

UNIVERSIDAD DE LA REPÚBLICA URUGUAY



FACULTAD DE INGENIERÍA

Salud en cercanía

Informe de Proyecto de Grado presentado por

Santiago Guerra, Sebastián Piceda, Nicolás Zorrozúa

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisora

Raquel Sosa

Montevideo, 21 de marzo de 2026



Salud en cercanía por Santiago Guerra, Sebastián Piceda, Nicolás Zorrozúa tiene licencia CC Atribución 4.0.

Resumen

En el contexto actual, el acceso equitativo a servicios de salud constituye un desafío relevante tanto para la ciudadanía como para las instituciones públicas y privadas. La falta de información georreferenciada, actualizada y confiable sobre los centros de atención médica dificulta la planificación, la gestión de recursos y la toma de decisiones informadas por parte de la población.

El presente proyecto propone el desarrollo de una plataforma web denominada Salud en Cercanía, cuyo objetivo es facilitar la visualización y el análisis de los servicios de salud en Uruguay, priorizando la información sobre centros médicos y policlínicas. La solución permite a los usuarios acceder a datos geográficos de los prestadores de salud, visualizar centros en un mapa interactivo y contribuir con información mediante mecanismos de Volunteered Geographic Information (VGI).

La propuesta se compone de un frontend web, desarrollado en React y Next.js, que ofrece interfaces diferenciadas para usuarios y administradores. Además, un backend basado en Python, encargado de la gestión de datos, autenticación y comunicación con la base de datos geográfica. Sumado a esto, se integran herramientas de Leaflet y OpenStreetMap para la representación espacial de los centros de salud.

La plataforma incorpora tres roles principales: Usuario, Administrador de plataforma y Administrador de prestador de salud, cada uno con funcionalidades específicas. Los usuarios pueden consultar servicios, agregar nuevos centros pendientes de validación y dejar reseñas, mientras que los administradores se encargan de validar la información ingresada, gestionar la base de datos de centros y visualizar estadísticas de uso.

El prototipo desarrollado busca mejorar la accesibilidad a la información sanitaria, fomentar la participación ciudadana y apoyar la toma de decisiones territoriales basadas en datos. Su diseño modular y extensible permite su adaptación a diferentes dominios que requieran gestión colaborativa de datos geográficos.

Palabras clave: Sistemas de Información Geográfica, Salud, Datos espaciales, VGI, Leaflet, OpenStreetMap, React, Next.js, Python

Índice general

1. Introducción **1**
- 1.1. Motivación . . . 1
- 1.2. Objetivos . . . 1
- 1.3. Aportes del proyecto . . . 2
- 1.4. Organización del documento . . . 3

2. Revisión de antecedentes **4**
- 2.1. Marco Teórico . . . 4
  - 2.1.1. Datos e Información Geográfica . . . 4
  - 2.1.2. Sistemas de Información Geográfica (GIS) . . . 5
  - 2.1.3. Información Geográfica Voluntaria (VGI) . . . 5
- 2.2. Trabajos Relacionados . . . 5
  - 2.2.1. Healthsites.io . . . 5
  - 2.2.2. AtuServicio . . . 8
  - 2.2.3. Doctoralia . . . 10
  - 2.2.4. Medicare . . . 12
- 2.3. Comparación de Plataformas . . . 14

3. Análisis y Requerimientos **16**
- 3.1. Contexto de trabajo . . . 16
- 3.2. Actores . . . 17
  - 3.2.1. Usuario . . . 17
  - 3.2.2. Administrador de Prestador de Salud . . . 17
  - 3.2.3. Administrador de Plataforma . . . 17
- 3.3. Vista de Casos de Uso . . . 18
- 3.4. Requerimientos Funcionales . . . 20
  - 3.4.1. Autenticación y gestión de cuenta . . . 20
  - 3.4.2. Búsqueda y visualización . . . 20
  - 3.4.3. Favoritos, reseñas y notificaciones . . . 20
  - 3.4.4. Contribución de datos . . . 21
  - 3.4.5. Administración de prestador de salud . . . 21
  - 3.4.6. Administración de plataforma . . . 21
- 3.5. Requerimientos No Funcionales . . . 21
  - 3.5.1. Seguridad . . . 21

3.5.2. Usabilidad . . . 22
3.5.3. Portabilidad . . . 22
3.5.4. Mantenibilidad y extensibilidad . . . 22

4. Propuesta de solución **23**
4.1. Arquitectura General del Sistema . . . 23
4.1.1. Vista lógica . . . 23
4.1.2. Estilo arquitectónico . . . 25
4.1.3. Subsistemas . . . 25
4.1.4. Vista de distribución . . . 26
4.1.5. Escalabilidad y seguridad . . . 26
4.1.6. Modelo de Dominio . . . 27
4.1.7. Flujos principales del sistema . . . 30
4.2. Análisis de Datos Externos . . . 32
4.2.1. Healthsites.io . . . 32
4.2.2. A Tu Servicio . . . 33

5. Implementación **35**
5.1. Tecnologías Utilizadas . . . 35
5.2. Arquitectura del sistema . . . 36
5.3. Backend . . . 36
5.3.1. Estructura del proyecto . . . 36
5.3.2. Capa de API . . . 37
5.3.3. Capa de servicios . . . 37
5.3.4. Acceso a datos . . . 38
5.4. Base de Datos . . . 38
5.4.1. Motor y extensiones . . . 38
5.4.2. Diseño del esquema . . . 38
5.4.3. Consultas geoespaciales . . . 38
5.5. Frontend . . . 39
5.5.1. Estructura de páginas . . . 39
5.5.2. Estado global con contextos . . . 40
5.5.3. Visualización geográfica . . . 41
5.5.4. Comunicación con el backend . . . 41
5.6. Autenticación y autorización . . . 41
5.6.1. JWT y cookies . . . 41
5.6.2. Control de acceso por roles . . . 41
5.6.3. Recuperación de contraseña . . . 42
5.7. Despliegue . . . 42
5.7.1. Backend . . . 43
5.7.2. Frontend . . . 44
5.7.3. Base de Datos . . . 44

6. Pruebas y Validaciones **45**
6.1. Casos de Prueba . . . 45
6.1.1. Autenticación y gestión de cuenta . . . 45
6.1.2. Visualización e interacción con el mapa . . . 45
6.1.3. Favoritos, reseñas y notificaciones . . . 46
6.1.4. Contribución VGI . . . 46
6.1.5. Panel del administrador de prestador . . . 46
6.1.6. Panel del administrador de plataforma . . . 47
6.2. Validación con usuarios . . . 47
6.3. Resultados de las pruebas . . . 48
6.3.1. Defectos de configuración y despliegue . . . 48
6.3.2. Defectos de errores en código . . . 49

7. Gestión del proyecto **50**
7.1. Etapas de la Gestión . . . 50
7.2. Planificación . . . 51
7.3. Herramientas . . . 54
7.4. Desviaciones . . . 54

8. Conclusiones y Trabajo Futuro **56**
8.1. Conclusiones . . . 56
8.2. Trabajo futuro . . . 57

Referencias **59**

A. Casos de uso **60**
A.1. Casos de Uso . . . 60

B. Testing exploratorio **71**
B.1. Testing Exploratorio Planificado . . . 71
B.1.1. Registro de usuario . . . 71
B.1.2. Logueo del usuario . . . 72
B.1.3. Logout del usuario . . . 72
B.1.4. Gestión de contraseña . . . 73
B.1.5. Casos de prueba para un usuario común con cuenta registrada . . . 73
B.2. Testing Exploratorio Basado en Sesiones . . . 74
B.2.1. Sesión 1: Experiencia de Usuario invitado . . . 75
B.2.2. Sesión 2: Experiencia de Usuario logueado . . . 76
B.2.3. Sesión 3: Experiencia de un Administrador de Prestador . . . 78
B.2.4. Sesión 4: Experiencia de un administrador de plataforma 79

C. Manual de usuario **80**
C.1. Introducción . . . 80
C.2. Usuario invitado . . . 80
C.3. Usuario común registrado en la aplicación . . . 84

C.4. Administrador de Prestador 95
C.5. Administrador de Plataforma 98

Capítulo 1

Introducción

1.1. Motivación

En la actualidad, las organizaciones enfrentan el desafío de gestionar grandes volúmenes de información vinculada a servicios públicos, entre ellos los relacionados con la salud. La distribución geográfica de los prestadores de servicios de salud, junto con la disponibilidad de recursos y la accesibilidad de la población a los mismos, constituyen factores clave para la planificación y toma de decisiones en políticas sanitarias.

Sin embargo, gran parte de esta información se encuentra dispersa o poco integrada, lo que dificulta su análisis desde una perspectiva territorial. A su vez, los sistemas actuales suelen requerir conocimientos técnicos especializados en Sistemas de Información Geográfica (GIS) o en programación para poder procesar y visualizar los datos de manera efectiva.

Ante esta problemática, surge la necesidad de desarrollar una plataforma que facilite la **recolección, visualización y análisis de información geográfica sobre prestadores de salud**, a partir de la **colaboración ciudadana (VGI – Volunteered Geographic Information)**. De esta forma, se busca promover la participación de los usuarios en la construcción de información geoespacial relevante, fomentando la transparencia y el acceso a datos actualizados, sin necesidad de poseer conocimientos avanzados en GIS.

1.2. Objetivos

El objetivo general del proyecto es **diseñar e implementar un prototipo de sistema web** que permita la **captura, gestión y visualización de información georreferenciada sobre servicios de salud**, integrando datos provenientes tanto de fuentes oficiales como de aportes ciudadanos.

Para cumplir con este objetivo general, se definen los siguientes **objetivos específicos**:

- Analizar el contexto actual de los sistemas de información geográfica aplicados a la gestión de servicios de salud.
- Identificar los requerimientos funcionales y no funcionales del sistema propuesto.
- Diseñar una arquitectura web distribuida que integre un backend, una base de datos geográfica y un frontend interactivo.
- Implementar un prototipo funcional que permita registrar, consultar y visualizar prestadores de salud en un mapa.
- Incorporar mecanismos de autenticación y control de acceso diferenciados según los roles de usuario.
- Validar la adecuación técnica y funcional del sistema mediante casos de uso simulados.

1.3. Aportes del proyecto

Los principales aportes de este trabajo son:

- La **definición de una arquitectura de software** orientada a la gestión colaborativa de datos geográficos en el dominio de la salud, basada en tecnologías abiertas y escalables.
- La **implementación de un prototipo web** que permite a los usuarios contribuir con información sobre prestadores de salud, mientras que los administradores pueden revisar, validar y publicar dichos datos.
- El **desarrollo de una interfaz de visualización** interactiva que utiliza mapas dinámicos para explorar los servicios disponibles según ubicación, tipo y cobertura.
- La integración de un **modelo de autenticación de usuarios y control de permisos**, que diferencia entre usuarios generales, administradores de prestadores de salud y administradores de plataforma.
- La **documentación detallada del sistema**, incluyendo manuales de uso y despliegue, que facilitan la replicabilidad y extensión futura del proyecto.
- La **validación del prototipo** en un contexto simulado de gestión de servicios de salud, demostrando su viabilidad técnica y su potencial para fortalecer la planificación territorial del sistema sanitario.

1.4. Organización del documento

El resto del documento se estructura de la siguiente manera:

- **Capítulo 2: Revisión de antecedentes.** Se introduce el marco teórico sobre datos geográficos, Sistemas de Información Geográfica (GIS) e Información Geográfica Voluntaria (VGI). Luego se analizan plataformas existentes con funcionalidades similares a las del proyecto y se presenta una comparación entre ellas.
- **Capítulo 3: Análisis y requerimientos.** Se identifican los actores del sistema y sus responsabilidades, y se detallan los requerimientos funcionales y no funcionales que guiaron el desarrollo. Además se presentan los casos de uso más relevantes de cada actor.
- **Capítulo 4: Propuesta de solución.** Se describe la arquitectura del sistema, el modelo de dominio y los flujos principales de interacción. También se explica el proceso seguido para obtener y preparar los datos geográficos de carga inicial.
- **Capítulo 5: Implementación.** Se detalla el stack tecnológico seleccionado y las decisiones técnicas tomadas durante el desarrollo, incluyendo la organización interna del backend, el diseño de la base de datos, la estructura del frontend, el mecanismo de autenticación y el esquema de despliegue.
- **Capítulo 6: Pruebas y validaciones.** Se describen los casos de prueba ejecutados por módulo, las sesiones de validación realizadas con usuarios externos al proyecto y un resumen de los defectos encontrados y corregidos.
- **Capítulo 7: Gestión del proyecto.** Se presenta la planificación inicial mediante un diagrama de Gantt, las herramientas utilizadas para la coordinación del equipo y el seguimiento del proyecto, y un análisis de las desviaciones ocurridas respecto a lo planificado.
- **Capítulo 8: Conclusiones y trabajo futuro.** Se exponen los resultados obtenidos en relación con los objetivos planteados y se proponen posibles líneas de trabajo para continuar el desarrollo de la plataforma.

Capítulo 2

Revisión de antecedentes

En este capítulo se definen y describen los conceptos fundamentales para el entendimiento del presente documento, así como también se introducen y analizan plataformas existentes con funcionalidades similares a Salud en Cercanía. Esto permite contextualizar la solución propuesta y destacar su aporte diferencial.

2.1. Marco Teórico

En esta sección se presentan los conceptos clave que constituyen la base conceptual del proyecto, entre ellos los datos geográficos, los Sistemas de Información Geográfica (GIS) y la Información Geográfica Voluntaria (VGI).

2.1.1. Datos e Información Geográfica

Los términos "datos geográficos" e "información geográfica" suelen utilizarse indistintamente, aunque poseen significados diferentes. Un dato representa una observación aislada del mundo real, mientras que la información geográfica (IG) se obtiene al estructurar, procesar y combinar esos datos con el fin de reducir la incertidumbre sobre el entorno. [1]

"Los datos constituyen lo que se llama IG cuando se almacenan sistemáticamente, se consultan, explotan, o cuando se combinan produciendo nueva información. La IG se crea, en última instancia, por el usuario."[1]

Las características principales de los datos geográficos son:

- Posición: Ubicación espacial del objeto (coordenadas o relación con otros objetos).
- Atributos temáticos: Propiedades que describen al objeto geográfico.
- Tiempo: Momento en que se observa el hecho geográfico.
- Relaciones espaciales: Conexiones entre distintos objetos geográficos.

- Dimensión: Define la geometría (punto, línea, polígono, etc.). [1]

2.1.2. Sistemas de Información Geográfica (GIS)

Se estima que cerca del 70 % de la información que manejamos tiene algún componente geográfico. Un Sistema de Información Geográfica (GIS, por su sigla en inglés) permite capturar, almacenar, analizar y representar dicha información, integrando tecnología, datos y personas.

"Un Sistema de Información Geográfica es un sistema que integra la tecnología informática, las personas y la información geográfica, y cuya principal función es capturar, analizar, almacenar, editar y representar los datos georreferenciados." [2]

2.1.3. Información Geográfica Voluntaria (VGI)

La Información Geográfica Voluntaria (VGI) se refiere a datos espaciales aportados libremente por usuarios [3], tanto de forma activa (ej. subir manualmente una ubicación) como pasiva (ej. mediante el GPS de un dispositivo móvil). Este tipo de información ha cobrado gran relevancia en áreas como salud, gestión de emergencias y planificación urbana.

"Se conoce como Información Geográfica Voluntaria al uso de Internet para crear, gestionar y difundir información geográfica aportada voluntariamente por los usuarios."[2]

Un ejemplo representativo de VGI es OpenStreetMap (OSM) [4], donde una comunidad global de usuarios colabora ingresando información sobre lugares y servicios.

Si bien la VGI permite una cobertura amplia y actualizada, presenta desafíos como la validación de la información y la homogeneidad de criterios entre los distintos colaboradores. [5]

En el contexto de este proyecto, la VGI permite que ciudadanos y profesionales de la salud aporten datos relevantes sobre infraestructuras sanitarias (ej. ubicación de policlínicas, horarios, accesibilidad, etc.), lo cual es de gran valor para otros usuarios.

2.2. Trabajos Relacionados

En esta sección se analizan plataformas y soluciones existentes con funcionalidades similares a Salud en Cercanía, destacando sus características principales. Posteriormente, se presenta una comparación general que permite ubicar al proyecto dentro del panorama actual.

2.2.1. Healthsites.io

Healthsites.io [6] es un sitio web que surge como respuesta al brote de ébola ocurrido en el 2014 [7], permite visualizar en un mapa de forma global instalaciones de salud (ya sean clínicas, hospitales, centros odontológicos, farmacias y

doctores), esto es posible gracias a que es un proyecto de código abierto donde los usuarios pueden colaborar de forma voluntaria ingresando la ubicación de instalaciones de salud que aún no hayan sido registradas. Además, también colabora con OpenStreetMap (OSM) [4] de forma bidireccional, para poder ingresar la ubicación de un nuevo establecimiento de salud, es necesario contar antes con una cuenta de OSM, ya que al crear, eliminar o actualizar un centro de salud en Healthsites.io, cualquiera de estos cambios se ven reflejados en la base de datos de OSM.

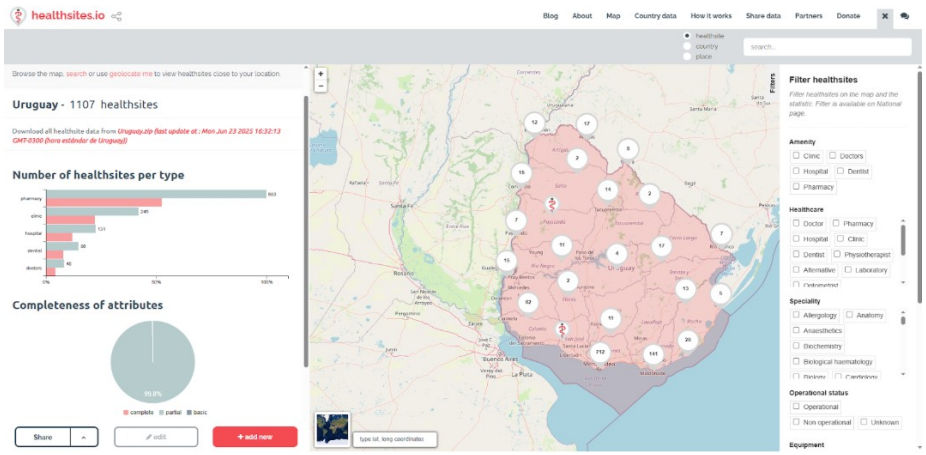


Figura 2.1: Interfaz de healthsites.io, del lado izquierdo se muestran todas las instalaciones de salud que hay en Uruguay, en el centro está el mapa el cual es posible utilizarlo como cualquier mapa en línea, y del lado derecho se puede observar algunos de los distintos filtros para la búsqueda.

