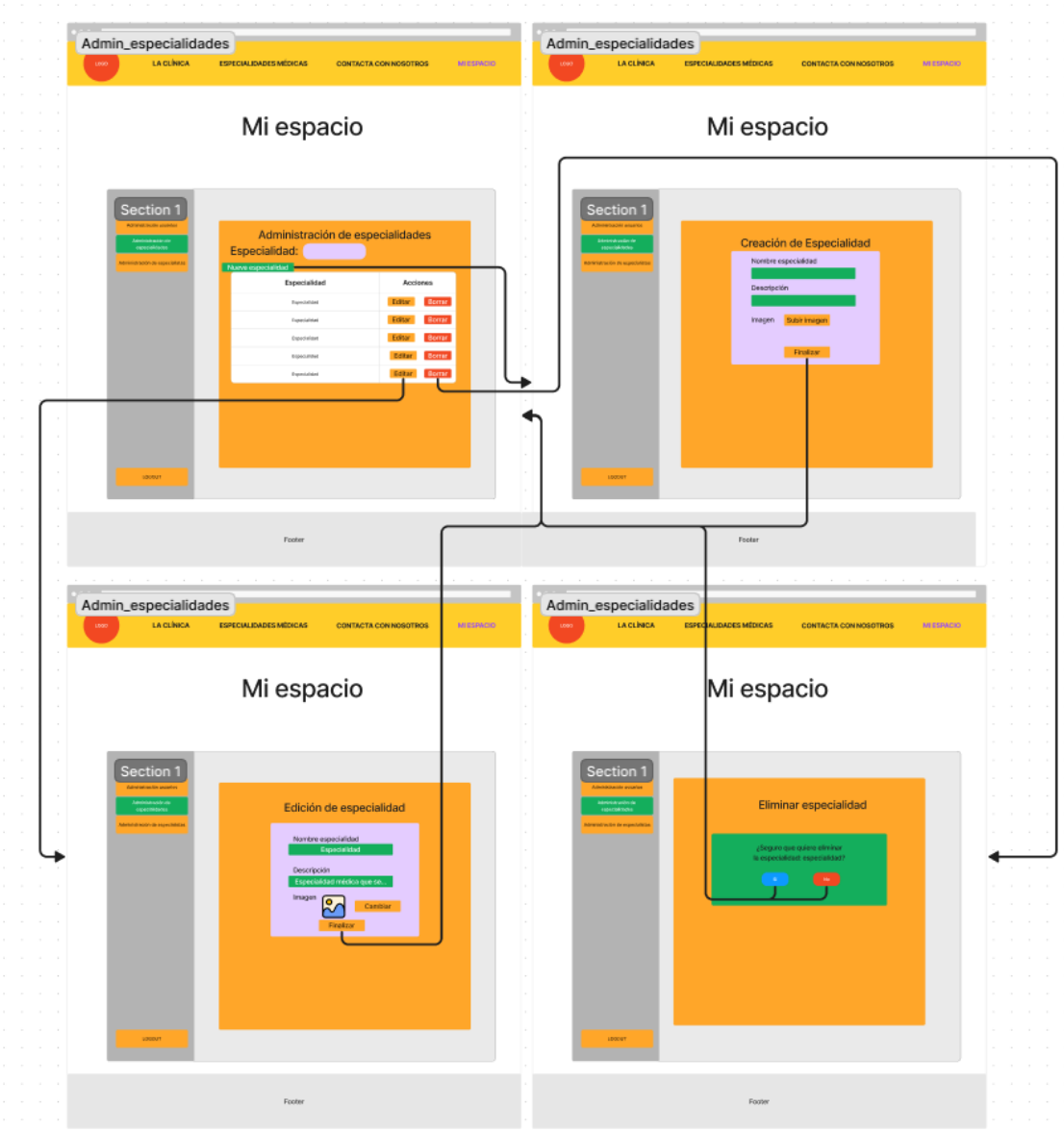

Fig 3 Prototipo para la gestión de especialidades por parte del administrador.

3) Gestión de especialistas

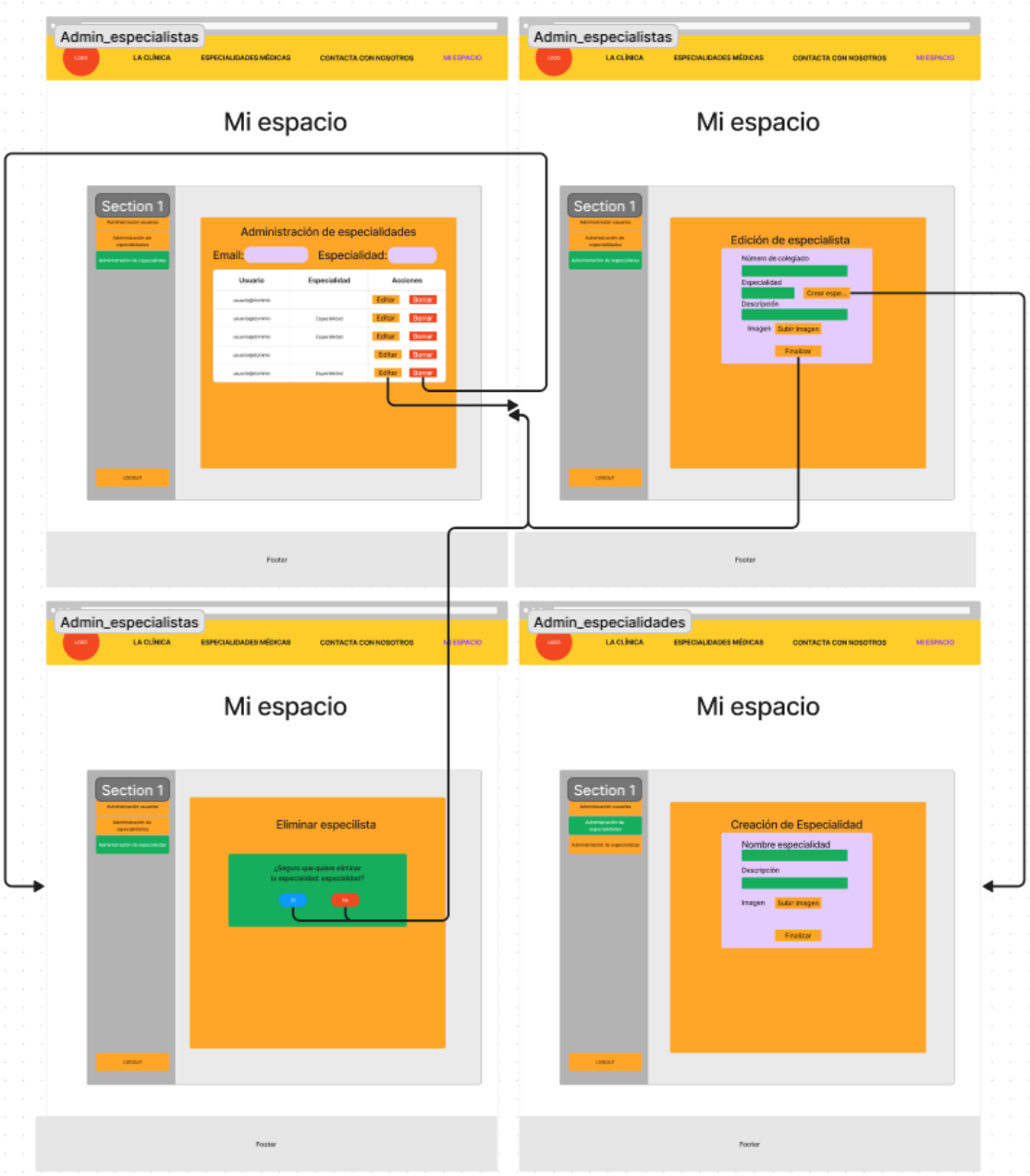

Fig 4 Prototipo para la gestión de especialistas por parte del administrador.

C. Mi Espacio - Rol Especialista

1) Consultar agenda diaria

Fig 5 Prototipo para la consulta de la agenda diaria por parte del especialista.

2) Gestión historia clínica - Medicación

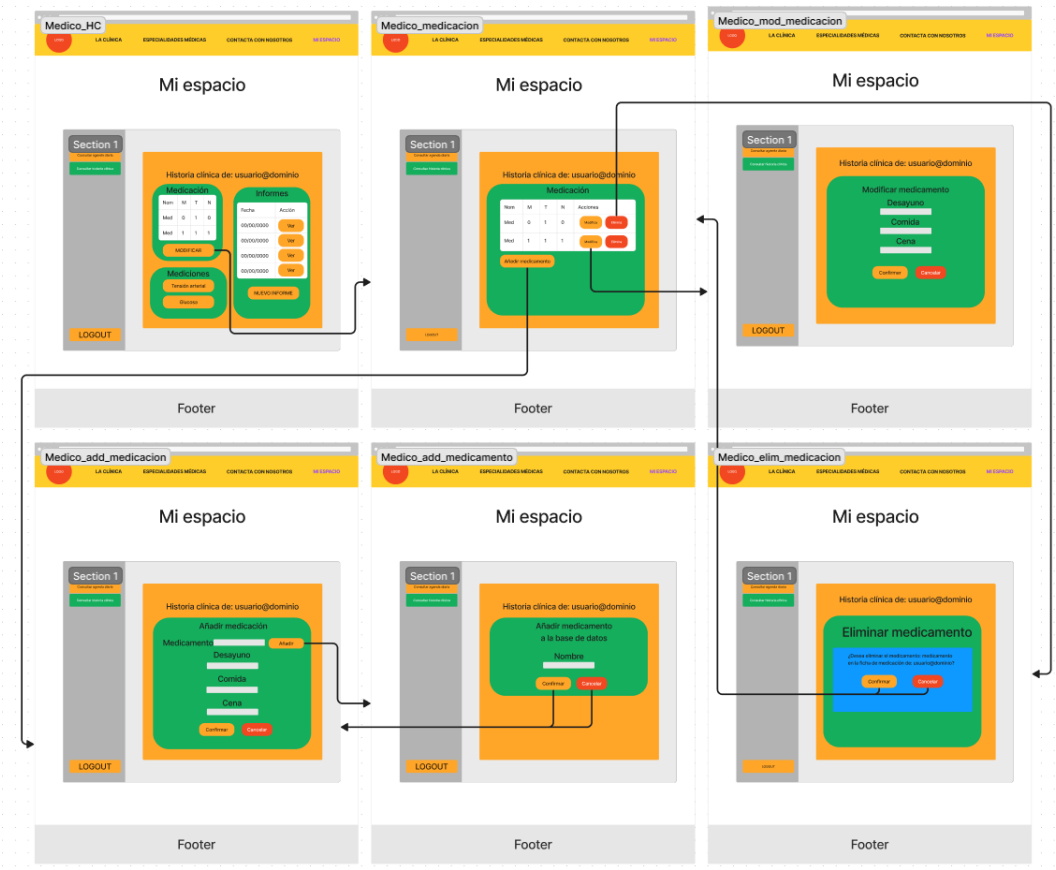

Fig 6 Prototipo para la gestión de la medicación por parte del especialista.

3) Gestión historia clínica - Mediciones de glucosa y tensión arterial

Fig 7 Prototipo para consulta de las mediciones de glucosa y tensión arterial del paciente.

4) Gestión historia clínica - Informes médicos

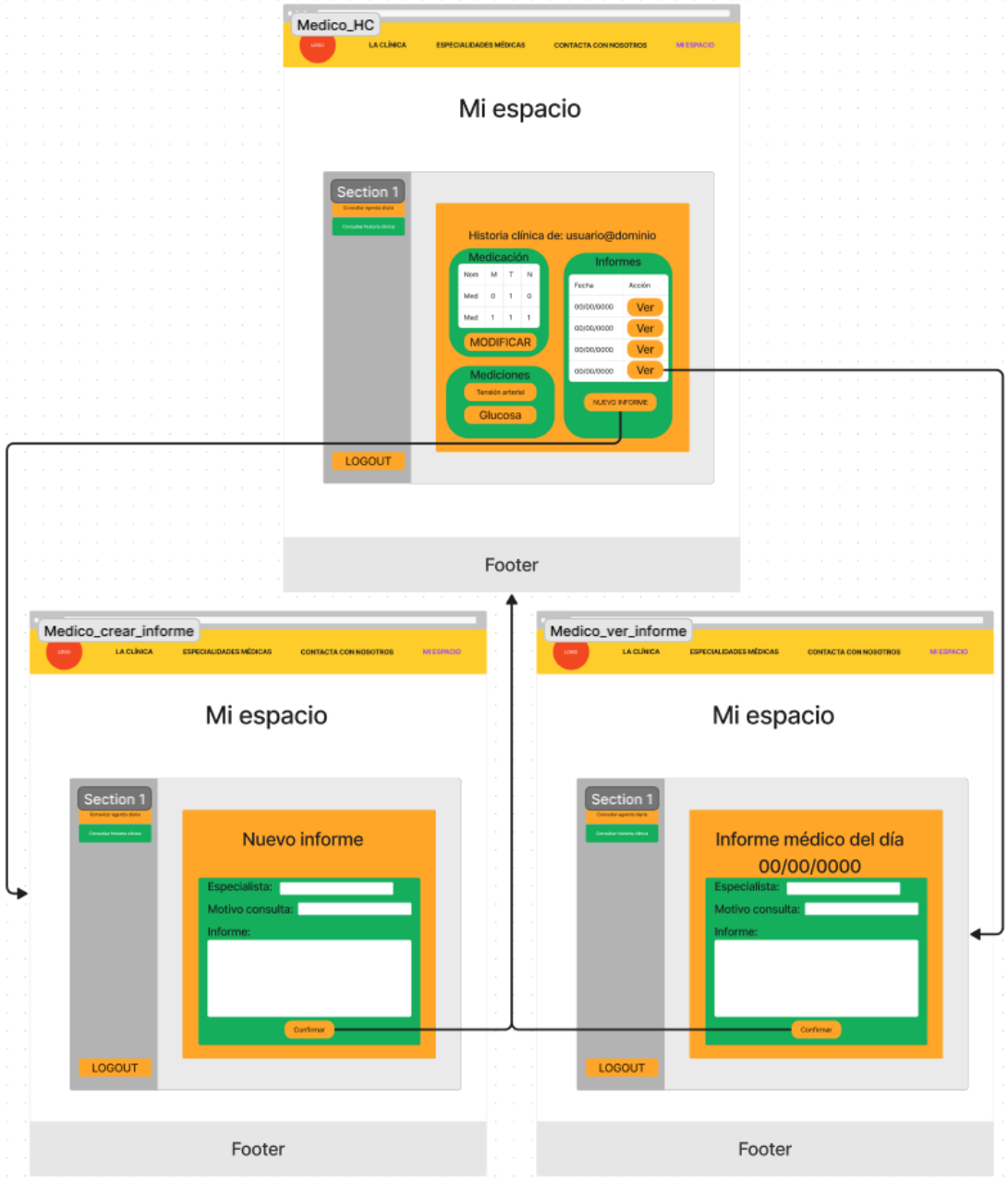

Fig 8 Prototipo para la consulta y generación de nuevos informes médicos.