En la figura 2.1podemos observar su interfaz, en la parte izquierda se muestran gráficos sobre la cantidad de cada tipo de instalación médica registrada hasta la fecha en Uruguay. En la zona central se encuentra el mapa por el cual el usuario puede navegar libremente, también puede geolicalizarse a sí mismo clickeando en la opción de geolocate me en la parte superior izquierda o puede directamente buscar en la barra superior derecha, dónde la búsqueda se puede realizar ya sea por:

- Nombre de la instalación médica, para este caso a medida de que el usuario va completando el nombre de la misma, en la barra de búsqueda se le muestran automáticamente todas las coincidencias registradas, aunque por ejemplo si el usuario ingresa “casmu”, no es posible especificar la zona de la policlínica y de forma análoga no se sabe que respuesta es la que busca el usuario, como se puede observar en la figura 2.2.

- Nombre del país y nombre de la zona, para este último antes se debe haber seleccionado el nombre del país en el filtro por país.



Figura 2.2: Coincidencias de auto completar la búsqueda de una instalación médica por parte del usuario.

Al hacer clic en un resultado de la búsqueda como se hace en la figura 2.3 al clickear sobre el primer resultado (CASMU), se puede observar que del lado izquierdo se muestra la información detallada del lugar, como lo es su ubicación (definida en texto y también en datos geográficos), su teléfono (en este caso no está definido), su horario de atención que tampoco está definido para este hospital, un link a OSM y algunos datos como lo son el tipo de instalación, que especialistas hay en esa instalación y si posee emergencia. También se muestra quien fue el usuario que ingresó los datos para esa instalación y cuando fue la última vez que se actualizaron.

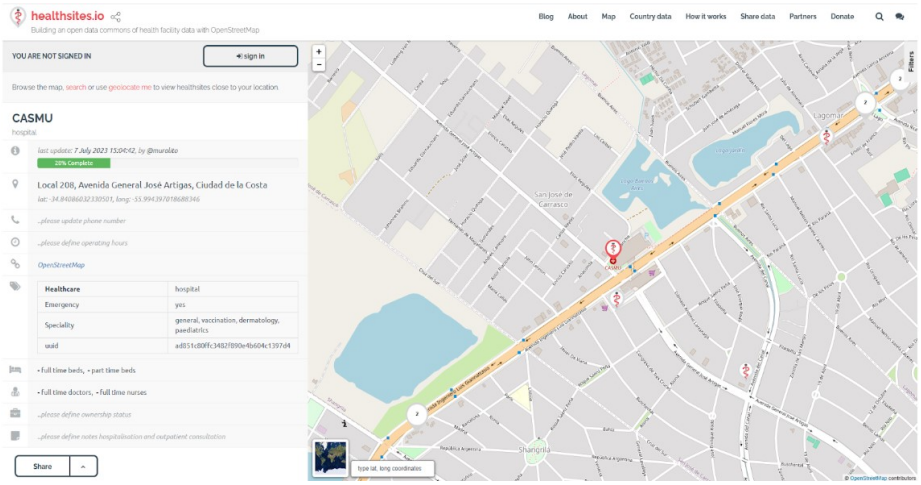


Figura 2.3: Información detallada al hacer clic sobre una instalación médica.

Adicionalmente a la búsqueda se le pueden agregar filtros los cuales se observan en la parte derecha de la figura 2.1, de las cuales se destacan: tipo de instalación, especialidad que se brinda, si cuenta con algún equipamiento específico, si tiene farmacia dentro de la misma instalación y si posee servicio de urgencia, entre otras.

Healthsites.io también cuenta con una API [8] la cual permite incluir fácilmente los datos geográficos de los centro de salud en cualquier proyecto, además permite obtener estos datos en formatos como GeoJSON, Shapefile, KML y CSV.

2.2.2. AtuServicio

ATuServicio [9] es un proyecto realizado por Data Uruguay [10] en colaboración con el Ministerio de Salud Pública [11], el cual permite visualizar y comparar cientos de indicadores de todos los prestadores de salud (públicos y privados) del país, para tomar decisiones informadas a la hora de elegir un prestador de salud, cambiar a otro en caso de poseer uno o simplemente evaluar la calidad que brinda el prestador asociado.

Se puede filtrar por departamento y también por institución como se puede observar en la parte superior de la figura 2.4, en caso de filtrar por departamento, se listan todas las instituciones que se encuentran es ese departamento con su información detallada sobre sus indicadores.

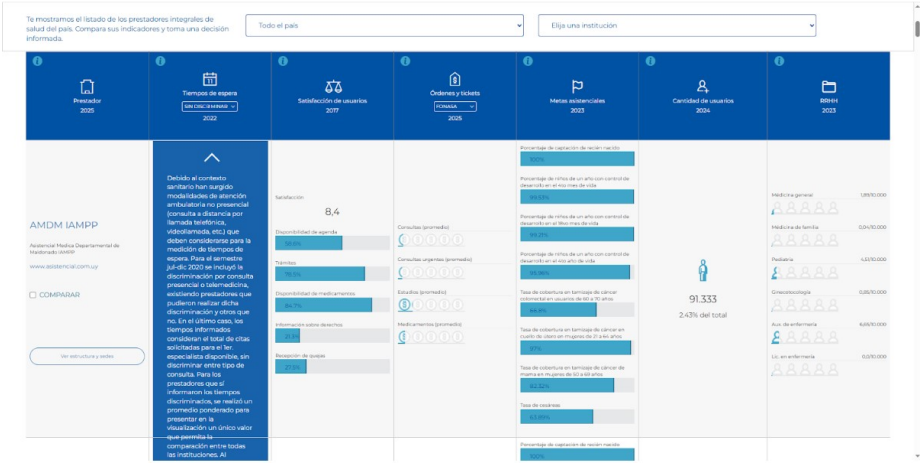


Figura 2.4: Interfaz de AtuServicio, donde se muestran distintos indicadores de los prestadores de salud.

Dentro de estos indicadores se pueden encontrar por ejemplo la cantidad de usuarios totales de la institución, los tiempos de espera promedio, el promedio de precios de consultas/estudios/medicamentos, nivel de satisfacción de los usuarios en diferentes ámbitos, y otros más. También es posible comparar dos o más prestadores a la vez y contrastar la información de uno frente al otro como se muestra en la figura 2.5.

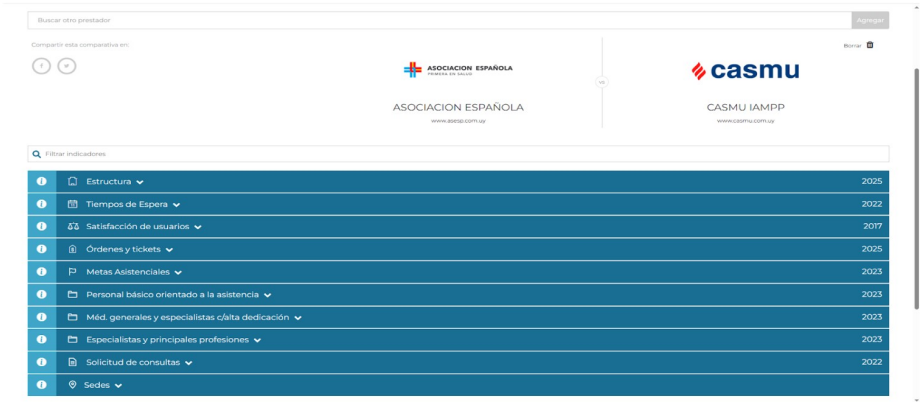


Figura 2.5: Comparación entre los prestadores Casmu y Asociación Española.

AtuServicio utiliza datos abiertos que se encuentran disponibles para su reutilización a través del Catálogo Nacional de Datos Abiertos de AGESIC, estos datos fueron de suma importancia para el desarrollo de Salud en Cercanía.

2.2.3. Doctoralia

Doctoralia es una plataforma que permite conectar a pacientes y profesionales de la salud, está disponible en algunos países de Latinoamérica y del mundo (aunque no está disponible en Uruguay). Los pacientes pueden ingresar en el buscador el nombre de la especialidad que buscan, el nombre de una enfermedad o el nombre de una institución médica, también pueden buscar por una zona específica, y pueden elegir si la consulta es en línea o presencial, como se observa en la 2.6.

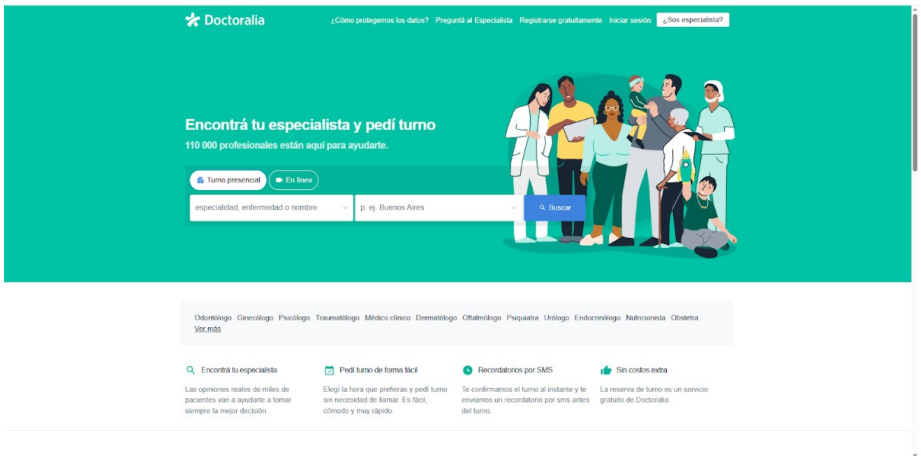


Figura 2.6: Interfaz de Doctoralia, donde se puede buscar por tipo de especialista, nombre de la enfermedad o nombre de la institución, zona en la que se desea buscar y si la consulta es presencial o en línea.

En la figura 2.7 se muestra como se buscó por tipo de especialista “odontólogo” en el barrio Palermo de Buenos Aires. En los resultados se puede observar que a la izquierda se encuentran los odontólogos encontrados para esa zona (también es posible encontrar instalaciones médicas como se mencionó anteriormente para la búsqueda) con su información detallada como por ejemplo su nombre completo, ubicación, precio de consulta (si lo especifica), calificación, cantidad de reseñas que posee y horarios disponibles para la consulta. A la derecha se muestra el mapa de la zona en la cual se realizó la búsqueda, el cual se puede ampliar y los puntos azules en el mapa son las ubicaciones de los especialistas que se muestran del lado izquierdo. Al clickear en la foto o nombre del especialista se puede ver toda la información anterior más información adicional como puede ser donde se graduó el especialista y su experiencia en el rubro, además

de la posibilidad de enviarle un mensaje directo o pedir turno, y la opción de poder agregarlo a tus favoritos. Ver figura 2.8.

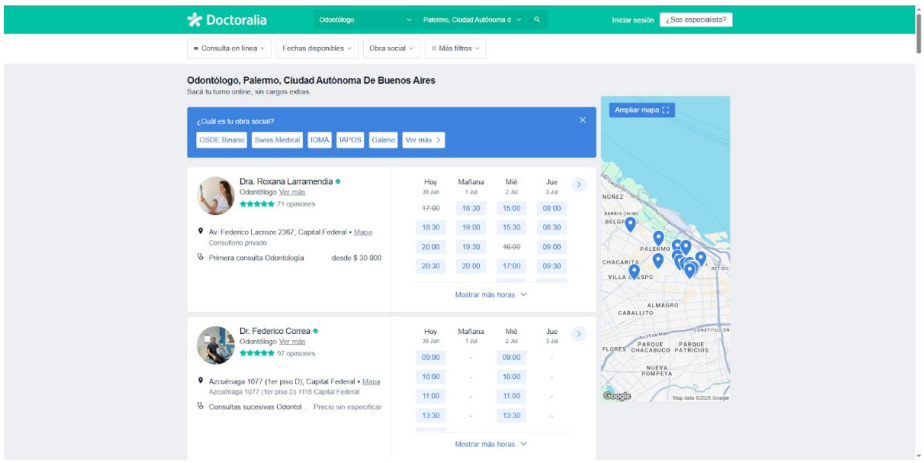


Figura 2.7: Resultados de la búsqueda de Odontólogos en el barrio Palermo de Buenos Aires, Argentina.

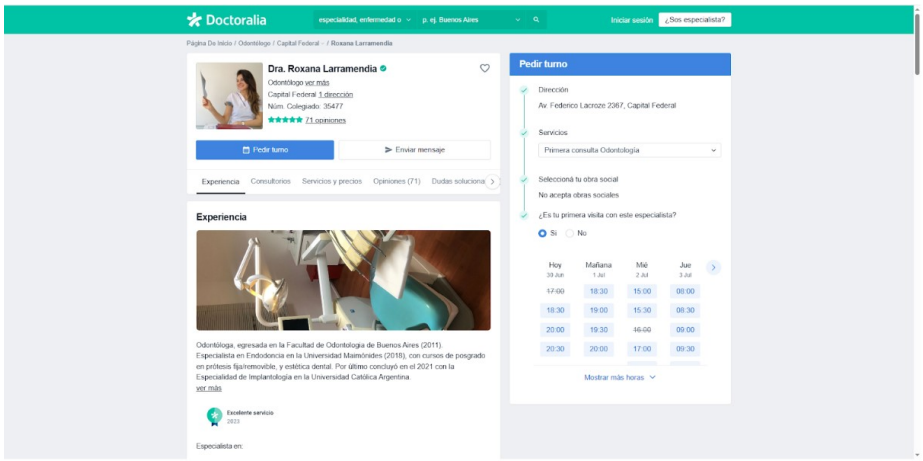


Figura 2.8: Información detallada sobre el especialista donde se puede ver además de los datos de la figura 2.6, donde se graduó el especialista y su experiencia y también es posible agregar el especialista a tus favoritos.

Adicionalmente, a la búsqueda se le pueden agregar varios filtros (figura 2.9), de los cuales podemos destacar el método de pago, si el especialista atiende niños y la accesibilidad para pacientes, es decir si cuenta con acceso para personas con problemas de visión, audición, en silla de ruedas o para mujeres embarazadas.

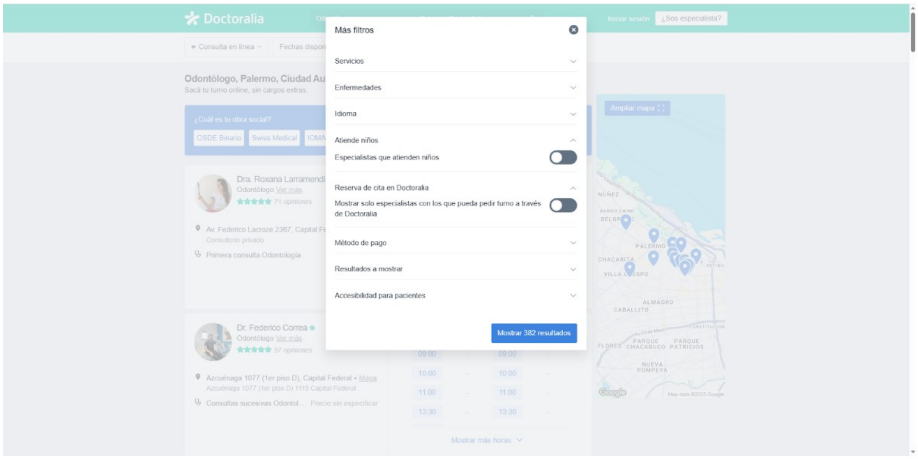


Figura 2.9: Filtros adicionales para la búsqueda realizada en la figura 2.6, entre los cuales se destacan: métodos de pago, si atiende niños y acceso para pacientes con dificultades.

2.2.4. Medicare

Medicare [13] es una plataforma del gobierno de los Estados Unidos que brinda seguro médico para personas mayores de 65 años o personas jóvenes con discapacidades [14]. Permite buscar por ubicación distintas instalaciones médicas o profesionales de la salud. Al igual que Healthsites.io al buscar una ubicación, auto completa la búsqueda con resultados parciales como se puede observar en la figura 2.10. Opcionalmente se puede buscar también por el nombre o el tipo de hospital.

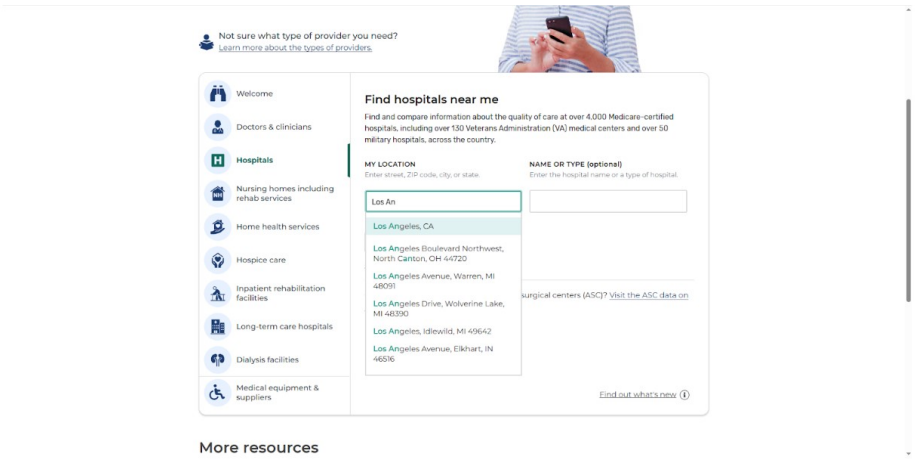


Figura 2.10: Coincidencias de autocompletar la búsqueda por zona del usuario. Similar a la búsqueda en Healthsites.io.

También similar a Doctoralia, en la figura 2.11, se observa que al ingresar una búsqueda por zona (en este caso Los Ángeles, California) y obtener los resultados, del lado izquierdo se muestran los hospitales (porque en la parte superior se seleccionó por hospitales) y del lado derecho el mapa, donde los puntos verdes indican la localización de los hospitales del lado izquierdo. Este sitio agrega como nueva funcionalidad la posibilidad de ordenar por cercanía de la zona seleccionada y también filtrar por distancia dentro de la zona seleccionada.

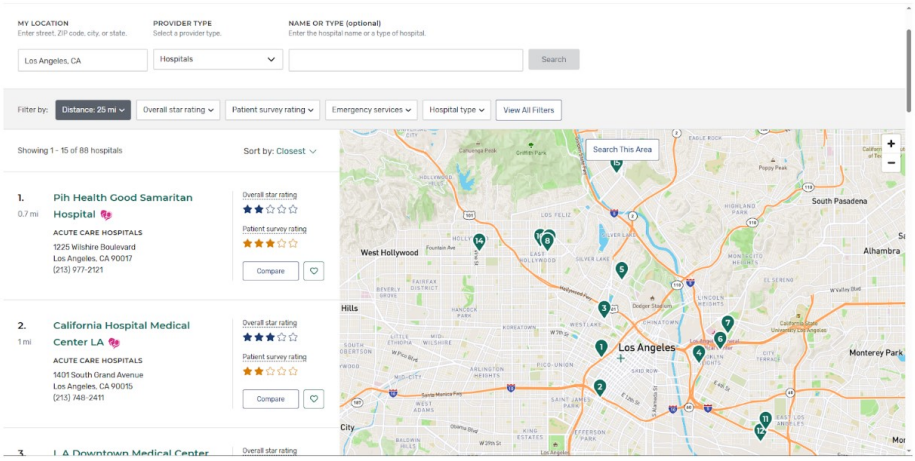


Figura 2.11: Resultados de la búsqueda de hospitales en Los Ángeles, California. Similar a los resultados de la búsqueda en Doctoralia.

Además de mostrar los hospitales con su nombre y dirección, también mues-

tra la calificación total y en lugar de mostrar las opiniones de los usuarios, muestra una calificación basada en una encuesta realizada a los usuarios que se atendieron en ese hospital. De forma adicional es posible añadir el hospital a favoritos y también es posible comparar dos o más hospitales como en AtuServicio. En la figura 2.12 se puede ver como es la comparación de dos hospitales donde se puede contrastar la información detallada de uno frente al otro.

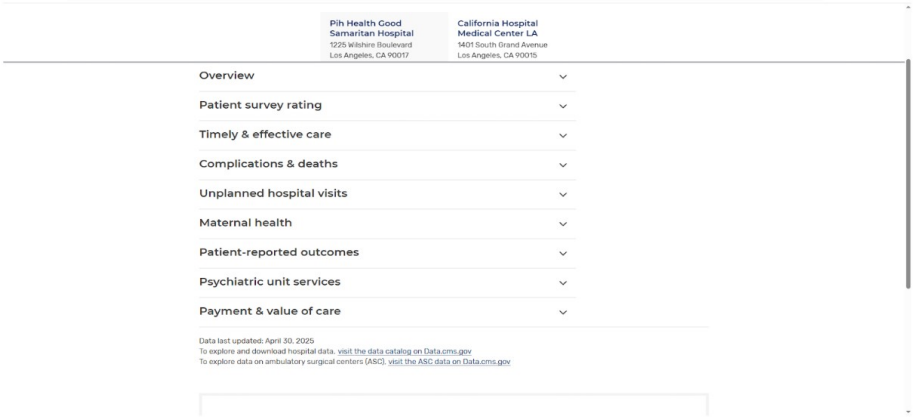


Figura 2.12: Resultado de comparar dos hospitales en Los Ángeles, California. Similar a la comparación que se realiza en AtuServicio.

2.3. Comparación de Plataformas

Se presenta a continuación una tabla comparativa entre las plataformas analizadas (tabla 2.1), destacando aspectos clave como: soporte VGI, visualización geográfica, nivel de detalle, funcionalidades de búsqueda, favoritos, comparación de resultados, filtros avanzados, etc.

Funcionalidad	**Healthsites.io**	**AtuServicio**	**Doctoralia**	**Medicare**
VGI	Sí	No	No	No
Vis. mapa	Sí	No	Sí	Sí
Info. detallada	No	Sí	Sí	Sí
Interfaz amigable	No	Sí	Sí	Sí
Búsq. calle/zona	No	No	Sí	Sí
Búsq. prestador	Sí	Sí	No	No
Búsq. especialidad	No	No	Sí	Sí
Favoritos	No	No	Sí	Sí
Reseñas	No	No	Sí	No
Comparar	No	Sí	No	Sí
Filtrar género	No	No	Sí	Sí

Tabla 2.1: Comparación de funcionalidades entre las plataformas analizadas.

Capítulo 3

Análisis y Requerimientos

En este capítulo se presenta el análisis detallado de los requerimientos del sistema Salud en Cercanía, definido a partir de la revisión de los objetivos del proyecto y del estudio de las necesidades de los distintos actores involucrados. Se identifican los actores principales y sus responsabilidades dentro del sistema, junto con los requerimientos funcionales y no funcionales que la solución debe satisfacer. Asimismo, se describen los casos de uso más relevantes que reflejan los principales flujos de interacción entre los usuarios y la plataforma.

Este análisis permite establecer una comprensión clara y estructurada de las funcionalidades esperadas, asegurando que el desarrollo posterior responda de forma adecuada a las necesidades planteadas y sirva como base sólida para el diseño e implementación del sistema.

3.1. Contexto de trabajo

El sistema está orientado a la ciudadanía en general, sin requerir conocimientos técnicos para su uso. Los usuarios acceden desde un navegador web para consultar policlínicas cercanas a su ubicación o buscar centros de salud en otras zonas del país. Los administradores de prestadores de salud acceden al mismo sistema con un perfil diferenciado, a través del cual gestionan la información de las instituciones a su cargo. El administrador de plataforma dispone de herramientas de moderación y supervisión que no están disponibles para los demás roles.

Esta diferenciación de perfiles determina tanto los flujos de interacción como los requerimientos del sistema, que se detallan en las secciones siguientes.

3.2. Actores

3.2.1. Usuario

Es el actor principal del sistema. Representa a cualquier persona que accede a la plataforma con el objetivo de buscar o compartir información sobre servicios de salud. Dicho usuario puede estar o no previamente registrado en la plataforma dependiendo del caso de uso.

Responsabilidades Principales

- Registrarse e iniciar sesión
- Buscar policlínicas cercanas o en otras zonas
- Consultar especialidades y servicios disponibles
- Leer y escribir reseñas junto a una calificación
- Guardar favoritos
- Sugerir nuevas policlínicas (VGI)
- Sugerir cambios en la información general de alguna policlínica existente (VGI).
- Sugerir dar de baja alguna policlínica que ya no exista y aún siga en el sistema (VGI).

3.2.2. Administrador de Prestador de Salud

Representa al personal autorizado por una institución de salud para gestionar la información de dicha institución dentro de la plataforma.

Responsabilidades Principales

- Registrar nuevas policlínicas de su prestador asignado.
- Actualizar información de alguna policlínica de su prestador asignado.
- Dar de baja policlínicas del prestador asignado que ya no operan.
- Cargar y actualizar datos de servicios y/o especialidades de policlínicas del prestador asignado.
- Validar o rechazar sugerencias de ubicación, de edición y de baja de policlínicas enviadas por usuarios.

3.2.3. Administrador de Plataforma

Es el actor responsable del mantenimiento general de la plataforma, su moderación y la supervisión de la información generada por los usuarios.

Responsabilidades Principales

- Revisar y moderar reseñas y calificaciones.
- Gestionar contenido inapropiado o inconsistente.
- Supervisar la actividad general de usuarios y prestadores.

3.3. Vista de Casos de Uso

En las figuras 3.1, 3.2 y 3.3, se presentan diagramas que relacionan los diferentes actores que interactúan en el sistema con los casos de uso.

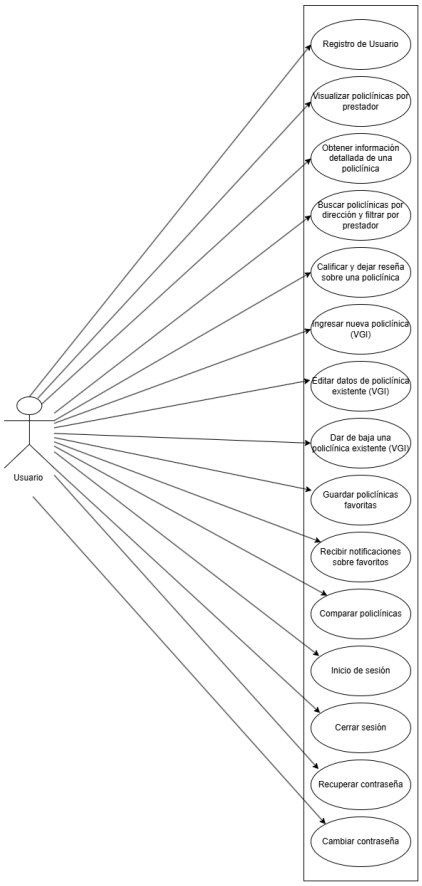


Figura 3.1: Casos de uso: Usuario



Figura 3.2: Casos de uso: Administrador de prestador de salud

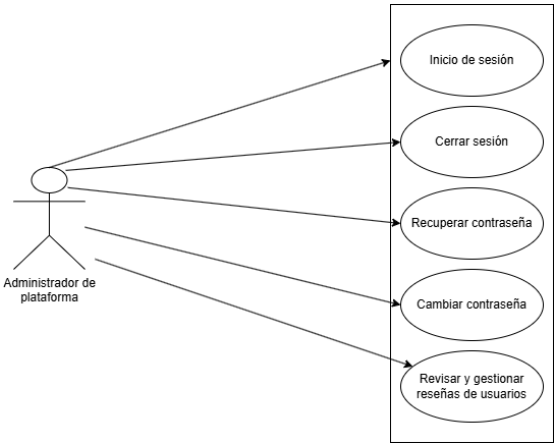


Figura 3.3: Casos de uso: Administrador de plataforma

3.4. Requerimientos Funcionales

A continuación se detallan los requerimientos funcionales del sistema, organizados por área de funcionalidad.

3.4.1. Autenticación y gestión de cuenta

- **RF-01**: El sistema debe permitir el registro de nuevos usuarios mediante correo electrónico y contraseña.
- **RF-02**: El sistema debe permitir el inicio y cierre de sesión, con la opción de mantener la sesión activa durante un período extendido.
- **RF-03**: El sistema debe ofrecer un flujo de recuperación de contraseña mediante el envío de un enlace por correo electrónico con validez limitada en el tiempo.
- **RF-04**: Los usuarios autenticados deben poder modificar su contraseña desde la configuración de su cuenta.

3.4.2. Búsqueda y visualización

- **RF-05**: El sistema debe permitir buscar policlínicas por nombre, por prestador de salud o por zona geográfica (calle o localidad), con tolerancia a errores tipográficos y ausencia de tildes.
- **RF-06**: Los resultados de búsqueda deben mostrarse tanto en un mapa interactivo como en una lista con información resumida.
- **RF-07**: El sistema debe mostrar la información detallada de cada policlínica, incluyendo su ubicación en el mapa, el prestador al que pertenece, las especialidades disponibles, los servicios con sus horarios de atención y las reseñas de otros usuarios.
- **RF-08**: El sistema debe permitir comparar múltiples policlínicas simultáneamente en una vista unificada.

3.4.3. Favoritos, reseñas y notificaciones

- **RF-09**: Los usuarios autenticados deben poder guardar policlínicas como favoritas y gestionar esa lista desde su perfil.
- **RF-10**: Los usuarios autenticados deben poder escribir reseñas sobre policlínicas, incluyendo una calificación numérica del 1 al 5 y un comentario de texto.
- **RF-11**: El sistema debe notificar a los usuarios cuando una policlínica que tienen en favoritos recibe cambios en su información o es dada de baja.
- **RF-12**: Los usuarios deben poder consultar sus notificaciones y marcarlas como leídas.

3.4.4. Contribución de datos

- **RF-13**: Los usuarios autenticados deben poder proponer la incorporación de nuevas policlínicas, especificando su nombre, ubicación en el mapa y prestador de salud asociado.
- **RF-14**: Los usuarios autenticados deben poder proponer la edición de información de una policlínica existente, indicando los datos que consideran incorrectos o desactualizados.
- **RF-15**: Los usuarios autenticados deben poder proponer la baja de una policlínica que ya no esté operativa.
- **RF-16**: Toda propuesta enviada por un usuario debe quedar en estado pendiente hasta que el administrador del prestador correspondiente la revise y apruebe o rechace.

3.4.5. Administración de prestador de salud

- **RF-17**: El administrador de prestador de salud debe poder registrar, editar y dar de baja policlínicas asociadas a su institución.
- **RF-18**: El administrador de prestador de salud debe poder gestionar las especialidades y servicios disponibles en cada policlínica, incluyendo sus horarios de atención.
- **RF-19**: El administrador de prestador de salud debe poder revisar, aprobar o rechazar las propuestas de incorporación, edición y baja de policlínicas enviadas por los usuarios.

3.4.6. Administración de plataforma

- **RF-20**: El administrador de plataforma debe poder revisar y moderar las reseñas enviadas por los usuarios antes de su publicación.
- **RF-21**: El administrador de plataforma debe poder gestionar los usuarios y prestadores de salud registrados en el sistema.

3.5. Requerimientos No Funcionales

3.5.1. Seguridad

- **RNF-01**: Las contraseñas de los usuarios deben almacenarse usando un algoritmo de hash seguro; el texto plano no debe persistir en ningún momento.
- **RNF-02**: La autenticación debe basarse en tokens JWT almacenados en cookies `HttpOnly`, de forma que no sean accesibles desde JavaScript y queden protegidos frente a ataques XSS.

- **RNF-03**: La base de datos no debe ser accesible desde el exterior de la red interna del sistema; el acceso debe estar restringido exclusivamente al backend.
- **RNF-04**: Los tokens de recuperación de contraseña deben tener una validez máxima de 15 minutos y solo pueden utilizarse una vez.

3.5.2. Usabilidad

- **RNF-05**: La interfaz debe ser funcional en los navegadores web, sin requerir la instalación de extensiones o complementos adicionales.
- **RNF-06**: Las búsquedas textuales deben ser tolerantes a variaciones en la escritura, incluyendo ausencia de tildes y errores tipográficos menores.

3.5.3. Portabilidad

- **RNF-07**: El entorno del backend y la base de datos debe poder iniciarse en cualquier máquina con `Docker`[1] instalado, sin requerir configuración manual adicional más allá de un archivo de variables de entorno.
- **RNF-08**: La configuración sensible del sistema —clave secreta, credenciales de base de datos y parámetros de correo— debe gestionarse exclusivamente mediante variables de entorno, sin quedar registrada en el repositorio de código.

3.5.4. Mantenibilidad y extensibilidad

- **RNF-09**: La arquitectura en capas del backend debe garantizar que los cambios en la lógica de negocio no requieran modificar la capa de presentación, y viceversa.
- **RNF-10**: La API REST debe estar diseñada de forma que permita integrar nuevos tipos de clientes, como aplicaciones móviles, sin necesidad de modificar el backend.

[1] https://www.docker.com/

Capítulo 4

Propuesta de solución

En este capítulo se describe el diseño de la solución propuesta para Salud en Cercanía. Se presenta la arquitectura del sistema y el modelo de dominio que guió las decisiones de diseño, así como el análisis de las fuentes de datos externas utilizadas para la carga inicial de la plataforma.

4.1. Arquitectura General del Sistema

El sistema Salud en Cercanía se diseñó siguiendo una arquitectura en capas estricta, garantizando una separación clara de responsabilidades, modularidad y facilidad de mantenimiento. A continuación, se presentan las vistas y subsistemas principales que componen la arquitectura.

4.1.1. Vista lógica

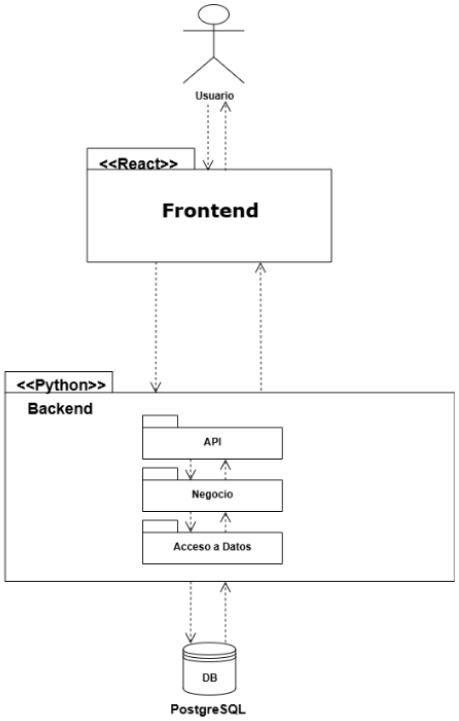


Figura 4.1: Vista Lógica

El sistema se compone de un *backend* desarrollado en `Python`[1] utilizando `FastAPI`[2], que implementa una API REST. El backend expone servicios para ser consumidos por el *frontend*, gestiona la lógica de negocio que se encarga de gestionar las peticiones del frontend y comunicarse con la base de datos.

Siguiendo una arquitectura en capas, el backend se estructura lógicamente en los siguientes componentes:

- **Capa de presentación (API/Controladores):** Define los endpoints HTTP expuestos al frontend. (por ejemplo, `/api/usuarios`, `/api/policlinicas`). Recibe solicitudes, valida datos de entrada y retorna respuestas.
- **Capa de lógica de negocio (Servicios):** Contiene las reglas de negocio del sistema. Procesa solicitudes de la capa de presentación y coordina operaciones con la capa de datos.
- **Capa de acceso a datos:** Gestiona la comunicación con la base de datos mediante consultas SQL parametrizadas, desacoplando la lógica de negocio del almacenamiento.

Cada capa se encuentra encapsulada y expone interfaces claras que facilitan la mantenibilidad y reutilización del sistema.

4.1.2. Estilo arquitectónico

El diseño adoptado es un **modelo por capas estricto**, donde el backend actúa como mediador único entre el frontend y la base de datos. Este estilo aporta tres beneficios clave: la **seguridad**, dado que todos los accesos a la base de datos se realizan a través del backend y el usuario nunca interactúa directamente con ella. La **mantenibilidad**, en un sistema por capas estricto, el nivel de modularidad característico de estos sistemas facilita enormemente las tareas de mantenimiento y ampliación, facilitando posibles mejoras o modificaciones a futuro. Y por último la **extensibilidad**, pues la API REST puede ser consumida por otros clientes en el futuro, como aplicaciones móviles, sin cambios en el backend.