D. Mi Espacio - Rol Paciente

1) Solicitar cita

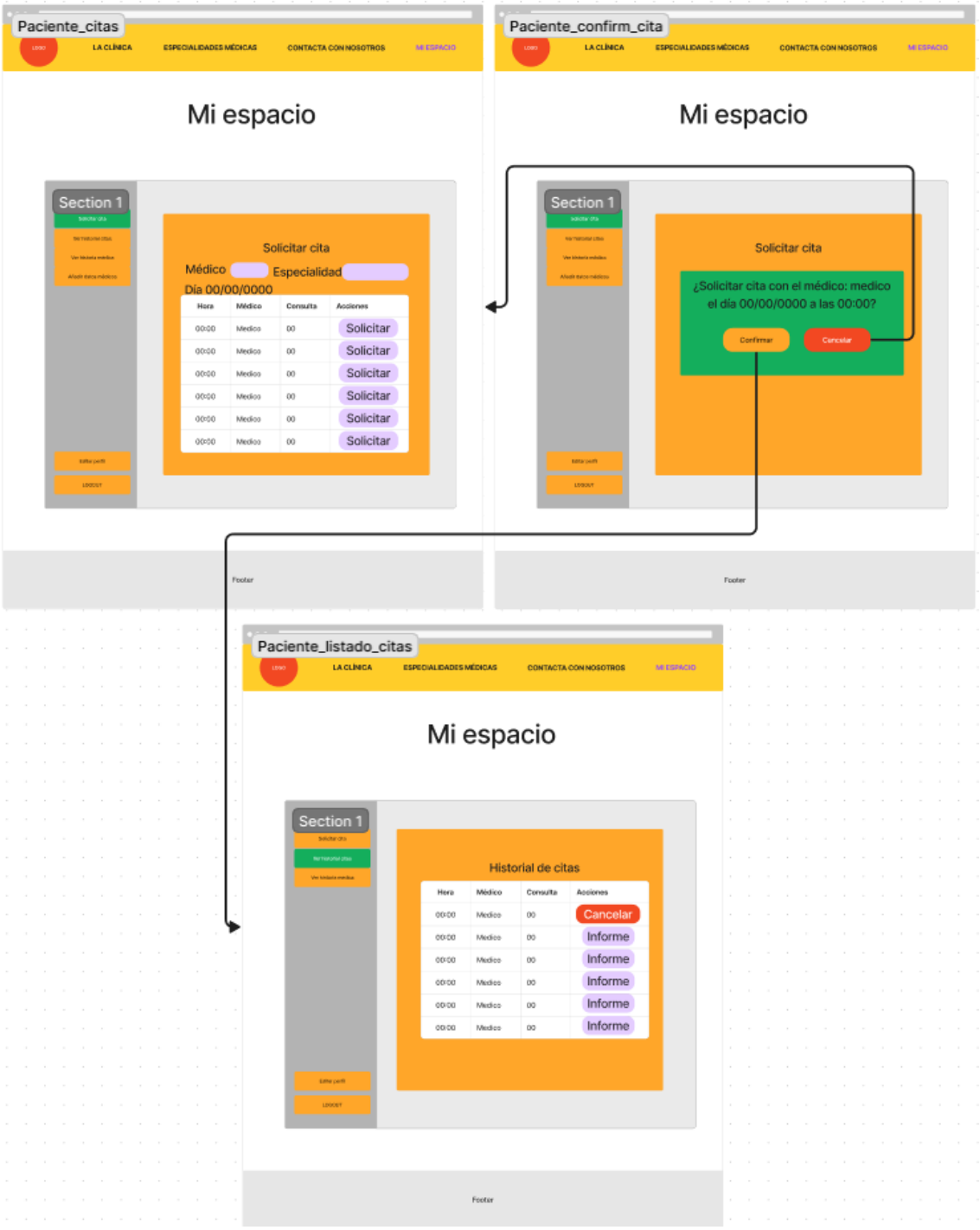

Fig 9 Prototipo para la solicitud de citas con especialistas por parte del paciente.