4.1.3. Subsistemas

Visto desde el sistema completo, la arquitectura contempla cinco componentes:

- **Frontend:** Interfaz gráfica para el usuario final. Captura las acciones del usuario y las envía al backend mediante solicitudes HTTP. No contiene lógica de negocio.

[1] https://www.python.org/
[2] https://fastapi.tiangolo.com/

- **Capa de API (controladores):** Puerta de entrada al backend. Valida solicitudes, gestiona la autenticación y autorización, y deriva las peticiones a la lógica de negocio.
- **Capa de lógica de negocio:** Núcleo del backend; procesa reglas de negocio, coordina flujos de datos entre entidades y valida permisos según el rol del usuario.
- **Capa de acceso a datos:** Gestiona operaciones sobre la base de datos, ocultando los detalles del motor de almacenamiento al resto del sistema.
- **Base de datos:** Almacenamiento persistente de la información, implementada en `PostgreSQL`[3] con la extensión `PostGIS`[4] para soporte geoespacial.

4.1.4. Vista de distribución

El sistema se despliega bajo un esquema *three-tier*, con separación en tres capas: frontend, backend y base de datos.

- **Capa de presentación (Frontend):** Servidor independiente que sirve los archivos estáticos de la aplicación Next.js[5], no está contenido en `Docker`. Se comunica exclusivamente con el backend mediante solicitudes HTTP.
- **Capa de lógica de negocio (Backend):** Desplegado en un contenedor `Docker`. Gestiona validaciones, autenticación, reglas de negocio y expone la API REST.
- **Capa de datos (Base de datos):** Ejecutada en un contenedor `Docker` independiente, protegida por la red interna de Docker. El puerto de la base de datos no se expone al exterior.

`Docker Compose` se emplea para gestionar los contenedores de backend y base de datos, configurando la red interna y los volúmenes persistentes necesarios.

4.1.5. Escalabilidad y seguridad

El uso de contenedores facilita escalar el sistema de forma horizontal: es posible levantar instancias adicionales del backend sin modificar la base de datos ni el frontend. Desde el punto de vista de la seguridad, cada capa está aislada en su propio entorno y el backend y la base de datos no exponen puertos públicos, lo que reduce la superficie de ataque. El frontend es el único componente accesible desde el exterior, y toda la comunicación entre él y el backend se realiza a través de la API REST.

[3] https://www.postgresql.org/
[4] https://postgis.net/
[5] https://nextjs.org/

4.1.6. Modelo de Dominio

El modelo de dominio representa las entidades principales del sistema y las relaciones entre ellas, sirviendo como base conceptual para el diseño de la arquitectura y la implementación. Este modelo permite identificar de forma clara qué información se gestiona, cómo se organiza y de qué manera interactúan los distintos elementos dentro de la plataforma.

En el modelo se destacan las siguientes entidades:

- **Usuario**: contiene los datos básicos de cada persona registrada en la plataforma, incluyendo credenciales de acceso, correo electrónico y el tipo de usuario (normal, administrador de prestador o administrador de plataforma).
- **Prestador de Salud**: representa a las instituciones de salud registradas en la plataforma, junto con sus datos de contacto y tipo (público, privado, etc.).
- **Policlinica**: está asociada a un prestador de salud y contiene información básica como nombre y médicos disponibles.
- **Servicio**: describe los servicios ofrecidos por una policlínica, incluyendo su descripción y horarios de atención.
- **Especialidad**: define las diferentes áreas médicas (por ejemplo, pediatría, ginecología, odontología) de la policlínica.
- **ResetPassword**: esta clase se utiliza para la recuperación de contraseña del usuario, en caso de haberla olvidado.
- **NuevoPrestador**: contiene los pedidos de agregar nuevos prestadores al sistema, dependiendo de quien realice el pedido, se aprueba automáticamente (administrador de prestador) o el pedido queda en estado de espera (un usuario)
- **EditarPrestador**: contiene los pedidos de editar la información de un prestador existente en el sistema, dependiendo de quien realice el pedido, se aprueba automáticamente (administrador de prestador) o el pedido queda en estado de espera (un usuario)
- **EliminarPrestador**: contiene los pedidos de eliminar prestadores que fueron dados de baja y aún existen en el sistema, dependiendo de quien realice el pedido, se aprueba automáticamente (administrador de prestador) o el pedido queda en estado de espera (un usuario)
- **NuevaEspecialidad**:contiene los pedidos de agregar nuevas especialidades para una policlínica perteneciente al sistema, dependiendo de quien realice el pedido, se aprueba automáticamente (administrador de prestador) o el pedido queda en estado de espera (un usuario)

- **EditarEspecialidad**: contiene los pedidos de editar la información de una especialidad existente para una policlínica en el sistema, dependiendo de quien realice el pedido, se aprueba automáticamente (administrador de prestador) o el pedido queda en estado de espera (un usuario)
- **EliminarEspecialidad**: contiene los pedidos de eliminar especialidades que ya no existen dentro de una policlínica y aún existen en el sistema, dependiendo de quien realice el pedido, se aprueba automáticamente (administrador de prestador) o el pedido queda en estado de espera (un usuario)
- **NuevaPoliclinica**: contiene los pedidos de agregar nuevas policlínicas al sistema, dependiendo de quien realice el pedido, se aprueba automáticamente (administrador de prestador) o el pedido queda en estado de espera (un usuario)
- **EditarPoliclinica**: contiene los pedidos de editar la información de una policlínica existente en el sistema, dependiendo de quien realice el pedido, se aprueba automáticamente (administrador de prestador) o el pedido queda en estado de espera (un usuario)
- **EliminarPoliclinica**: contiene los pedidos de eliminar policliínicas que ya no existen y aún existen en el sistema, dependiendo de quien realice el pedido, se aprueba automáticamente (administrador de prestador) o el pedido queda en estado de espera (un usuario)
- **Favorito**: vincula a un usuario con una policlínica que desea guardar para acceder rápidamente en el futuro.
- **Reseña**: permite a los usuarios valorar y comentar sobre policlínicas, incluyendo una puntuación, texto y fecha de la reseña.
- **Notificacion**: gestiona los mensajes que recibe un usuario sobre el estado de sus pedidos de sugerencias (si fueron aprobados o rechazados), además también se notifica si se da de alta una policlínica de un prestador el cual el usuario ya tiene marcado como favorito alguna de sus policlínicas.
- **Ubicacion**: contiene la dirección y coordenadas geográficas de las policlínicas, permitiendo su representación en el mapa.

Además, se incluyen enumeraciones que estructuran ciertos atributos del sistema:

- `TipoUsuario`: clasifica a los usuarios en normales, administradores de prestador y administradores de la plataforma.
- `TipoPrestador`: identifica a los prestadores como públicos, privados u otros.

El siguiente diagrama muestra gráficamente el modelo de dominio propuesto y las relaciones entre sus entidades:

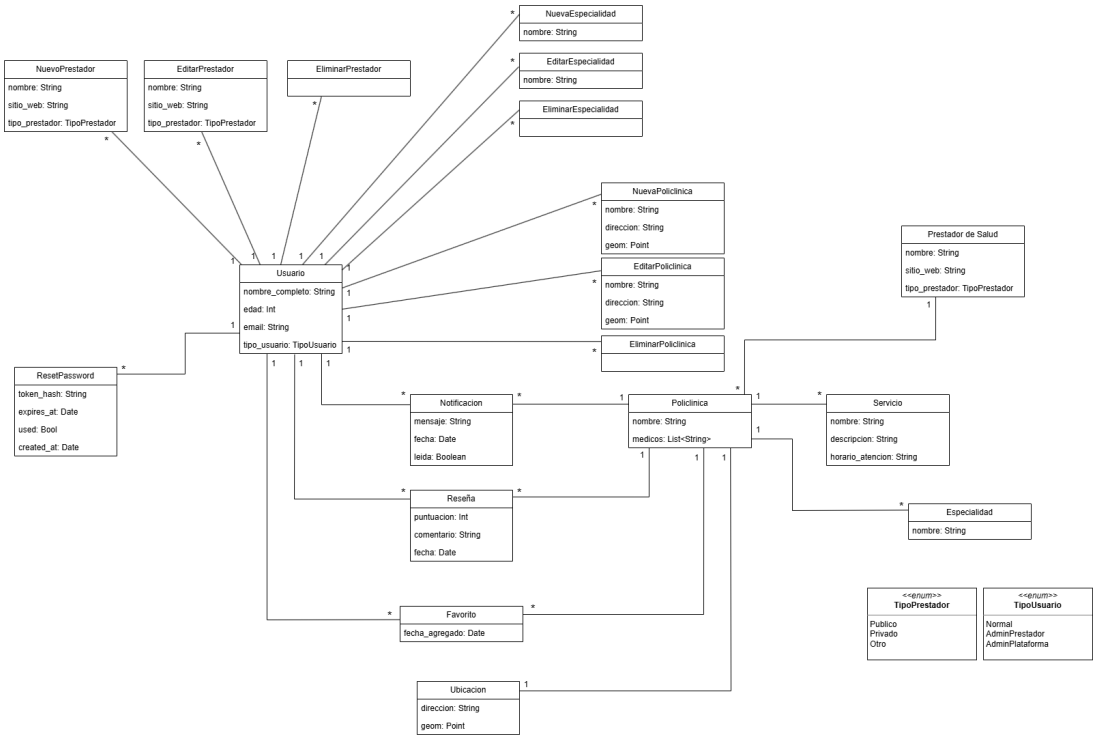


Figura 4.2: Modelo de dominio

4.1.7. Flujos principales del sistema

A continuación se describen dos flujos de interacción que involucran múltiples componentes del sistema y son representativos de su funcionamiento general.

Flujo de contribución VGI: propuesta de nueva policlínica. Cuando un usuario autenticado desea proponer la incorporación de una nueva policlínica, accede al formulario de propuesta, selecciona el prestador de salud al que pertenece, ingresa el nombre de la policlínica, su dirección y marca su ubicación en el mapa interactivo. Al enviar el formulario, el frontend realiza una petición al backend con esos datos, que los almacena en la tabla `Nueva_Policlinica` a la espera de ser aprobados o rechazados por el Administrador de Prestador correspondiente.

El Administrador visualiza la propuesta en su panel de gestión y desde allí puede revisarla y decidir si aprobarla o rechazarla. Si la rechaza, el estado de la propuesta se actualiza y el usuario proponente recibe una notificación con el resultado. Si la aprueba, el servicio del backend registra la policlínica en la tabla principal `Policlinica`, crea su ubicación en `Ubicacion_Policlinica` y consulta qué usuarios tienen como favorita alguna policlínica del mismo prestador. A cada uno de esos usuarios se le genera una `Notificacion` avisando de la apertura del nuevo centro, además de notificarle al usuario que realizo el pedido que el mismo fue aprobado. Este último paso ocurre en la misma transacción de aprobación, coordinado desde la capa de servicios del backend.

En la figura 4.3 se detalla el diagrama de secuencia del sistema correspondiente al flujo descrito.

Flujo de recuperación de contraseña. El usuario accede a la página de recuperación e ingresa su correo electrónico. El backend verifica internamente si existe una cuenta asociada a ese correo y, en caso afirmativo, genera el token correspondiente. En cualquier caso, la respuesta al frontend es idéntica, evitando así que un atacante pueda determinar si un correo está registrado en el sistema. En lugar de almacenarlo en texto plano, se guarda su hash calculado con `SHA-256` en la tabla `PasswordReset`, junto con la fecha y hora de expiración (15 minutos desde la solicitud). A continuación se envía un correo al usuario con un enlace que contiene el token original.

Cuando el usuario accede al enlace e ingresa una nueva contraseña, el backend recalcula el hash del token recibido y lo compara con el almacenado. Si coinciden y el registro no ha expirado ni fue utilizado previamente, la contraseña del usuario se actualiza con el nuevo valor hasheado y la entrada en `PasswordReset` se marca como usada. Este esquema garantiza que una eventual filtración de la base de datos no exponga los tokens de recuperación en texto plano.

En la figura 4.4 se detalla el diagrama de secuencia del sistema del flujo de recuperación de contraseña.

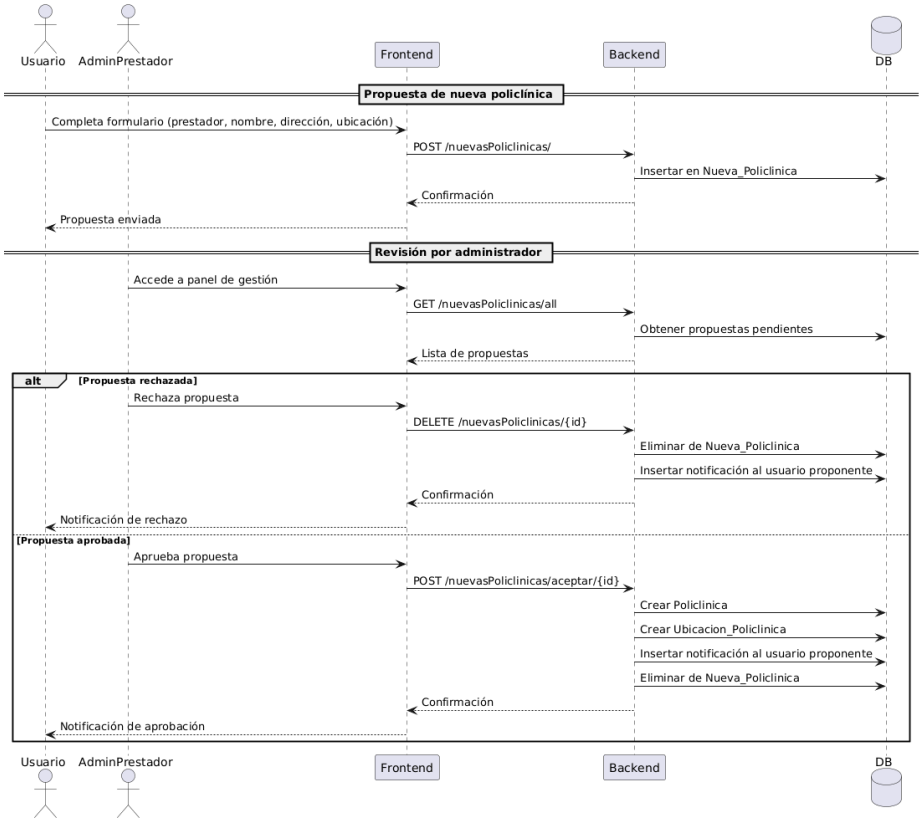


Figura 4.3: Propuesta de nueva policlínica

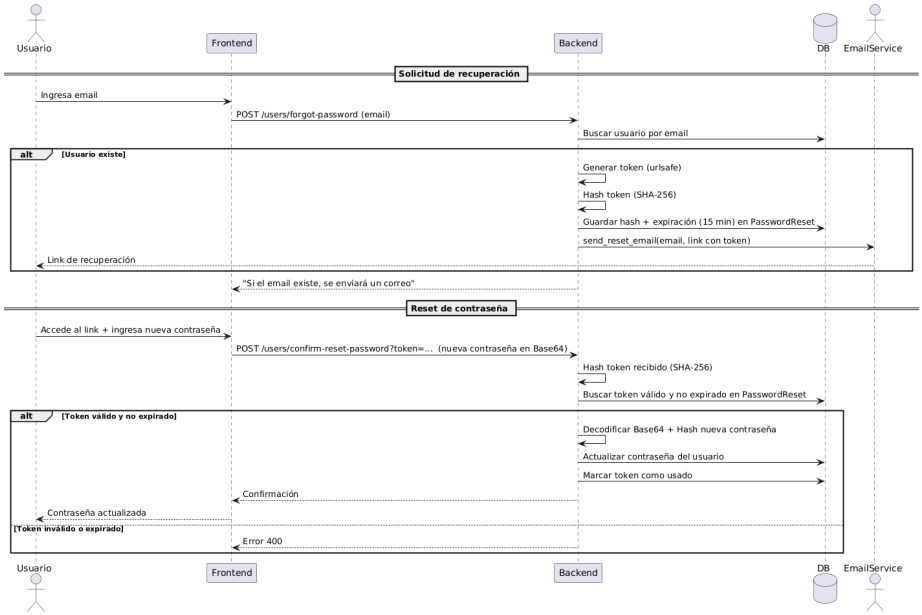


Figura 4.4: Recuperación de contraseña

4.2. Análisis de Datos Externos

En esta sección se explica cómo fue el procedimiento llevado a cabo para obtener los datos utilizados en la carga inicial del sistema.

Inicialmente, se consideraron diferentes alternativas para obtener los datos.

4.2.1. Healthsites.io

Una de las primeras alternativas consideradas fue obtener los datos del sitio web *healthsites.io*, el cual fue presentado anteriormente en el capítulo 2. A través de su API es posible filtrar por país y obtener todas las instituciones médicas de Uruguay.

Luego de obtener los datos —en este caso en formato *shapefile*— y realizar un filtrado considerando únicamente los tipos de institución de interés (clínicas y hospitales, descartando farmacias y centros odontológicos), se obtuvieron los datos correspondientes a dichas instituciones.

Tras analizarlos, se observó que la cantidad de resultados obtenidos no era muy grande (menos de 150 instituciones). Además, no se contaba con información sobre los servicios y especialidades de las mismas, un dato que resulta muy importante para el proyecto. En los datos obtenidos únicamente se detallaba si la institución contaba con servicio de emergencia.

Por lo tanto, a pesar de que los datos ya se encontraban georreferenciados —lo cual representaba una ventaja para Salud en Cercanía—, al considerar los pros y los contras se decidió descartar esta alternativa y continuar con la investigación.

4.2.2. A Tu Servicio

Otra alternativa considerada surgió a partir del sitio web *A Tu Servicio*. En este caso se analizaron dos posibles enfoques.

El primero consistió en aplicar la técnica de *web scraping* sobre el sitio con el objetivo de obtener los datos de los prestadores de salud. Los datos obtenidos incluían el nombre del prestador, su sitio web y la dirección de sus policlínicas. A diferencia de los datos obtenidos a partir de Healthsites, la cantidad de instituciones en este caso era considerablemente mayor (cerca de 1200 policlínicas). Sin embargo, estos datos tampoco incluían información sobre los servicios y especialidades de las instituciones. Además, en este caso las direcciones no se encontraban georreferenciadas, por lo que esta alternativa también fue descartada.

Finalmente, se analizó una tercera opción, también relacionada con *A Tu Servicio*, correspondiente a los datos utilizados por el propio sitio. Estos se encuentran disponibles para su descarga como datos abiertos en la página de Datos Abiertos del Gobierno. En este caso, la cantidad de datos obtenidos fue la mayor (exactamente 1325 instituciones) y, a diferencia de los datos obtenidos mediante *web scraping* del sitio *A Tu Servicio*, aquí sí se encontraba disponible la información sobre los servicios y especialidades de cada institución.

No obstante, al analizar estos datos también surgieron algunos inconvenientes. Uno de ellos fue que los datos descargados contenían errores ortográficos, tanto en las direcciones como en los departamentos. Sin embargo, estos errores resultaron relativamente sencillos de corregir, ya que presentaban patrones similares en la mayoría de los casos, lo que permitió solucionarlos de forma programática.

El inconveniente más importante estuvo relacionado con el proceso de geocodificación de los datos, es decir, la conversión de las direcciones textuales en coordenadas geográficas, ya que en este caso tampoco se contaba con las direcciones georreferenciadas.

Se intentó resolver este problema de forma automatizada, pero no fue posible debido a que muchas direcciones se encontraban incompletas. Por ejemplo, en algunos casos únicamente se disponía del nombre de la avenida en la que se encontraba la institución, sin contar con el número de puerta ni con la referencia de alguna intersección que permitiera obtener una localización más precisa. Asimismo, para varias instituciones ubicadas en el interior del país, las direcciones consistían únicamente en el número de ruta y el kilómetro correspondiente, lo cual en la mayoría de los casos no resultaba posible de geocodificar de forma automática.

Por lo tanto, finalmente fue necesario realizar el proceso de geocodificación de forma manual, lo que introduce un nuevo factor de error asociado al error humano.

Una vez realizado este proceso manual, que consistía en ingresar la dirección textual de cada institución en Google Maps —considerado uno de los catálogos de direcciones más completos y detallados del mundo— y obtener sus coordenadas geográficas correspondientes, se logró alcanzar el resultado deseado.

Antes de concluir esta sección, cabe aclarar que los datos de los prestadores de salud obtenidos del catálogo de datos abiertos del gobierno fueron recolectados en el año 2016. Por lo tanto, durante el proceso de geocodificación no siempre fue posible encontrar las policlínicas buscadas en Google Maps.

De las 1325 instituciones médicas obtenidas a partir de los datos abiertos, únicamente fue posible georreferenciar aproximadamente 800 instituciones.

En relación con estos resultados, es necesario considerar tanto el posible error humano como el margen de error asociado a Google Maps. Cuando no se obtenía un resultado al buscar una dirección, no era posible determinar con certeza la causa. Esto podía deberse a que los datos son antiguos y la institución haya cerrado (como fue el caso de Casa de Galicia, la cual figuraba en los datos pero cerró sus puertas en el año 2021), a que la dirección no fuera lo suficientemente precisa —situación que pudo ocurrir en muchos casos donde únicamente se disponía de la ruta y el kilómetro, o donde solo se conocía la avenida en la que se encontraba la institución—, o a que los nombres de las policlínicas no coincidieran exactamente con los registrados en los datos. En estos últimos casos fue necesario tomar decisiones respecto a su inclusión; sin embargo, cuando la dirección coincidía pero el nombre difería, la institución se incorporaba igualmente al resultado. Otra posibilidad, aunque poco probable, es que la institución no estuviera registrada en la base de datos de Google Maps.

Una situación similar ocurre con los servicios y especialidades de las instituciones. Al tratarse de datos correspondientes al año 2016, es posible que algunas instituciones hayan incorporado nuevos servicios o especialidades con el paso del tiempo o, por el contrario, hayan dejado de ofrecer algunos de ellos.

A pesar de estas limitaciones, dado que Salud en Cercanía es un proyecto basado en la recolección de datos por parte de los ciudadanos —es decir, un proyecto VGI—, se espera que los datos de carga inicial, en caso de contener errores, puedan ser corregidos progresivamente por los usuarios del sitio.

Capítulo 5

Implementación

Este capítulo describe el proceso de implementación de Salud en Cercanía, detallando las tecnologías seleccionadas, la organización interna del sistema y las decisiones técnicas más relevantes tomadas durante el desarrollo.

5.1. Tecnologías Utilizadas

La elección del stack tecnológico estuvo orientada por criterios de productividad, integración entre herramientas y familiaridad del equipo.

Para el **backend** se utilizó `Python 3.10` junto con el framework `FastAPI`, que permite definir endpoints REST con validación automática de datos a través de modelos Pydantic[1]. La manipulación de los resultados de la base de datos se realizó con `pandas` y `geopandas`, lo que facilita el trabajo con datos geoespaciales en formato tabular sin necesidad de un ORM tradicional.

El **frontend** fue desarrollado con `React`[2] y `Next.js` (versión 15), utilizando el sistema de rutas basado en el directorio de la aplicación (*App Router*). Se optó por `TypeScript`[3] para contar con tipado estático, lo que resultó conveniente para mantener la coherencia entre los modelos del backend y los tipos del frontend. Para los estilos se utilizó `Tailwind CSS`[4] junto con la biblioteca de componentes `Shadcn/ui`, basada en Radix UI, que provee componentes accesibles y altamente personalizables.

Para la **visualización geográfica** se integró `Leaflet.js`[5] con tiles de *OpenStreetMap*, lo que permitió desplegar un mapa interactivo sin depender de servicios de mapas de pago.

La **base de datos** es `PostgreSQL` con la extensión `PostGIS`, que agrega soporte nativo para tipos de datos y funciones geoespaciales, resultando de gran utilidad para implementar las búsquedas por proximidad de forma eficiente.

[1] https://docs.pydantic.dev/latest/
[2] https://es.react.dev/
[3] https://www.typescriptlang.org/
[4] https://tailwindcss.com/
[5] https://leafletjs.com/

El sistema se despliega mediante `Docker` y `Docker Compose`, lo que permite levantar el entorno completo con un solo comando, independientemente del sistema operativo. El control de versiones se gestionó con `Git` sobre un repositorio remoto en `GitLab`.

5.2. Arquitectura del sistema

La implementación sigue la arquitectura definida en el capítulo anterior, organizada en tres grandes bloques:

- **Frontend**: aplicación web encargada de la interfaz gráfica y la interacción con el usuario. Se comunica exclusivamente con el backend mediante peticiones HTTP a la API REST.
- **Backend**: implementado en `Python` utilizando `FastAPI`. Concentra la lógica de negocio, la gestión de la seguridad, la validación de datos y la exposición de los endpoints. Se ejecuta dentro de un contenedor `Docker`.
- **Base de datos**: alojada en un contenedor `Docker` separado, aísla el almacenamiento del resto del sistema. El acceso se realiza únicamente a través del backend, reforzando el encapsulamiento y la seguridad.

`Docker Compose` gestiona los contenedores, configurando redes internas y volúmenes persistentes para garantizar la integridad de los datos y la comunicación segura entre backend y base de datos.

5.3. Backend

5.3.1. Estructura del proyecto

El código del backend se organiza siguiendo el patrón de capas descrito en la propuesta de solución:

```
Backend/src/
|-- api/            # Rutas y endpoints HTTP
|-- core/           # Modulos de dominio
|   |-- usuarios/
|   |-- policlinicas/
|   |-- prestadores/
|   |-- especialidades/
|   |-- servicios/
|   |-- favoritos/
|   |-- notificaciones/
|   |-- resenas/
|   |-- security/
|   +-- db/         # Acceso a base de datos y queries SQL
```

```
|-- dependencies/   # Middleware de autenticacion y autorizacion
+-- utils/          # Utilidades compartidas
```

Cada módulo dentro de `core/` sigue la misma estructura interna: un archivo de modelos Pydantic, un archivo de servicios con la lógica de negocio, otro con las queries SQL y archivos JSON con los mensajes de error e información. Esto hace que el código de cada dominio quede autocontenido y que incorporar un nuevo módulo sea predecible.

5.3.2. Capa de API

Los endpoints HTTP se definen usando routers de FastAPI. Los principales grupos de endpoints son:

- `/api/auth`: login y logout.
- `/api/users`: registro, consulta y gestión de usuarios, incluyendo cambio y recuperación de contraseña.
- `/api/policlinicas`: consulta, creación, edición y eliminación de policlínicas, con soporte de búsqueda geográfica por proximidad.
- `/api/prestadores`: gestión de prestadores de salud.
- `/api/especialidades` y `/api/servicios`: especialidades y servicios asociados a cada policlínica.
- `/api/favoritos`, `/api/notificaciones` y `/api/resenas`: funcionalidades propias del usuario autenticado.
- Endpoints de propuestas y aprobaciones (`/api/nuevasPoliclinicas`, `/api/editarPoliclinicas`, `/api/eliminarPoliclinicas` y sus equivalentes para prestadores y especialidades): gestionan el flujo de moderación de cambios en la plataforma.

Cada endpoint declara explícitamente sus dependencias de autenticación y autorización, manteniéndolas desacopladas de la lógica propia del endpoint.

5.3.3. Capa de servicios

La capa de servicios concentra la lógica de negocio del sistema. Como ejemplo, cuando se aprueba la incorporación de una nueva policlínica, el servicio correspondiente no solo registra el dato en la base de datos sino que también consulta que usuarios tienen como favorito alguna policlínica asociada al prestador del cual se dió de alta esa nueva policlínica, y les envía una notificación avisándoles de la creación de la nueva policlínica. Este tipo de coordinación entre distintas entidades es responsabilidad de los servicios, y no de los controladores ni de las queries.

5.3.4. Acceso a datos

Para el acceso a la base de datos se optó por queries SQL escritas directamente, sin utilizar un ORM en modo objeto-relacional. Las consultas se ejecutan a través de un `DatabaseManager` centralizado que usa `SQLAlchemy Core` para gestionar la conexión y `pandas` para convertir los resultados en DataFrames, facilitando su manipulación antes de transformarlos en modelos Pydantic para la respuesta.

Todas las queries utilizan parámetros nombrados en lugar de interpolación de strings, lo que previene inyecciones SQL por construcción.

5.4. Base de Datos

5.4.1. Motor y extensiones

Se utilizó `PostgreSQL 17` junto con tres extensiones:

- `PostGIS`: agrega el tipo de dato `GEOMETRY` y funciones como `ST_Distance` y `ST_DWithin`, necesarias para las búsquedas de policlínicas por proximidad geográfica.
- `pg_trgm`: habilita búsquedas por similitud de texto usando trigramas, lo que hace más tolerante la búsqueda por nombre de especialidad ante variaciones en la escritura o errores tipográficos.
- `unaccent`: normaliza texto eliminando tildes, mejorando la búsqueda cuando los usuarios no las incluyen.

5.4.2. Diseño del esquema

El esquema de la base de datos se inicializa a partir del archivo `init.sql`, que es inyectado en el contenedor de la base de datos al levantarse por primera vez. Las tablas principales del sistema se detallan en el Cuadro 5.1.

Además de estas tablas, existe un conjunto de tablas de propuestas que sostienen el flujo de moderación: `Nueva_Policlinica`, `Editar_Policlinica`, `Eliminar_Policlinica`, y sus equivalentes para prestadores y especialidades. Cada una registra el usuario que realizó la propuesta, los datos propuestos y la fecha, permitiendo que los administradores la revisen y aprueben o rechacen.

5.4.3. Consultas geoespaciales

La búsqueda de policlínicas por proximidad se apoya en las capacidades de PostGIS. Dado que las coordenadas están almacenadas en WGS84 (SRID 4326), para calcular distancias reales en metros se proyectan al sistema métrico SRID 3857, lo que permite usar `ST_Distance` para filtrar resultados dentro de un radio definido por el usuario. Para acelerar estas consultas se creó un índice de tipo GiST sobre el campo de geometría:

Tabla	Descripción
`Usuario`	Usuarios del sistema. Incluye email, contraseña hasheada, tipo (`Normal`, `AdminPrestador`, `AdminPlataforma`) y, opcionalmente, el prestador asociado.
`Prestador_de_Salud`	Instituciones de salud registradas en la plataforma.
`Policlinica`	Centros de atención asociados a un prestador de salud.
`Ubicacion_Policlinica`	Dirección y coordenadas de cada policlínica. Las coordenadas se almacenan en el campo `geom` de tipo `GEOMETRY(Point, 4326)`, usando el sistema de referencia WGS84.
`Especialidad`	Especialidades médicas ofrecidas por una policlínica.
`Servicio`	Servicios disponibles en una policlínica, con nombre, descripción y horario.
`Reseña`	Reseñas de usuarios sobre policlínicas (puntuación 1–5 y comentario).
`Favorito`	Relación entre un usuario y una policlínica guardada como favorita.
`Notificacion`	Mensajes generados para un usuario, con fecha y estado de lectura.
`PasswordReset`	Tokens de restablecimiento de contraseña, con hash SHA-256 y fecha de expiración.

Tabla 5.1: Tablas principales de la base de datos.

```
CREATE INDEX idx_ubicacion_geom
    ON Ubicacion_Policlinica USING gist (geom);
```

De forma análoga, para las búsquedas por nombre de especialidad se creó un índice GIN con soporte de trigramas:

```
CREATE INDEX idx_especialidad_nombre_trgm
    ON Especialidad USING gin (nombre gin_trgm_ops);
```

5.5. Frontend

5.5.1. Estructura de páginas

El frontend sigue la convención del App Router de Next.js, donde cada carpeta dentro de `app/` define una ruta de la aplicación. Las páginas principales son:

- `/`: página de inicio.

- `/login` y `/registro`: autenticación de usuarios.
- `/recuperar-contrasena` y `/reset-password`: flujo de recuperación de contraseña.
- `/visualizar`: listado y búsqueda de policlínicas con mapa interactivo.
- `/visualizar/[id]`: vista detallada de una policlínica, con especialidades, servicios, reseñas y ubicación en el mapa.
- `/comparar`: comparación lado a lado de múltiples policlínicas.
- `/perfil`: datos del usuario autenticado y lista de favoritos.
- `/notificaciones`: bandeja de notificaciones del usuario.
- `/admin`: panel del administrador de plataforma (moderación de reseñas y estadísticas generales).
- `/admin-prestador`: panel del administrador de prestador (gestión de policlínicas y revisión de propuestas).
- `/configuracion`: configuración de cuenta y cambio de contraseña.

5.5.2. Estado global con contextos

En aplicaciones web modernas es común que múltiples componentes necesiten acceder o modificar información compartida, como los datos de la sesión del usuario, sus preferencias o ciertas configuraciones de la aplicación. Manejar este tipo de información mediante el pasaje de datos entre componentes (conocido en React como props) puede volver el código más complejo y difícil de mantener, especialmente a medida que crece la cantidad de componentes involucrados. Por esta razón, muchas aplicaciones utilizan mecanismos de estado global que permiten centralizar y compartir datos entre distintas partes de la interfaz.

Para manejar el estado compartido entre componentes se utilizó la API de contextos de React, sin recurrir a librerías externas de manejo de estado como Redux[6]. Se definieron cinco contextos principales: `AuthContext` (datos de la sesión, login, logout y registro), `FavoritesContext` (policlínicas favoritas, sincronizadas con el backend), `NotificationsContext` (notificaciones del usuario y conteo de no leídas), `ReviewsContext` (carga y envío de reseñas) y `CompareContext` (policlínicas seleccionadas para comparar).

Todos los contextos se inyectan desde el layout raíz de la aplicación, lo que los hace accesibles desde cualquier página sin necesidad de pasar props manualmente a través de los componentes.

[6] https://es.redux.js.org/

5.5.3. Visualización geográfica

Para los mapas se integró `Leaflet.js` con tiles de `OpenStreetMap`. Dado que Next.js realiza renderizado del lado del servidor y Leaflet depende de APIs del navegador (`window`, `document`), los componentes de mapa se cargan de forma dinámica usando `next/dynamic` con la opción `ssr: false`, garantizando que Leaflet se inicialice únicamente en el cliente.

Se desarrollaron dos componentes reutilizables para los distintos usos del mapa: un selector de ubicación y una vista de solo lectura para mostrar la ubicación al usuario final.

5.5.4. Comunicación con el backend

La comunicación con la API está organizada en una carpeta `lib/api/`, donde cada entidad tiene su propio cliente TypeScript que encapsula las llamadas `fetch` e incluye la configuración base: URL del backend, inclusión automática de cookies con `credentials: 'include'` y manejo de errores. Esta centralización simplifica el mantenimiento: si cambia la URL base o la estrategia de autenticación, la modificación se hace en un único lugar.

5.6. Autenticación y autorización

5.6.1. JWT y cookies

La autenticación se basa en tokens JWT firmados con el algoritmo `HS256`. Al iniciar sesión, el backend genera el token y lo envía al cliente almacenado en una cookie con los atributos `HttpOnly`, `Secure` y `SameSite=None`. Al ser `HttpOnly`, la cookie no es accesible desde JavaScript, lo que protege el token frente a ataques XSS. En las solicitudes posteriores, el navegador la incluye automáticamente y el backend la valida a través de una dependencia de FastAPI.

El usuario puede optar por una sesión persistente mediante la opción "Recordarme", en cuyo caso la validez del token se extiende a 60 días en lugar de la duración por defecto.

5.6.2. Control de acceso por roles

El sistema maneja tres roles de usuario: `Normal`, `AdminPrestador` y *AdminPlataforma*. La autorización se implementa como dependencias de FastAPI inyectadas en los endpoints que requieren un rol específico, lo que permite declarar el control de acceso de forma explícita en la propia definición del endpoint.

El rol `AdminPrestador` tiene acceso restringido a los recursos de su propio prestador. La dependencia correspondiente verifica no solo que el usuario tenga ese rol sino también que el recurso al que intenta acceder pertenezca al prestador asignado.

5.6.3. Recuperación de contraseña

El flujo de recuperación funciona de la siguiente manera: el usuario solicita el restablecimiento ingresando su email; el sistema genera un token seguro, almacena su hash SHA-256 en la tabla `PasswordReset` con una fecha de expiración de 15 minutos, y envía un email con un enlace que contiene el token en texto plano. Cuando el usuario accede al enlace y envía una nueva contraseña, el backend calcula el hash del token recibido, lo compara con el almacenado, verifica que no haya expirado ni sido usado previamente, y recién entonces actualiza la contraseña. Almacenar el hash en lugar del token original protege ante una eventual filtración de la base de datos.