2) Historial de citas

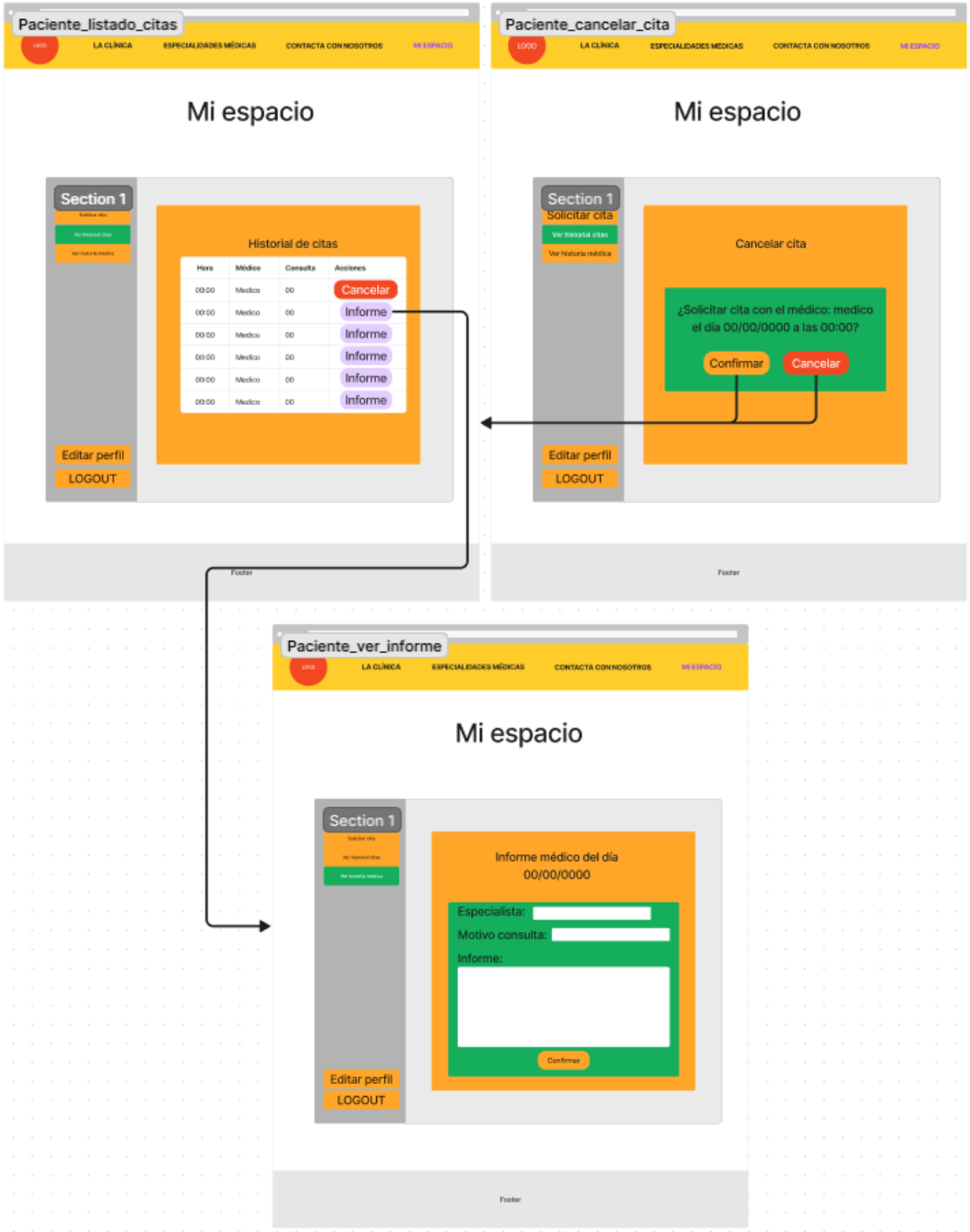

Fig 10 Prototipo para la gestión de citas por parte del paciente.

3) Historial clínico

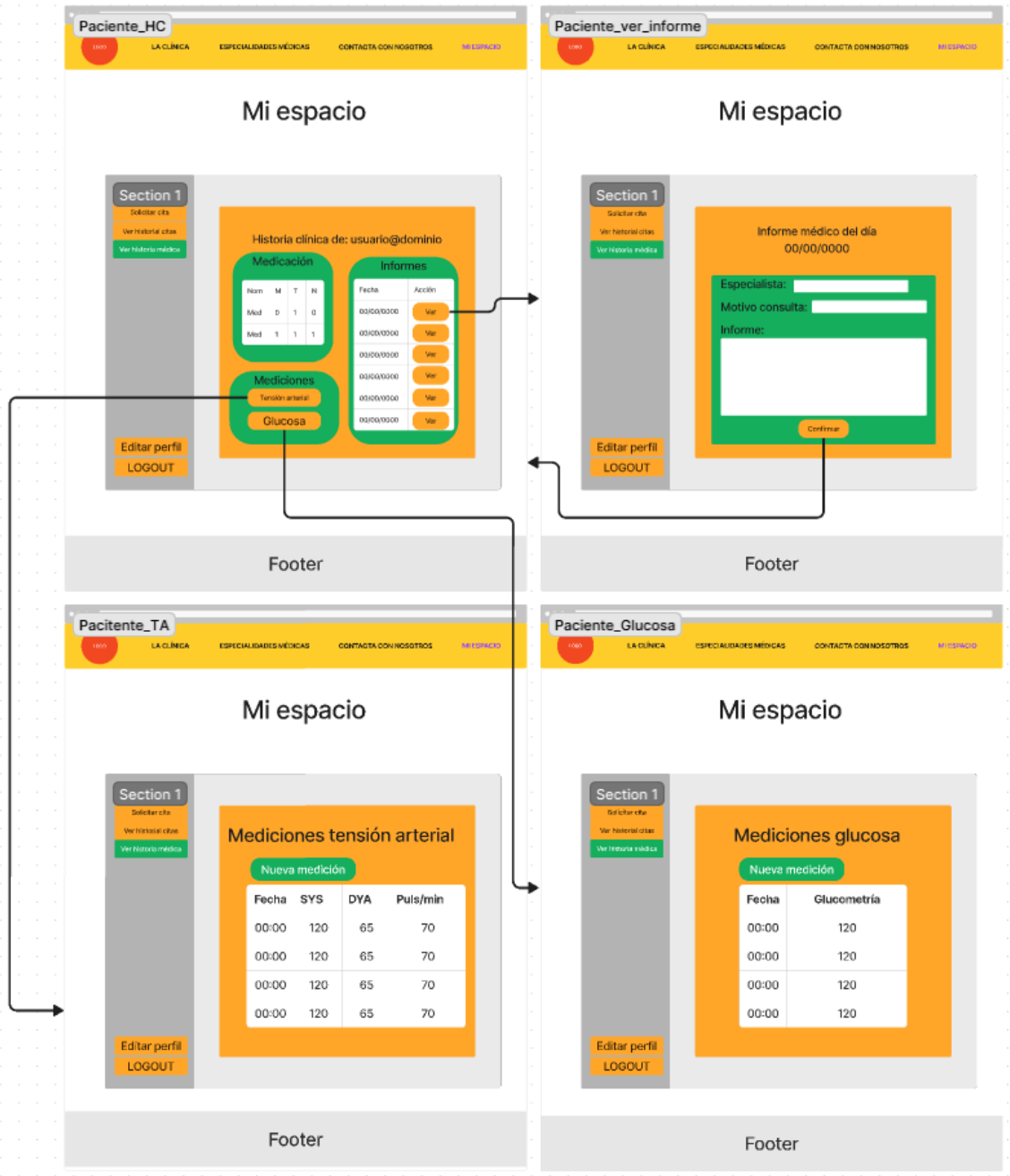

Fig 11 Prototipo para la inserción de datos de mediciones de tensión arterial y glucosa por parte del paciente.

4) Editar perfil

Fig 12 Prototipo para la edición del perfil por parte del paciente.

3. TECNOLOGÍAS USADAS

A. Angular

Angular es un *framework* de desarrollo de aplicaciones web creado por Google, diseñado para facilitar la creación de aplicaciones dinámicas y de una sola página (SPA). Angular se basa en el paradigma de arquitectura de componentes, lo que significa que las aplicaciones se construyen mediante la composición de componentes reutilizables. Estos componentes encapsulan la funcionalidad y la interfaz de usuario de la aplicación, lo que facilita la organización del código y el mantenimiento a medida que la aplicación crece en complejidad.

Fig 13 Logo de Angular.

Hemos utilizado esta tecnología ya que hoy en día es uno de los *frameworks* más utilizados a nivel de desarrollo en cuanto a la parte de *frontend*. Además, debido a su arquitectura de modelo-vista-presentador (MVP) igual que la gran cantidad de funciones que trae incluidas como el enlazado de datos bidireccional, inyección de dependencias, enrutados y servicios entre otros, permite conseguir de manera sencilla, una aplicación escalable, mantenible y eficiente.

B. Figma

Figma es una herramienta de diseño de interfaces de usuario (UI) basada en la nube que permite a los diseñadores crear, colaborar y compartir diseños de aplicaciones web y móviles de manera eficiente. Es una aplicación todo en uno que abarca desde la creación de *wireframes* y prototipos hasta el diseño visual y la generación de especificaciones de diseño.

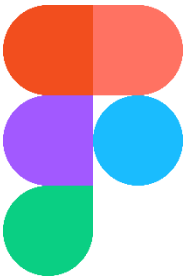

Fig 14 Logo de Figma.

Hemos decidido usar esta tecnología para el maquetado de nuestro sitio web debido a la posibilidad de colaboración en tiempo real remota que ofrece. A su vez, al estar en la nube y ser compatible con diferentes sistemas operativos permite una gran accesibilidad, lo cual junto a su diseño amigable facilita su uso e implementación.

C. Postman

Es una plataforma que permite diseñar, probar, documentar y monitorear interfaces de programación de aplicaciones (APIs) de manera eficiente. A través de la interfaz gráfica de usuario permite la creación y envío de diferentes tipos de solicitudes HTTP (GET, POST, PUT, DELETE…) así como características adicionales como las cabeceras que se van a enviar, el contenido del cuerpo, etc. a APIs. Además, permite otras características más avanzadas como la automatización de pruebas o la generación de documentación automática.

Fig 15 Logo de Postman.

Hemos decidido utilizar esta tecnología porque permite comprobar el funcionamiento de nuestra API REST de una forma rápida y sencilla y sin requerir de tener un *frontend* funcional para realizar solicitudes y ver el resultado de estas.

D. Git

Es un sistema de control de versiones distribuido que permite llevar un registro de los cambios realizados en el código fuente de un proyecto a lo largo del tiempo. Gracias a Git los desarrolladores pueden realizar un seguimiento de las modificaciones realizadas en un proyecto, así como revertir cambios anteriores si es necesario facilitando y haciendo más efectivo el desarrollo colaborativo con otros miembros del equipo de desarrollo.

Fig 16 Logo de Git.

Además, su modelo descentralizado permite que cada desarrollador tenga una copia completa en su propio sistema del historial de cambios que ha ido sufriendo el proyecto, permitiendo de esa manera un flujo de trabajo flexible e independiente que posibilita la sincronización de cambios con el repositorio central cuando sea necesario.

E. GitHub

Es una plataforma de desarrollo colaborativo basada en la nube que utiliza Git como control de versiones. Permite a los desarrolladores almacenar, gestionar y colaborar en proyectos de software de manera eficiente al trabajar directamente sobre un repositorio remoto alojado en la nube.

Fig 17 Logo de GitHub.

La plataforma ofrece además una variedad de herramientas y características como por ejemplo el seguimiento de problemas, la gestión de proyectos, la revisión de código o la integración y el despliegue continuo (CI/CD).

F. Express.js

Es un *framework* de desarrollo de aplicaciones web para Node.js que a su vez se trata de un entorno de ejecución de JavaScript en el lado del servidor. Destaca principalmente por ser ligero y flexible.

Fig 18 Logo de Express.js.

Express.js simplifica la creación de servidores web basados en Node.js al proporcionar una capa de abstracción sobre el servidor HTTP nativo de Node.js, facilitando de esa manera la definición de rutas, el manejo de solicitudes y respuestas, el manejo del *middleware* (función que es ejecutada entre la recepción de una solicitud HTTP y el envío de la respuesta por parte del servidor) y la configuración de la aplicación.

Además, Express.js es altamente extensible, lo que permite integrar multitud de bibliotecas para agregar funcionalidades adicionales cuando sea necesario.

Hemos decidido utilizar esta tecnología porque queríamos aprovechar el trabajo de fin de ciclo para aprender el funcionamiento de una tecnología de *backend* diferente a la vista durante el curso. Eso junto con sus características de alta velocidad y rendimiento, su facilidad de escalabilidad y flexibilidad nos hizo decantarnos por utilizar Node.js junto con Express.js en el *backend*.

G. WebStorm

Es un potente entorno de desarrollo integrado (IDE) desarrollado por la compañía JetBrains y que está diseñado específicamente para el desarrollo de aplicaciones web que utilizan tecnologías como HTML, CSS, JavaScript y Typescript, así como *frameworks* relacionados como Angular, React, NestJS o Express.js.

Fig 19 Logo de WebStorm.

Proporciona un conjunto de herramientas que ayudan a escribir, editar, depurar y refactorizar código de una manera eficiente, rápida y simple.

Entre sus características principales destacan su autocompletado inteligente, su análisis de código estático, la depuración integrada, la fácil integración con sistemas de control de versiones y el soporte para *frameworks* y bibliotecas de código.

H. MySQL

MySQL es un sistema de gestión de bases de datos relacional (organiza los datos en tablas relacionadas entre sí) de código abierto ampliamente utilizado en todo el mundo. Ofrece una sólida combinación de rendimiento, confiabilidad y facilidad de uso, lo que lo convierte en una opción popular en el desarrollo de aplicaciones.

Fig 20 Logo de MySQL.

Hemos utilizado esta base de datos debido a la escalabilidad que permite al poder manejar un gran volumen de datos al igual que de cantidad de usuarios concurrentes se refiere. Además, ofrece una amplia gama de características de seguridad como la encriptación de datos, autenticación de usuarios y permisos, lo que permite que sea una base de datos muy versátil y fácil de integrar en diferentes entornos.

I. MySQL Workbench

MySQL Workbench es una herramienta gráfica de diseño y administración de bases de datos que se utiliza junto con MySQL para simplificar tareas de desarrollo y administración.


Fig 21 Logo de MySQL Workbench.

Lo hemos utilizado debido a que ofrece una interfaz intuitiva que permite a los desarrolladores y administradores de bases de datos realizar diversas tareas como diseño de esquemas, consulta y manipulación de datos, y optimización de consultas de manera gráfica, entre otras.

J. *Procedural Language / Structured Query Language* (PL/SQL)

PL/SQL es un lenguaje de programación procedimental que sirve como extensión del estándar SQL y permite incluir capacidades de programación procedural. Con P/-SQL los desarrolladores de bases de datos pueden escribir bloques de código que pueden realizar diversas acciones tales como la manipulación de datos o el control de flujo de ejecución.

Fig 22 Logo de PL/SQL.

Hemos decidido usar esta tecnología ya que con ella podemos conseguir automatizar procesos en la base de datos de nuestra aplicación, por ejemplo, generar eventos que a una hora determinada del día lleven a cabo el truncado de una tabla o que llamen a un determinado procedimiento que lleve a cabo borrados secuenciales de datos que ya no sean necesarios como por ejemplo prescripciones de medicamentos que ya no están activas al haber superado la fecha de finalización del tratamiento.

K. Bootstrap

Bootstrap es un popular *framework* de código abierto para desarrollo *frontend*, utilizado para crear interfaces web y aplicaciones con mayor rapidez y eficiencia.

Fig 23 Logo de Bootstrap.

Hemos decidido utilizarlo ya que ofrece una gran variedad de componentes y estilos predefinidos que pueden ser utilizados directamente en el sitio web como son botones, formularios, modales y barras de navegación entre otros. A su vez, es una gran herramienta al facilitar la creación de diseños responsivos que se adaptan automáticamente al ancho de la pantalla al igual que es fácil de utilizar ya que se usan clases CSS intuitivas para aplicar los estilados.