5.7. Despliegue

Para publicar la solución desarrollada del proyecto, se decidió usar el proveedor de alojamiento Render[7]. El motivo central de esta decisión radica en que de que no existen muchos proveedores de alojamiento que ofrecen planes gratuitos acordes a las necesidades del proyecto. Asimismo, este servicio cuenta con ciertas desventajas que se comentarán más adelante, pero para propósito demostrativo del sistema, este proveedor de alojamiento es más que suficiente.

En particular, el plan gratuito permitió realizar el despliegue de dos servicios web independientes, correspondientes al frontend de la aplicación y al backend del sistema, los cuales se encuentran accesibles públicamente y pueden comunicarse entre sí a través de la red. Esto facilita la arquitectura cliente-servidor utilizada por el proyecto, permitiendo que la interfaz web interactúe correctamente con la API del sistema.

Además, la plataforma ofrece servicios de despliegue para bases de datos accesibles públicamente, la cual queda visible para que el backend pueda comunicarse y utilizarla de manera remota, sin necesidad de alojar ambas partes en un mismo dispositivo. De igual manera, al estar expuesta de manera pública, se permite el monitoreo de la propia base desde dispositivos externos, lo cual a propósito de administración y mantenimiento del mismo puede ser altamente útil facilitando el trabajo remoto con esta.

Por último, uno de los beneficios de peso que conlleva el uso de Render, es que este se puede asociar a una rama de un proyecto de github o gitlab el cual toma como referencia y utiliza para hacer el despliegue. Funcionando como el despliegue CI/CD de gitlab o la funcion actions de github y generando así un sistema de desarrollo continuo en el cual todos los commits realizados a la rama asociada al proyecto de Render, serán leidos por este, copiados y deployeados automáticamente por el servidor. De esta forma, el despliegue a producción se vuelve algo tan simple como hacer un commit o un merge a la rama de

[7] https://render.com/

producción asociada a Render. El único problema particular que existe con la configuración actual del proyecto es que Render no reconoce el servicio gitlab autogestionado por la facultad. Por lo que para lograr el despliegue del proyecto en Render, se decidió establecer en el proyecto de gitlab, una copia automática que se encarga de crear un "mirror" de la rama, es decir, una copia sincronizada con los últimos cambios, en un proyecto de github aparte. Creando asi una copia de la rama utilizada en producción que se sincroniza con cada cambio que se haga sobre la rama existente en gitlab. De esta forma, Render es capaz de desplegar el proyecto a pesar de ser desarrollado en un servicio autogestionado y no reconocido por el servicio de alojamiento web.

Las principales limitaciones del plan gratuito de Render se relacionan con restricciones en la persistencia de datos, disponibilidad y recursos asignados. En particular, la base de datos puede ser eliminada luego de un período de inactividad prolongado y presenta límites en su capacidad de almacenamiento, lo que la hace poco adecuada para entornos productivos.

Asimismo, los servicios web pueden entrar en estado de suspensión tras períodos de inactividad, generando demoras en el tiempo de respuesta al reactivarse. A esto se suman restricciones en los recursos disponibles (CPU y memoria), lo que puede afectar el rendimiento ante múltiples usuarios concurrentes.

Por último, el plan gratuito limita las posibilidades de escalado y configuración del entorno, dificultando la adaptación del sistema ante un aumento en la demanda. En este sentido, si bien estas características resultan aceptables para fines demostrativos, no son adecuadas para un despliegue productivo, para el cual sería recomendable optar por un plan de pago.

Para el deployment del sistema se levantaron tres servicios principales en la plataforma Render.

5.7.1. Backend

Para el backend se decidió mantener el servicio contenedorizado mediante Docker. Esta decisión se tomó principalmente por motivos de consistencia del entorno de ejecución y facilidad de implementación. Si bien Render provee entornos de ejecución gestionados, el uso de Docker permite definir explícitamente el entorno completo en el que se ejecuta la aplicación, incluyendo versiones de dependencias, configuración del sistema y herramientas necesarias para su funcionamiento. De esta forma, se evita depender completamente de la configuración del entorno provisto por la plataforma, algo que escapa al control directo del desarrollador.

Mediante el uso de contenedores se asegura una consistencia tanto en la ejecución como en las dependencias del sistema, garantizando que el backend se comporte de la misma manera tanto en el entorno de desarrollo como en el entorno de despliegue.

Adicionalmente, Render permite configurar fácilmente el puerto de ejecución del servicio y el dominio público a través del cual se expone la API REST, lo que facilita la integración con el frontend de la aplicación.

Por otro lado, además de la configuración del puerto de ejecución del servicio, Render también ofrece un entorno de configuración visual intuitivo, lo cual simplifica considerablemente el mantenimiento y ajuste de los servicios desplegados, especialmente para desarrolladores con menor experiencia en infraestructura cloud, por lo cual, podría ser una buena opción si se planea escalar a futuro el sistema o el equipo de desarrollo trabajando en el proyecto.

5.7.2. Frontend

Para el frontend no se utilizó contenedorización mediante Docker. Esto se debe a que durante el desarrollo del sistema tampoco se utilizó Docker para la ejecución de la interfaz web, por lo que se optó por mantener un modelo de despliegue más simple y directo.

El frontend, desarrollado utilizando Next.js, fue desplegado como un servicio web estándar de Render, el cual se encarga de realizar automáticamente del proceso de instalación de dependencias, compilación del proyecto y ejecución del servidor web correspondiente, esto lo hace utilizando los comandos definidos por el equipo en la etapa de configuración de despliegue del web service.

Este enfoque permite simplificar el proceso de despliegue, ya que Render se encarga de gestionar el entorno de ejecución necesario para aplicaciones basadas en Node.js.

5.7.3. Base de Datos

Para el almacenamiento de la información del sistema se utilizó el servicio administrador de bases de datos PostgreSQL ofrecido por Render.
Este servicio permite desplegar una base de datos accesible públicamente, utilizada por el backend para almacenar y consultar la información necesaria para el funcionamiento del sistema.

Capítulo 6

Pruebas y Validaciones

En este capítulo se describe el proceso de pruebas llevado a cabo sobre el sistema desarrollado. Para esta tarea, no se contó con herramientas de testing automatizado. Las pruebas se realizaron de forma manual por los integrantes del equipo, complementadas con sesiones de validación con usuarios externos al proyecto.

El enfoque adoptado fue el de *testing exploratorio planificado*: se definieron de antemano los casos de prueba a ejecutar, cubriendo los flujos principales del sistema, y durante la ejecución se registraron los resultados obtenidos y los defectos encontrados. Estos fueron clasificados y corregidos antes de la entrega final.

6.1. Casos de Prueba

Se definieron casos de prueba organizados por módulo funcional, siguiendo los requerimientos presentados en el capítulo 3. Para cada caso se especificó una descripción, los pasos a seguir, el resultado esperado y el resultado obtenido. En la tabla 6.1 se presenta un resumen del total de casos ejecutados por módulo.

6.1.1. Autenticación y gestión de cuenta

Las pruebas de autenticación cubrieron el flujo completo de registro de un nuevo usuario, inicio de sesión con credenciales correctas e incorrectas, cierre de sesión y recuperación de contraseña por email. Se verificó que el sistema rechace intentos de acceso con datos inválidos y que el token de recuperación expire pasados los 15 minutos definidos en la especificación.

6.1.2. Visualización e interacción con el mapa

Se verificó que las policlínicas se carguen correctamente sobre el mapa y que la interacción con los marcadores muestre la información básica del centro en el popup. Se probaron los filtros de búsqueda por nombre y dirección, validando

Módulo	Descripción	Estado
Autenticación	Registro, login, logout, recuperación y cambio de contraseña	OK
Visualización	Búsqueda, filtros, mapa interactivo y detalle de policlínica	OK
Favoritos y reseñas	Gestión de favoritos y creación de reseñas	OK
Contribución VGI	Propuesta de nueva policlínica, edición y baja	OK
Panel del prestador	Alta/baja de policlínica, carga de servicios y revisión de propuestas VGI	OK
Panel del administrador	Moderación de reseñas y estadísticas	OK

Tabla 6.1: Resumen de casos de prueba ejecutados por módulo.

que los resultados se actualicen en tiempo real. También se navegó al detalle de distintas policlínicas para comprobar que las especialidades, servicios, horarios y reseñas se muestren de forma correcta.

6.1.3. Favoritos, reseñas y notificaciones

Se comprobó que un usuario autenticado pueda marcar y desmarcar policlínicas como favoritas y que éstas aparezcan en su perfil. Para las reseñas se verificó el flujo completo: creación de la reseña, aparición en la vista del centro y gestión de la misma a cargo del administrador de plataforma. Se validó también que una reseña eliminada por el administrador no sea visible para otros usuarios.

En cuanto a las notificaciones, se verificó que al aprobarse la incorporación de una nueva policlínica asociada a un prestador que el usuario tiene en favoritos, llegue la notificación correspondiente. También se validó la notificación que se genera en caso de que se cree una nueva policlínica asociada a un prestador de salud que se encuentre entre las policlínicas favoritas del usuario.

6.1.4. Contribución VGI

Se realizaron pruebas sobre el flujo de Información Geográfica Voluntaria (VGI), contemplando la propuesta de incorporación de una nueva policlínica con su correspondiente ubicación en el mapa, la edición de datos existentes y la solicitud de cierre de un centro. En cada uno de los casos, se verificó que la propuesta quedara registrada en estado pendiente y que el administrador del prestador de salud correspondiente pudiera gestionarla (aprobándola o rechazándola) desde su panel de control.

6.1.5. Panel del administrador de prestador

Se probaron las funcionalidades disponibles para el rol `AdminPrestador`: alta manual de una policlínica, baja de una existente, carga de especialidades y

servicios y gestión de propuestas ingresadas por usuarios. También se verificó que el administrador solo pueda operar sobre los recursos de su propio prestador. Además, se validó que al aprobar o rechazar una propuesta el estado se actualice correctamente y que el usuario proponente reciba la notificación correspondiente.

6.1.6. Panel del administrador de plataforma

Se probaron las funcionalidades del rol `AdminPlataforma`: revisión y moderación de reseñas y visualización de estadísticas generales de la plataforma.

6.2. Validación con usuarios

Adicionalmente a las pruebas internas del equipo, se realizaron sesiones de validación con personas externas al proyecto. Estos utilizaron la plataforma, ejecutando los diferentes casos de uso definidos para el sistema, como por ejemplo buscar una policlínica por nombre, consultar su detalle en el mapa, agregar un favorito y dejar una reseña.

El objetivo de estas sesiones no fue medir métricas formales de usabilidad sino obtener retroalimentación sobre aspectos que resultaran confusos para alguien que interactuara con el sistema por primera vez. Los resultados obtenidos se muestran en la figura 6.1:

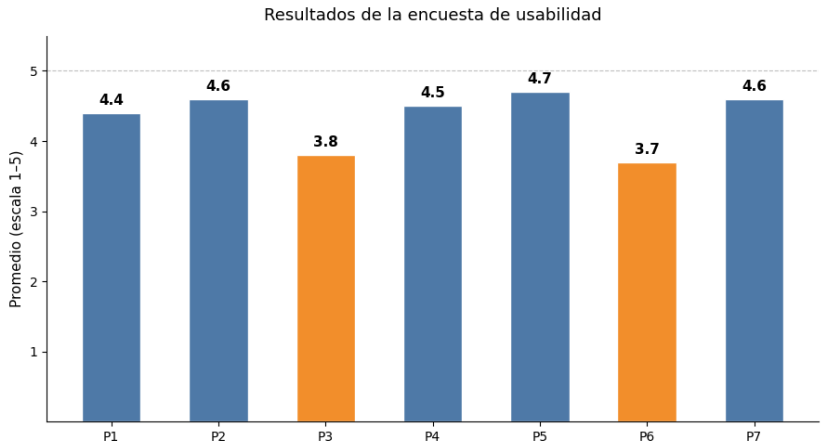


Figura 6.1: Resultados de la encuesta de usabilidad (n=10, escala 1–5).

Las respuestas se midieron en una escala del 1 al 5, donde 1 representa total desacuerdo y 5 representa total acuerdo. Cada barra corresponde a las siguientes afirmaciones:

- **P1:** Encontré la aplicación fácil de usar.

- **P2:** Me resultó sencillo encontrar policlínicas cercanas a mi ubicación.
- **P3:** La información sobre cada policlínica (nombre, dirección, horarios) es clara y suficiente.
- **P4:** El mapa me ayudó a entender dónde está ubicada cada policlínica.
- **P5:** El proceso para proponer una nueva policlínica fue claro y sencillo.
- **P6:** La aplicación muestra toda la información que necesito sobre cada policlínica.
- **P7:** Volvería a usar esta aplicación si necesito encontrar una policlínica.

6.3. Resultados de las pruebas

A lo largo del proceso de pruebas se identificaron y resolvieron defectos de diferentes tipos.

6.3.1. Defectos de configuración y despliegue

En esta categoría se identificaron tres defectos:

- **resenas/resenias**: Este defecto se introdujo durante la etapa de desarrollo. El mismo consistía en que el sistema intentaba acceder a un endpoint del backend mediante la ruta `/resenas/`, cuando la ruta correcta era `/resenias/`. Esta inconsistencia fue detectada a partir de las pruebas realizadas con usuarios en la etapa de producción.
- **Recuperar contraseña**: Este defecto fue introducido en la etapa de despliegue y detectado en la etapa de producción. Al realizar las pruebas del sistema luego de desplegar la aplicación, se observó que el endpoint para recuperar la contraseña del usuario, el cual utiliza la herramienta Gmail para el envío de correos electrónicos automáticos, no funciona en dicha etapa, a pesar de operar correctamente en la etapa de desarrollo. Esto se debe a que Render no permite el uso de Gmail para este tipo de transacciones, con el fin de evitar el spam.
- **Configuración de la cookie del usuario**: Al igual que el primer defecto, se introdujo en la etapa de desarrollo y se detectó en la etapa de producción. El motivo de este defecto fue debido a que el atributo `samesite` de la cookie de autenticación se encontraba hardcodeado con el valor `lax`. Esta configuración resultó inadecuada para el funcionamiento correcto del sistema en determinados flujos (cambiar la contraseña y cerrar sesión).

6.3.2. Defectos de errores en código

Para este categoría se detectaron 3 defectos:

- **Comparar policlínicas por parte de distintos usuarios en el mismo equipo**: El defecto consistía en que si un usuario hacía una comparación, cuando cerraba su sesión e ingresaba con otro usuario en el mismo equipo, la comparación seguía apareciendo para este último.
- **Reseteo de campos en el formulario para proponer edición**: El inconveniente se producía cuando un usuario realizaba una propuesta de edición para los datos de una policlínica, al rellenar los campos de `nombre` y `dirección` con los nuevos datos, y luego clickear la ubicación en el mapa, los campos anteriores se reseteaban a los originales. De forma similar ocurría en el caso de que un usuario administrador de prestador realizaba una edición sobre los datos de la policlínica dentro de `Ver detalles`.
- **Persistencia de datos en el formulario para reportar cierre de policlínica**: Este comportamiento inesperado se presentaba cuando un usuario reportaba un cierre de policlínica y completaba el formulario indicando el motivo del reporte, al clickear en cancelar o clickear afuera del formulario y luego volver a clickear en Reportar Cierre de Policlínica, los datos anteriores ingresados en el formulario seguían apareciendo.

En términos generales, el sistema respondió de forma correcta a los flujos principales definidos en los requerimientos funcionales. Todos los defectos encontrados fueron en su mayoría de bajo impacto y pudieron corregirse sin necesidad de cambios estructurales en la arquitectura, a excepción de recuperar contraseña. La solución para este defecto implica utilizar una api de pago para el envío automático de emails, por lo que se decidió no corregirlo ya que no era una funcionalidad crítica para el funcionamiento de la aplicación.

Capítulo 7

Gestión del proyecto

En este capítulo se presenta la gestión integral del proyecto, abarcando las actividades de planificación, seguimiento y control llevadas a cabo durante su desarrollo. Se detalla el cronograma de trabajo definido inicialmente, representado mediante un diagrama de Gantt, y se analizan las principales desviaciones ocurridas respecto a la planificación original.

Asimismo, se describe la organización del equipo y el esquema de reuniones establecidas, comparando las instancias planificadas con las efectivamente realizadas. Este análisis permite evaluar el grado de cumplimiento de los hitos propuestos, la eficiencia en la comunicación interna y la capacidad de adaptación frente a imprevistos.

El objetivo de este capítulo es ofrecer una visión completa del proceso de gestión del proyecto, identificando aprendizajes, buenas prácticas y oportunidades de mejora que contribuyan a la ejecución de proyectos futuros con una planificación y coordinación más efectivas.

7.1. Etapas de la Gestión

Para la realización de este proyecto, se llevaron a cabo distintas etapas las cuales fueron definidas como:

1. Estudio Inicial del dominio, conceptos base y trabajos relacionados. Relevamiento y Definición de Requerimientos
2. Diseño de la solución. Desarrollo de pruebas de concepto técnicas, definición y armado del ambiente de desarrollo.
3. Implementación
4. Pruebas del Sistema y Documentación Final

En la etapa inicial del proyecto se estudia el dominio del mismo y se realiza una investigación de antecedentes, intentando encontrar otros trabajos relacionados con la temática del proyecto de grado para así poder comprender mejor

el contexto del mismo, además de servir como punto de partida. Dentro de esta instancia también se inicia con el análisis y definición de requerimientos. Toda esta etapa abarca los primeros dos meses de proyecto.

Durante la segunda etapa se finaliza con la definición de requerimientos, se define la arquitectura del sistema y se construye el entorno de desarrollo para implementar la solución diseñada. Esta etapa abarca los meses 3 y 4 del proyecto.

La tercera etapa del proyecto se basa en la implementación de la solución diseñada para cumplir con los requerimientos definidos anteriormente. Dicha etapa comprende los meses 5 y 6 de proyecto.

Finalmente, en la ultima etapa del proyecto, se realizan pruebas para validar la solución implementada y también se finaliza con la elaboración del informe, el cual se fue construyendo en el transcurso de las etapas anteriores del proyecto. Esta etapa abarca los meses 7 y 8 de proyecto.

7.2. Planificación

Al inicio del proyecto, se realizaron diferentes estimaciones para planificar su desarrollo. En base a esto, se elaboró un diagrama de Gantt con las principales tareas previstas (ver figura 7.1).

Una vez finalizado el proyecto, se construyó un nuevo diagrama de Gantt que refleja la ejecución real de las actividades (ver figura 7.2). La comparación entre ambos permite observar diferencias entre la planificación inicial y el desarrollo efectivo, dejando en evidencia ajustes en los tiempos de algunas tareas.