L. Handlebars.js

Handlebars.js es un motor de plantillas JavaScript que simplifica la generación de HTML al permitir la creación de plantillas de forma más organizada y eficiente.

Fig 24 Logo de Handlebars.js

Hemos utilizado esta tecnología ya que permite reutilizar fragmentos de HTML en múltiples partes de la aplicación sin necesidad de duplicar el mismo código HTML. A su vez facilita la inserción dinámica de datos en las plantillas, permitiendo vincular datos a tus plantillas y luego renderizarlas con los datos específicos, lo que es especialmente útil en aplicaciones web dinámicas donde los datos cambian frecuentemente.

M. *Sassy Cascading StyleSheets* (SCSS)

SCSS es una extensión de CSS y una evolución de *Syntactically Awesome Stylesheets* (SASS) que ofrece una sintaxis más avanzada y poderosa para la escritura de hojas de estilo en la web. Introduce características adicionales que no están presentes en CSS tradicional, como variables, anidamiento, mixins, herencia y operaciones matemáticas, lo que permite a los desarrolladores escribir estilos de manera más modular.

Fig 25 Logo de SASS-SCSS.

Hemos decido añadir esta tecnología como forma de estilado de nuestro proyecto para tener una mayor flexibilidad y adaptabilidad de nuestras hojas de estilo al generar gracias a ella código de estilado menos repetitivo y más modularizado.

N. *JavaScript Object Notation Web Tokens* (JWT)

Los JSON Web Tokens o JWT son un pequeño paquete de información seguro y compacto que permite transmitir datos entre dos partes (*frontend* y *backend*) de forma segura. Está formado por tres partes: el encabezado que describe el tipo de token y el algoritmo de firma que se ha utilizado, la carga útil que contiene la información que se quiere transmitir y la firma que se utiliza para verificar que el mensaje no ha sido alterado por ninguna de las partes.

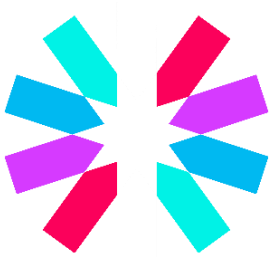

Fig 26 Logo de JSON Web Token.

Hemos decidido utilizarlo ya que se está convirtiendo actualmente en el estándar de aplicaciones web y APIs para la autenticación y transferencia de información entre cliente y servidor. En nuestro caso nos sirve principalmente como mecanismo de defensa contra la intrusión al utilizarlo como barrera de autenticación a través de *middlewares* de Node.js/Express.js.

O. Swagger

Swagger es una herramienta de código abierto que se utiliza para diseñar, construir, documentar y consumir servicios web REST. Permite describir la funcionalidad de una API de una manera clara y estructurada, utilizando un formato JSON o YAML llamado *OpenAPI Specification*.

Fig 27 Logo de Swagger.

En estos ficheros se definen *endpoints* de su API, los parámetros que aceptan, los tipos de datos que devuelven, los códigos de respuesta, entre otros detalles.

Hemos decidido utilizarlo porque es una forma sencilla y ágil de generar documentación automatizada.

P. Swagger UI

Swagger UI es una interfaz de usuario generada automáticamente a partir de los documentos JSON o YAML de Swagger. Utilizando estos documentos genera una interfaz web dinámica que permite explorar y probar los *endpoints* de la API directamente desde el navegador.

Fig 28 Logo de Swagger UI.

Esta interfaz muestra un listado de los *endpoints* de la API junto con detalles sobre los parámetros que aceptan y los códigos de respuesta que devuelven. Así mismo permite a los usuarios enviar solicitudes HTTP a la API rellenando formularios para comprobar el comportamiento en tiempo real de esta.

Hemos decido utilizar esta tecnología porque nos parece una manera interesante, dinámica y atractiva de mostrar la documentación de la API no sólo mostrando los datos de esta sino también permitiendo el ejecutar llamadas a la API para comprobar respuesta.

Q. JSDoc

JSDoc es una convención y una herramienta para documentar código JavaScript. Permite a los desarrolladores describir de manera clara y estructurada el comportamiento y la estructura del código a través de comentarios especiales en el código fuente que luego son procesados por la herramienta para generar documentación legible de forma automática a través de ficheros HTML.

Fig 29 Logo de JsDoc.

Esta herramienta permite agregar anotaciones a los comentarios (@function, @param, @return, @public, @description, @async…) que proporcionan información extra sobre los tipos de datos que se esperan recibir o retornar, el tipo de función del que se trata, su modificador de acceso, etc.

La decisión de incluir esta tecnología es porque queríamos documentar no sólo las rutas de la API sino también las clases y métodos utilizados para mejorar la mantenibilidad y compresión del código.

R. Mocha, Chai y Supertest

Mocha, Chai y Supertest son herramientas para realizar pruebas y testeos en aplicaciones Node.JS y en especial en APIs.

Fig 30 Logos de Mocha, Chai y Supertest.

a) **Mocha**: Es un framework de pruebas unitarias y de integración para JS diseñado para ser simple, flexible y fácil de usar. Mocha proporciona una estructura para escribir y ejecutar pruebas de forma organizada permitiendo definir suites de pruebas, casos de prueba, etc.

b) **Chai**: Es una biblioteca de aserciones para Node.JS que permite la utilización de *expect, should, assert…* y, de esa forma, posibilitar elegir el estilo con el que se prefieren escribir las pruebas de una manera legible y comprensible.

c) **Supertest**: Es una biblioteca de pruebas HTTP para Node.JS que se utiliza principalmente para realizar pruebas de integración APIs RESTful. Permite a los desarrolladores realizar solicitudes HTTP a su API y realizar aserciones sobre las respuestas recibidas, facilitando de esa forma la automatización de pruebas extremo a extremo y la verificación del comportamiento correcto de la API.

En conjunto son herramientas con un alto potencial y proporcionan un conjunto de funcionalidades para escribir, organizar y ejecutar pruebas en aplicaciones JS y APIs.

4. DIAGRAMAS DE ENTIDAD-RELACIÓN

A. Diagrama de Chen

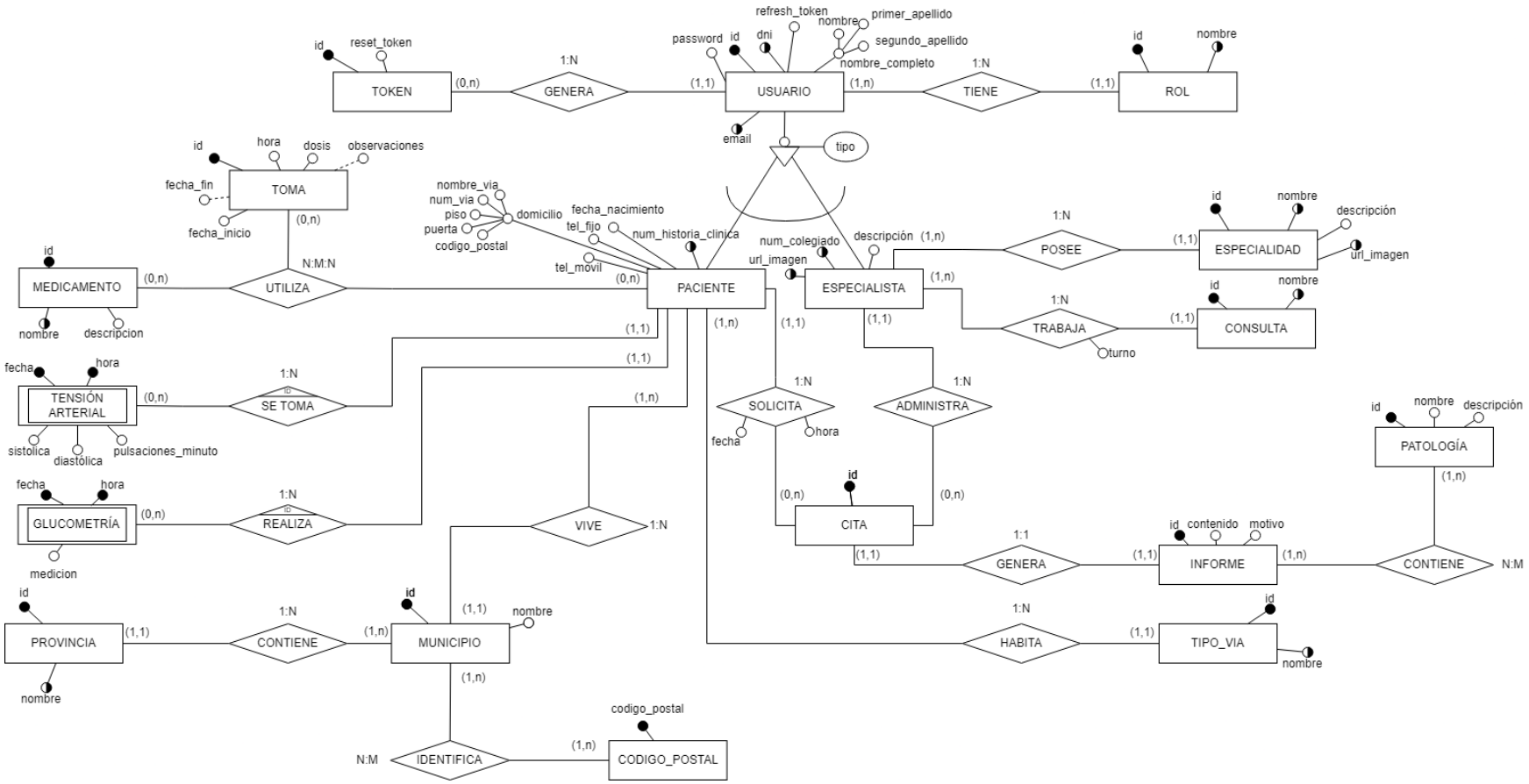

Fig 31 Diagrama de entidad - relación (diagrama de Chen).

B. Diagrama de estructura de datos

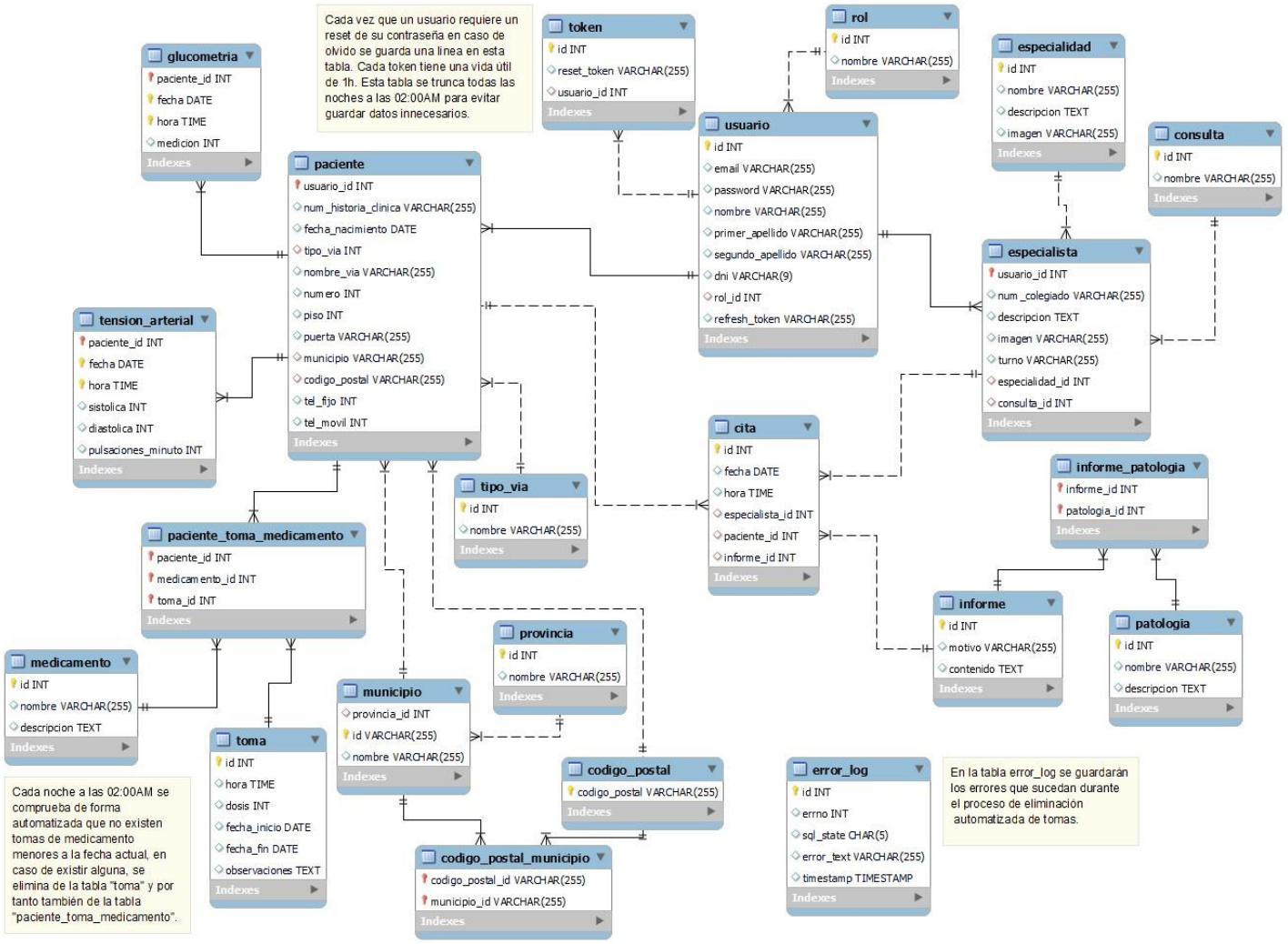

Fig 32 Diagrama de entidad-relación (diagrama de estructura de datos).