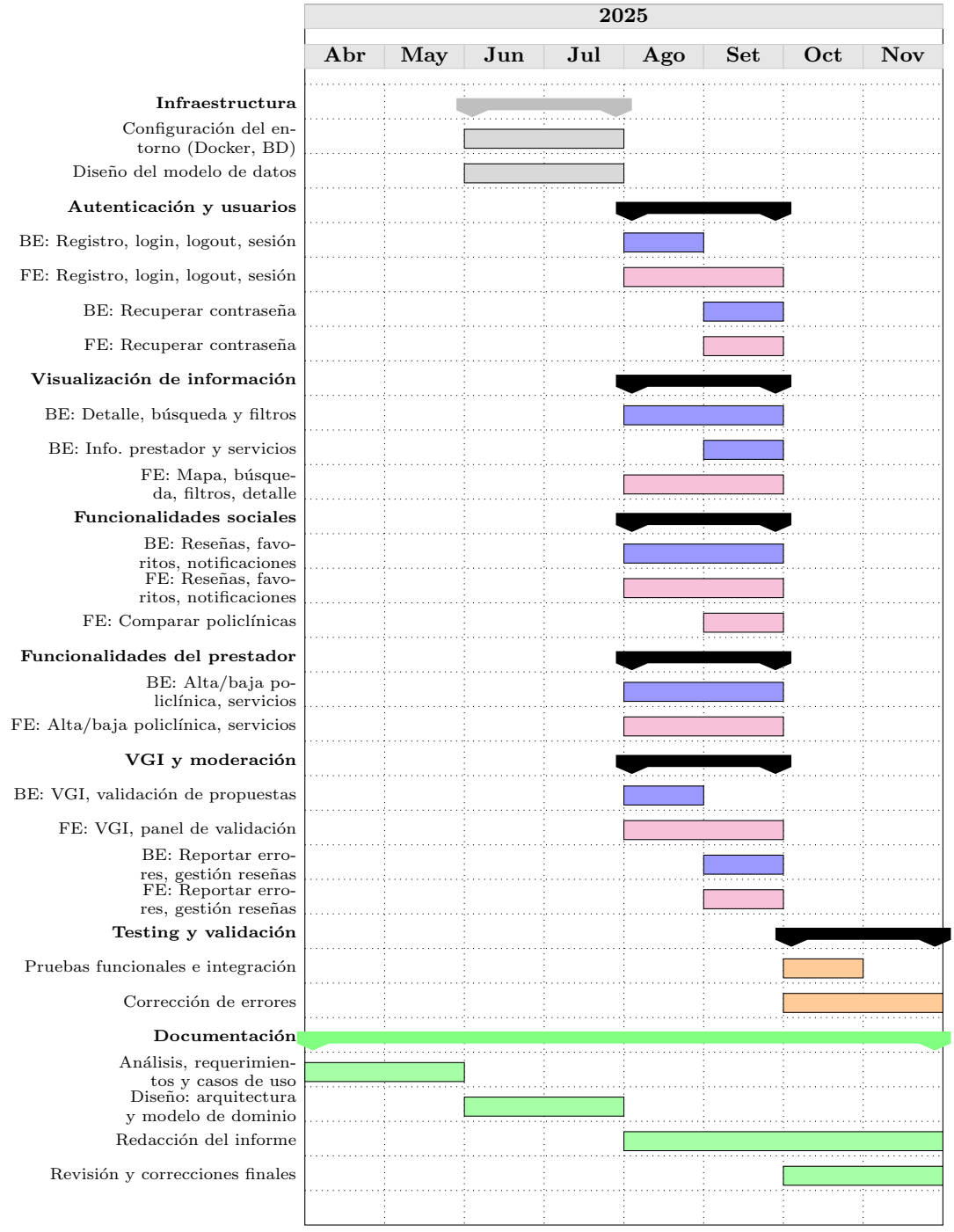


Figura 7.1: Planificación inicial del proyecto *Salud en Cercanía* (abril 2025 – noviembre 2025).

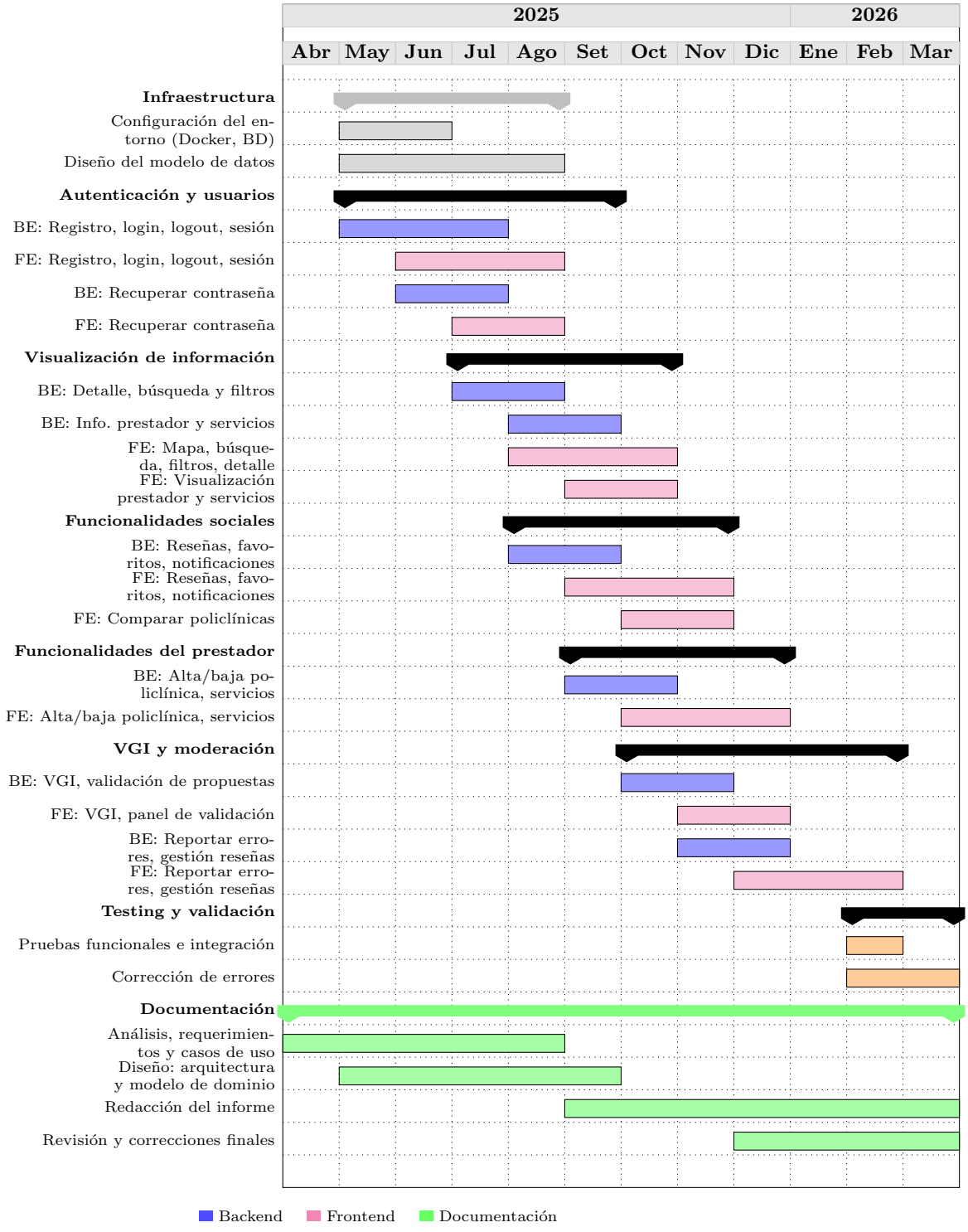


Figura 7.2: Diagrama de Gantt del proyecto *Salud en Cercanía* (abril 2025 – marzo 2026).

7.3. Herramientas

Para la realización del proyecto se emplearon diversas herramientas destinadas a facilitar la planificación, el desarrollo y la documentación del trabajo. Estas herramientas permitieron coordinar las tareas del equipo, gestionar el código fuente y mantener organizada la documentación generada durante el proyecto.

- **Trello**: Herramienta de gestión de proyectos basada en tableros Kanban. Se utilizó para organizar las tareas, asignar responsabilidades y realizar el seguimiento del avance mediante listas y tarjetas.[1]
- **Google Meet**: Plataforma de videoconferencias utilizada para la comunicación sincrónica del equipo. Permitió realizar reuniones periódicas de coordinación y revisión de avances.[2]
- **Zoom**: Plataforma de videoconferencias al igual que Google Meet, esta herramienta en particular fue utilizada para la comunicación sincrónica del equipo con la tutora del proyecto, permitió realizar reuniones quincenales para poder compartir avances del proyecto por parte del equipo y correcciones del mismo por parte de la tutora.[3]
- **Gmail**: Servicio de correo electrónico desarrollado por Google, que permite enviar, recibir y organizar correos. Se adoptó dicha herramienta para poder coordinar, y reagendar reuniones con la tutora del proyecto, además de poder enviar directamente algún documento para su revisión posterior.[4]
- **GitLab**: Plataforma de control de versiones basada en Git. Se empleó para la gestión del repositorio del proyecto y el trabajo colaborativo mediante ramas (*branches*).[5]
- **Google Drive**: Servicio de almacenamiento en la nube para la gestión de archivos. Se utilizó para compartir documentos, almacenar material de referencia y mantener respaldos de la documentación relevante.[6]
- **Overleaf**: Entorno colaborativo en línea para la edición de documentos LaTeX, permitió la redacción en conjunto del informe final.[7]

7.4. Desviaciones

Durante el desarrollo del proyecto ocurrieron ciertas desviaciones en algunas etapas del mismo, lo cual retrasó el cronograma establecido al inicio.

[1] https://trello.com/es
[2] https://meet.google.com/landing
[3] https://www.zoom.com/es
[4] https://workspace.google.com/intl/es-419/gmail/
[5] https://about.gitlab.com/
[6] https://workspace.google.com/intl/es-419/products/drive/
[7] https://es.overleaf.com/

Para la primera etapa del proyecto no hubo ninguna desviación, la misma fue llevada a cabo en el tiempo estimado de dos meses.
En la segunda etapa ocurrió la primera desviación del proyecto: hubo que realizar diferentes ajustes sobre el documento de requerimientos, esto retrasó el diseño de la arquitectura y también la implementación del entorno de desarrollo. Agregado a esto, durante el último mes de esta etapa todos los integrantes del equipo debieron rendir pruebas semestrales (parciales y exámenes).

Como consecuencia, el inicio de la etapa de implementación se vio demorado. El mismo contó con otra desviación, en este caso hubo un fallo de comunicación entre dos integrantes del grupo que se encontraban implementando el backend, lo que resultó en ambos integrantes realizando las mismas tareas sin haber respetado lo estipulado en la herramienta Trello. Debido a esta suma de desviaciones, la primera fase de la etapa de implementación se vió afectada nuevamente. A raíz de esto, el grupo logró reforzar la comunicación, además de decidir llevar a cabo más reuniones para notificar el estado de las tareas que se encontraba realizando cada integrante e incrementar el uso de las herramientas de gestión.

Posteriormente, se registraron nuevas desviaciones durante la fase de integración entre frontend y backend, lo que contribuyó a extender los plazos previstos. Otro problema fue que se realizaron cambios en los requerimientos y en la arquitectura del sistema, por lo tanto hubo que realizar modificaciones, no solo en los documentos sino también en la implementación de la solución.

Finalmente, durante la fase de despliegue del proyecto en la nube, se encontró un problema, el cual resultó en tener que reconsiderar la solución inicialmente seleccionada. Por lo tanto, fue necesario realizar diferentes modificaciones sobre componentes ya implementados.

Capítulo 8

Conclusiones y Trabajo Futuro

En este capítulo se exponen los resultados más relevantes obtenidos durante el desarrollo del proyecto, considerando el nivel de cumplimiento de los objetivos definidos en la etapa inicial y las dificultades que se presentaron en el proceso. Asimismo, se plantean posibles mejoras y líneas de trabajo futuras orientadas a la evolución de la solución implementada.

8.1. Conclusiones

Tras la finalización del proyecto, se cumplieron todos los objetivos planteados inicialmente, lo que dio lugar a una solución estable. Como etapa inicial del proyecto, se realizó una investigación extensa sobre plataformas orientadas a los servicios de salud que, además, utilizan información geográfica. Dichas plataformas fueron analizadas en profundidad en el capítulo 2, donde también se realizó una comparación entre ellas. Si se agrega el producto final *Salud en Cercanía* a la comparación anterior, se obtiene la tabla 8.1:

Funcionalidad	**Healthsites.io**	**AtuServicio**	**Doctoralia**	**Medicare**	**Salud en Cercanía**
VGI	Sí	No	No	No	Sí
Vis. mapa	Sí	No	Sí	Sí	Sí
Info. detallada	No	Sí	Sí	Sí	Sí
Interfaz amigable	No	Sí	Sí	Sí	Sí
Búsq. calle/zona	No	No	Sí	Sí	Sí
Búsq. prestador	Sí	Sí	No	No	Sí
Búsq. especialidad	No	No	Sí	Sí	No
Favoritos	No	No	Sí	Sí	Sí
Reseñas	No	No	No	Sí	Sí
Comparar	No	Sí	Sí	Sí	Sí
Filtrar género	No	No	Sí	Sí	No

Tabla 8.1: Comparación de funcionalidades entre las plataformas analizadas, incluyendo el proyecto Salud en Cercanía.

A partir del análisis de la tabla, se observa que *Salud en Cercanía* cumple con el objetivo general del proyecto, ya que se trata de una plataforma que permite la captura, gestión y visualización de información georreferenciada sobre los servicios de salud, integrando datos provenientes tanto de fuentes oficiales como de aportes ciudadanos.

En la segunda etapa, se identificaron los requerimientos funcionales y no funcionales del sistema propuesto. Asimismo, se diseñó su arquitectura y se documentó todo lo anterior en un documento específico, el cual fue actualizado de forma constante durante el desarrollo del proyecto.

Durante la etapa de implementación, se desarrolló un prototipo funcional que cumple con el objetivo planteado de registrar, consultar y visualizar prestadores de salud en un mapa. En el transcurso de esta etapa, se incorporó un mecanismo de registro de usuarios y un sistema de control de acceso diferenciado según el rol de los mismos.

Finalmente, se realizaron diversos casos de prueba sobre el prototipo implementado y se solucionaron los errores detectados en los mismos, lo que permitió validar el producto desarrollado en este proyecto.

8.2. Trabajo futuro

Como trabajo futuro, se establecen los siguientes puntos:

- **Inicio de sesión con cuenta Google**: Hasta el momento, la única forma de ingresar a la plataforma es registrándose directamente en la misma. La posibilidad de iniciar sesión mediante una cuenta de Google podría facilitar significativamente el proceso actual de registro de usuarios.

- **Posibilidad de tener asignado un Prestador de Salud y recibir notificaciones sobre este**: Actualmente, solo se pueden marcar como favoritas las policlínicas de un prestador. En caso de que se dé de alta una nueva policlínica de ese mismo prestador, y el usuario tenga alguna de sus policlínicas marcada como favorita, se le notifica sobre la apertura de esta última. Asignar directamente un prestador al usuario permitiría enviar la notificación de apertura de una nueva policlínica sin necesidad de verificar si el usuario tiene alguna policlínica marcada como favorita. Esto implicaría un proceso más simple y eficiente, que podría mejorar el rendimiento de la aplicación.

- **Mostrar datos administrativos del Prestador utilizando la plataforma de A Tu Servicio**: La plataforma A Tu Servicio tuvo gran importancia en el desarrollo de Salud en Cercanía, como se mencionó anteriormente. Gracias a sus datos abiertos, no solo fue posible obtener información sobre los prestadores de salud, sino también diversos datos acerca de sus policlínicas. Además, podrían incorporarse datos de carácter administrativo sobre los prestadores de salud, como el total de afiliados, el tiempo promedio de espera para obtener un turno, el promedio de precios

de medicamentos, consultas y estudios, entre otros. Estos datos podrían resultar de gran interés para los usuarios al momento de elegir un prestador de salud.

- **Agregar más filtros de búsqueda**: Aunque actualmente es posible filtrar por policlínicas de un prestador, se muestran todas las policlínicas de ese prestador a nivel nacional, lo cual puede resultar incómodo para el usuario, ya que podría encontrarse con demasiados puntos en el mapa, cuando quizás solo le interesan los de una zona específica. Por lo tanto, incorporar un filtrado por zona (por ejemplo, departamento o barrio) podría mejorar la experiencia de usuario. Asimismo, podrían añadirse otros filtros, como por especialidad.

- **Poder registrar los nombres de los médicos de cada policlínica y su especialidad**: Inicialmente, se consideró esta posibilidad, pero no fue posible implementarla debido a que los datos iniciales de carga del sistema no contaban con dicha información. No obstante, a través de aportes VGI por parte de los usuarios, sería posible obtener estos datos. Asimismo, en relación con el punto anterior, también podría implementarse un filtrado de especialistas por género.

- **Recompensas a los usuarios con mayor aporte de información VGI**: Una posible mejora consiste en incentivar a los usuarios a aportar información VGI relevante para el sistema. Actualmente, en cada solicitud VGI se registra qué usuario la envía, por lo que no sería complejo contabilizar cuántas solicitudes válidas aporta cada uno. Esto permitiría implementar un sistema de ranking con recompensas.

- **Usar una API de pago para recuperar la contraseña vía mail**: Durante la etapa de desarrollo, esta funcionalidad fue implementada utilizando Gmail para el envío automático de correos electrónicos. Sin embargo, en producción no es posible utilizar este mecanismo, ya que Render no permite establecer conexiones con Gmail para este fin, como medida para evitar el spam. Por lo tanto, para habilitar esta funcionalidad en un entorno productivo, es necesario utilizar una API de pago para el envío de correos electrónicos.

Referencias

[1] M. A. Bernabé-Poveda y C. M. López-Vázquez, Fundamentos de las Infraestructuras de Datos Espaciales (IDE), Madrid: UPM Press, 2012, págs. 69-72.

[2] Víctor Olaya, Sistemas de Información Geográfica, 2014, págs. 3, 8, 172.

[3] Goodchild, M. F. (2007). Citizens as Sensors: The World of Volunteered Geography. GeoJournal, Vol.69(4), págs. 211–221.