5. DIAGRAMA DE CASOS DE USO

A. Herencia de actores

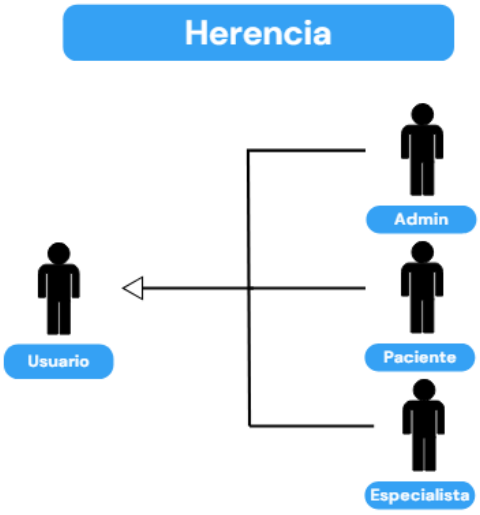

Fig 33 Herencia de actores del diagrama de casos de uso.

B. Casos de uso del usuario

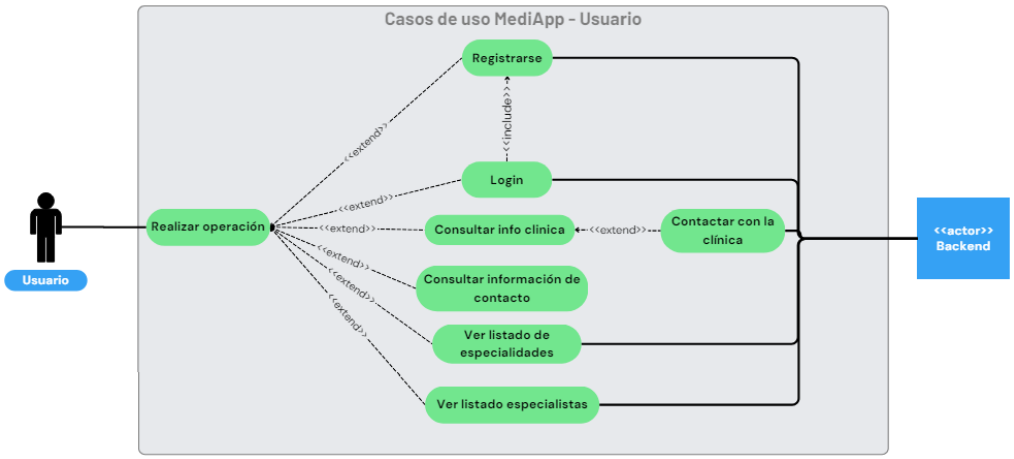

Fig 34 Diagrama de casos de uso del usuario.

C. Casos de uso del administrador

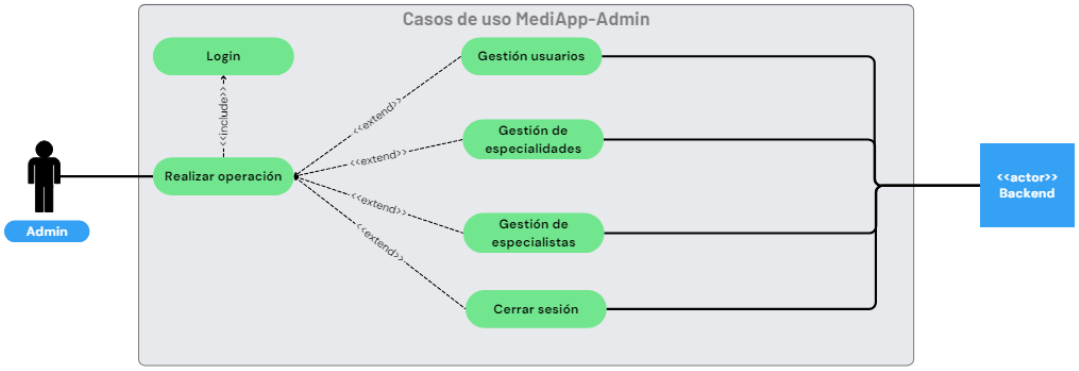

Fig 35 Diagrama de casos de uso del administrador.

D. Casos de uso del paciente

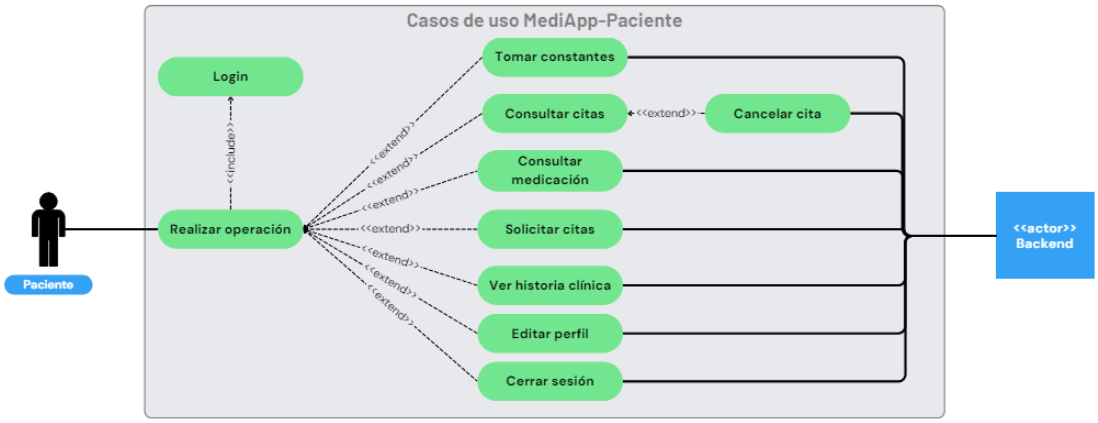

Fig 36 Casos de uso del paciente.

E. Casos de uso del especialista

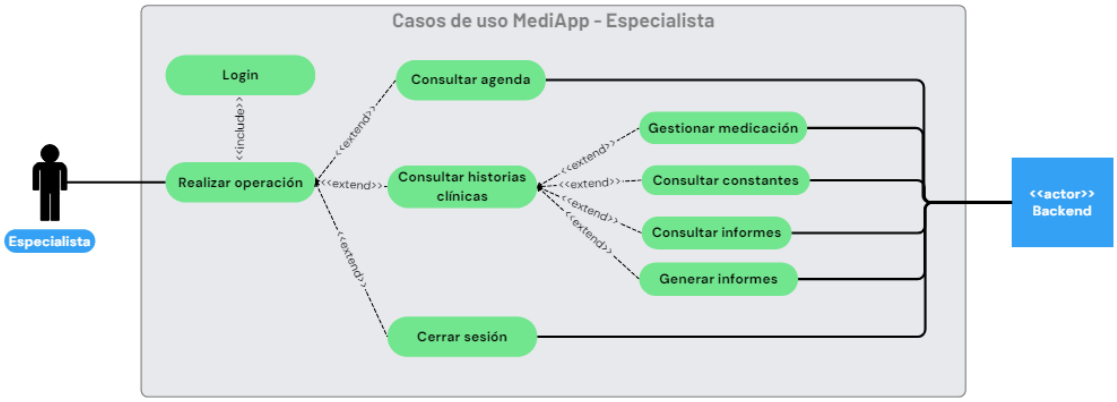

Fig 37 Casos de uso del especialista

MARCO PRÁCTICO

CONCLUSIONES

BIBLIOGRAFÍA

WEBGRAFÍA