[4] OpenStreetMap, Último acceso 04-03-2026, 2006. dirección: https://www.openstreetmap.org/

[5] VGIscience, Último acceso 04-03-2026, 2017. dirección: https://www.vgiscience.org/

[6] Healthsites Último acceso 04-03-2026, 2016. dirección: https://healthsites.io/

[7] Healthsites/About, Último acceso 17-03-2026, 2006. dirección: https://healthsites.io/about

[8] Healthsites API, Último acceso 04-03-2026, 2016. dirección: https://healthsites.io/api/docs/

[9] ATuServicio Último acceso 04-03-2026, 2015. dirección: https://atuservicio.msp.gub.uy/

[10] Data Uruguay, ATuServicio, Último acceso 04-03-2026, 2012. dirección: https://data.org.uy/proyectos/atuservicio-uy/

[11] ATuServicio, sobre el proyecto, Último acceso 04-03-2026, 2015. dirección: https://atuservicio.msp.gub.uy/sobre$_e l_p royecto$

[12] Doctoralia, Último acceso 04-03-2026, 2007. dirección: https://www.doctoralia.es/

[13] Medicare US, Último acceso 04-03-2026, 1998. dirección: https://www.medicare.gov/

[14] Medicare/About , Último acceso 17-03-2026, 2001. dirección: https://www.medicare.gov/about-us)

Anexo A

Casos de uso

Este anexo tiene como objetivo describir en mayor profundidad los casos de uso que existen en el sistema desarrollado, a modo de complementar la información proporcionada en la sección correspondiente del informe.

A.1. Casos de Uso

1. **Nombre:** Registro de usuario.

   Objetivo: Permitir a nuevos usuarios crear una cuenta para acceder a la plataforma.

   Actores: Usuario sin registrar.

   Precondiciones: No tener una cuenta registrada.

   Descripción: El usuario se registra completando un formulario con datos básicos.

   Flujo principal:

   a) El usuario accede al formulario de registro.

   b) Completa los campos requeridos: nombre, correo, contraseña, etc.

   c) El sistema valida y guarda los datos.

   Flujos alternativos: 2.A Si el correo ya está registrado, se muestra un mensaje de que el correo ya está registrado.

   Postcondiciones: La cuenta queda registrada y disponible.

2. **Nombre:** Inicio de sesión.

   Objetivo: Permitir que usuarios y administradores accedan a la plataforma con su cuenta.

   Actores: Usuario, Administrador de prestador, Administrador de plataforma.

Precondiciones: Tener una cuenta previamente registrada.

Descripción: El actor accede al sistema ingresando sus credenciales para autenticarse.

Flujo principal:

a) El actor accede a la página de inicio de sesión.

b) Ingresa su correo electrónico y contraseña.

c) El sistema valida las credenciales.

d) Si son válidas, se inicia la sesión.

Flujos alternativos:

3.A Si las credenciales son incorrectas, se muestra un mensaje de error.

3.A.1 El actor puede reintentar o seleccionar Olvidé mi contraseña.

Postcondiciones: El actor accede a su panel correspondiente dentro del sistema.

3. **Nombre:** Cerrar sesión.

   Objetivo: Permitir al actor finalizar su sesión actual de forma segura.

   Actores: Usuario, Administrador de prestador, Administrador de plataforma.

   Precondiciones: Estar autenticado.

   Descripción: El actor selecciona la opción para cerrar sesión y el sistema invalida su token de acceso.

   Flujo principal:

   a) El actor accede a su menú de usuario.

   b) Selecciona la opción Cerrar sesión.

   c) El sistema invalida la sesión y redirige al inicio.

   Flujos alternativos: -

   Postcondiciones: La sesión se termina y se retorna al usuario a la pantalla de inicio.

4. **Nombre:** Recuperar contraseña.

   Objetivo: Permitir que usuarios que olvidaron su contraseña puedan recuperarla.

   Actores: Usuario, Administrador de prestador de salud, Administrador de plataforma.

   Precondiciones: Tener una cuenta registrada.

   Descripción: El usuario solicita un restablecimiento de contraseña mediante correo electrónico.

   Flujo principal:

a) El actor accede a la opción Olvidé mi contraseña.

b) Ingresa su correo registrado.

c) El sistema valida que el correo exista.

d) Se envía un enlace de recuperación al correo.

e) El usuario accede al enlace y crea una nueva contraseña.

Flujos alternativos: 3.A Si el correo no está registrado, el sistema no hace nada.

Postcondiciones: El usuario puede iniciar sesión con la nueva contraseña.

5. **Nombre:** Cambiar contraseña.

   Objetivo: Permitir que los usuarios puedan cambiar su contraseña.

   Actores: Usuario, Administrador de prestador de salud, Administrador de plataforma.

   Precondiciones: Acceder a la plataforma y autenticarse.

   Descripción: El usuario cambia la contraseña actual por una nueva.

   Flujo principal:

   a) El actor accede a su panel de configuración.

   b) Selecciona la opción cambiar contraseña.

   c) Ingresa su nueva contraseña y la confirma.

   Flujos alternativos: -

   Postcondiciones: El usuario cambió su contraseña.

6. **Nombre:** Visualizar policlínicas por prestador.

   Objetivo: Permitir a los usuarios conocer las policlínicas que ofrece cada prestador.

   Actores: Usuario.

   Precondiciones: Acceder a la plataforma.

   Descripción: El usuario accede a la plataforma, selecciona un prestador y visualiza sus policlínicas.

   Flujo principal:

   a) Selecciona un prestador de salud.

   b) El sistema despliega todas las policlínicas asociadas a ese prestador.

   Flujos alternativos: -

   Postcondiciones: El usuario visualiza todas las policlínicas pertenecientes al prestador que seleccionó.

7. **Nombre:** Obtener información detallada de una policlínica.

   Objetivo: Permitir que cualquier usuario acceda a una vista detallada sobre una policlínica específica.

   Actores: Usuario.

   Precondiciones: Acceder a la plataforma.

   Descripción: El usuario selecciona una policlínica de un prestador previamente seleccionado y visualiza su información detallada, como nombre, sitio web, servicios, especialidades, calificación y reseñas.

   Flujo principal:

   a) El usuario accede a la plataforma.

   b) Navega a la sección de prestadores de salud.

   c) Selecciona un prestador de la lista.

   d) El sistema muestra una lista con todas las policlínicas del prestador seleccionado.

   e) El usuario selecciona una policlínica.

   Flujos alternativos: 3.A Si el prestador seleccionado no tiene ninguna policlínica, se muestra un mensaje indicando que no hay datos disponibles.

   Postcondiciones: El usuario obtiene la información detallada sobre la policlínica seleccionada.

8. **Nombre:** Buscar policlínicas por dirección y filtrar por prestador.

   Objetivo: Permitir a los usuarios buscar policlínicas cercanas a una dirección y aplicar filtros.

   Actores: Usuario.

   Precondiciones: Acceder a la plataforma.

   Descripción: El usuario introduce una dirección y el sistema muestra policlínicas cercanas con opción a aplicar filtros.

   Flujo principal:

   a) Ingresa una dirección en el buscador.

   b) El sistema geolocaliza la dirección.

   c) Se muestran las policlínicas cercanas en el mapa.

   d) El usuario puede aplicar filtros.

   Flujos alternativos: -

   Postcondiciones: El usuario obtiene una lista filtrada de policlínicas cercanas.

9. **Nombre:** Calificar y dejar reseña sobre una policlínica.

   Objetivo: Permitir a los usuarios dejar una calificación y escribir una reseña sobre una policlínica.

   Actores: Usuario.

   Precondiciones: Acceder a la plataforma y autenticarse.

   Descripción: El usuario accede a la ficha de una policlínica y deja una calificación y/o reseña.

   Flujo principal:

   a) El usuario accede a la ficha de una policlínica.

   b) Elige una calificación de entre 1 y 5 estrellas y redacta una reseña.

   c) Envía la reseña.

   d) El sistema guarda y muestra la reseña.

   Flujos alternativos: 5.A El usuario solo deja calificación sin comentario.

   Postcondiciones: La reseña queda visible para otros usuarios.

10. **Nombre:** Ingresar nueva policlínica (VGI).

    Objetivo: Permitir que un usuario aporte información sobre una policlínica que no figura en la plataforma.

    Actores: Usuario.

    Precondiciones: Acceder a la plataforma, autenticarse y seleccionar agregar nueva policlínica.

    Descripción: El usuario accede a un formulario para registrar una nueva policlínica con su ubicación e información básica.

    Flujo principal:

    a) El usuario selecciona el prestador para el cual quiere agregar una nueva policlínica.

    b) El usuario selecciona en Agregar Nueva Policlínica.

    c) Ingresa los datos: nombre, dirección y ubicación geográfica.

    d) Envía el formulario.

    e) El sistema guarda la información como pendiente de validación.

    Flujos alternativos: 3.A El sistema detecta que ya existe una policlínica con esos datos y notifica al usuario.

    Postcondiciones: La policlínica queda registrada como propuesta en espera de validación.

11. **Nombre:** Editar datos de policlínica existente (VGI).

    Objetivo: Permitir que un usuario pueda proponer una edición sobre los datos de una policlínica que estén desactualizados.

    Actores: Usuario.

    Precondiciones: Acceder a la plataforma, autenticarse y seleccionar en proponer edición sobre una policlínica seleccionada previamente.

    Descripción: El usuario accede a un formulario para editar los datos de la policlínica seleccionada.

    Flujo principal:

    a) El usuario selecciona el prestador de la policlínica para la cual quiere proponer una edición de sus datos.

    b) El usuario selecciona en Proponer Edición.

    c) Ingresa los datos actualizados (nombre, dirección y ubicación geográfica).

    d) Envía el formulario.

    e) El sistema guarda la información como pendiente de validación.

    Flujos alternativos: -

    Postcondiciones: El pedido de edición de los datos de la policlínica queda registrado como propuesta en espera de validación.

12. **Nombre:** Reportar cierre de una policlínica existente (VGI).

    Objetivo: Permitir que un usuario pueda reportar el cierre de una policlínica que ya no existe y aún esté disponible en el sistema.

    Actores: Usuario.

    Precondiciones: Acceder a la plataforma, autenticarse y seleccionar reportar cierre de policlínica.

    Descripción: El usuario accede a un formulario detallando el motivo de pedido del cierre de la policlínica.

    Flujo principal:

    a) El usuario selecciona el prestador para el cual quiere dar de baja una policlínica.

    b) El usuario selecciona en Reportar Cierre de Policlínica.

    c) Ingresa el motivo por el cual realizó este pedido.

    d) Envía el formulario.

    e) El sistema guarda la información como pendiente de validación.

    Flujos alternativos: -

    Postcondiciones: El pedido de cierre de policlínica queda registrado como propuesta en espera de validación.

13. **Nombre:** Guardar policlínicas favoritas.

    Objetivo: Permitir al usuario guardar policlínicas de su interés para acceder rápidamente a ellas en el futuro.

    Actores: Usuario.

    Precondiciones: Acceder a la plataforma y autenticarse.

    Descripción: El usuario selecciona policlínicas para agregarlas a su lista de favoritos.

    Flujo principal:

    a) El usuario navega por el mapa o los resultados de búsqueda.

    b) Selecciona una policlínica.

    c) Presiona el ícono con forma de corazón.

    d) El sistema guarda la policlínica en la lista personal del usuario.

    Flujos alternativos:

    2.A Si la policlínica ya está en favoritos, se ofrece la opción de quitarla.

    2.A.1 El usuario puede seleccionar ."Eliminar de favoritos".

    Poscondiciones: La policlínica queda registrada (o removida) en la lista de favoritos del usuario.

14. **Nombre:** Recibir notificaciones sobre favoritos.

    Objetivo: Informa al usuario si se abrió una nueva policlínica de un prestador sobre el cual tiene como favorito alguna de sus policlínicas.

    Actores: Usuario.

    Precondiciones: Acceder a la plataforma, autenticarse y tener al menos una policlínica en la lista de favoritos.

    Descripción: El sistema notifica al usuario cada vez que se dé de alta una policlínica de un prestador para el cual ya tiene una policlínica como favorita.

    Flujo principal:

    a) El sistema detecta que se agregó una nueva policlínica y debe notificar sobre ello a todos los usuarios que tengan al menos una policlínica como favorita que pertenezca al mismo prestador de la policlínica que fue dada de alta previamente.

    b) El usuario recibe la notificación en su bandeja de notificaciones.

    Flujos alternativos: -

    Poscondiciones: El usuario es informado sobre la apertura de la nueva policlínica.

15. **Nombre:** Comparar policlínicas.

    Objetivo: Brindar al usuario una herramienta para comparar varias policlínicas en cuanto a servicios, calificaciones, horarios, etc.

    Actores: Usuario.

    Precondiciones: Acceder a la plataforma.

    Descripción: El usuario selecciona múltiples policlínicas y accede a una vista comparativa.

    Flujo principal:

    a) El usuario selecciona dos o más policlínicas.

    b) Elige la opción Çomparar".

    c) El sistema muestra una tabla con datos relevantes: servicios, horarios, calificación, etc.

    Flujos alternativos: 1.A Si el usuario selecciona una sola, se muestra un mensaje indicando que necesita al menos dos para comparar.

    Poscondiciones: El usuario puede analizar las diferencias entre las opciones y tomar una mejor decisión.

16. **Nombre:** Cargar información de servicios y/o especialidades por parte del prestador.

    Objetivo: Permitir que los prestadores de salud carguen su oferta de servicios y/o especialidades en la plataforma.

    Actores: Administrador de prestador de salud.

    Precondiciones: Acceder a la plataforma y autenticarse como administrador del prestador.

    Descripción: El administrador accede a su panel y carga servicios y/o especialidades asociados a sus policlínicas.

    Flujo principal:

    a) El administrador accede al panel de administración.

    b) Selecciona la opción de agregar/editar servicios y/o especialidades.

    c) Completa los datos requeridos.

    d) Guarda los cambios.

    Flujos alternativos: 4.A El sistema detecta inconsistencias en los datos antes de guardarlos y notifica al administrador.

    Postcondiciones: Los servicios y/o especialidades quedan actualizados y visibles para los usuarios.

17. **Nombre:** Alta de policlínica por parte del prestador.

    Objetivo: Permitir que los administradores de un prestador de salud registren nuevas policlínicas.

    Actores: Administrador de prestador de salud.

    Precondiciones: Acceder a la plataforma y autenticarse como administrador del prestador.

    Descripción: Desde su panel de administración, el prestador puede dar de alta nuevas policlínicas que sean del prestador al cual está asignado.

    Flujo principal:

    a) El administrador inicia sesión.

    b) Accede al panel de policlínicas.

    c) Selecciona .ªgregar nueva policlínica".

    d) Completa los datos: nombre, dirección, sitio web, servicios, especialidades, ubicación en el mapa, etc.

    e) El sistema guarda la policlínica como activa.

    Flujos alternativos:

    4.A Si faltan datos obligatorios, el sistema muestra un mensaje de validación para completar los campos.

    4.B Si ya existe una policlínica registrada con la misma dirección y nombre, el sistema informa que ya está en el sistema y no permite duplicados.

    4.B.1 El administrador puede optar por editar la información existente.

    Postcondiciones: La policlínica aparece en el sistema.

18. **Nombre:** Baja de policlínica por parte del prestador.

    Objetivo: Permitir que los administradores de un prestador de salud eliminen policlínicas existentes.

    Actores: Administrador de prestador de salud.

    Precondiciones: Acceder a la plataforma y autenticarse como administrador del prestador.

    Descripción: Desde su panel de administración, el prestador puede dar de baja policlínicas existentes que sean del prestador al cual está asignado.

    Flujo principal:

    a) El administrador accede a la lista de policlínicas registradas.

    b) Selecciona una policlínica activa.

    c) Elige la opción "Dar de baja".

    d) El sistema solicita confirmación.

e) Al confirmar, la policlínica se marca como inactiva y deja de ser visible para los usuarios.

Flujos alternativos:

3.A El administrador cancela la baja y no se realizan cambios.

Postcondiciones: La policlínica ya no está disponible para los usuarios.

19. **Nombre:** Validar información ingresada por usuarios (VGI).

    Objetivo: Permitir que un administrador revise y valide datos ingresados por usuarios sobre agregar una nueva policlínica, actualizar los datos de una policlínica existente y dar de baja una policlínica que ya no existe.

    Actores: Administrador de prestador de salud.

    Precondiciones: Acceder a la plataforma y autenticarse como administrador.

    Descripción: El administrador visualiza las propuestas ingresadas por usuarios sobre policlínicas del prestador al cual el administrador está asignado, y decide si aprobarlas o rechazarlas.

    Flujo principal:

    a) El administrador accede a la sección de propuestas.

    b) Visualiza una lista de propuestas.

    c) Revisa los datos e información aportada.

    d) Aprueba o rechaza la inclusión de la nueva policlínica.

    e) El sistema envía una notificación al usuario que realizó dicha propuesta indicando si la misma fue aprobada o rechazada.

    Flujos alternativos: -

    Poscondiciones: La policlínica es aprobada y publicada, o descartada con registro del motivo.

20. **Nombre:** Revisar y gestionar reseñas de usuarios.

    Objetivo: Permitir al administrador moderar contenido enviado por los usuarios.

    Actores: Administrador de plataforma.

    Precondiciones: Acceder a la plataforma y autenticarse. Tener reseñas ingresadas por usuarios.

    Descripción: El administrador revisa las reseñas y puede eliminarlas si así lo considera en caso de que la misma contenga contenido inapropiado.

    Flujo principal:

    a) El administrador accede a la sección de gestión de reseñas.

b) Visualiza la lista de reseñas publicadas.

c) Selecciona una reseña y puede elegir eliminarla.

Flujos alternativos: 2.A No hay reseñas nuevas por revisar.

Poscondiciones: El contenido de la plataforma es moderado para garantizar calidad y respeto.

Anexo B

Testing exploratorio

B.1. Testing Exploratorio Planificado

Para la generación de los casos de prueba, se tomó en cuenta el Documento de casos de Uso creado para el proyecto.

Para la generación de las pruebas se tienen los usuarios:

- adminPlataforma@mail.com (Administrador de Plataforma).
- adminPrestador@mail.com (Administrador de Prestador).
- test@test.com (Usuario normal)

A continuación se describen los distintos casos de prueba desarrollados:

B.1.1. Registro de usuario

Registro de usuario con email y contraseña válidos.

Descripción El caso de prueba pretende validar el registro de un usuario en el sistema con un email bien formado y una contraseña válida.

Actores Usuarios sin cuenta registrada.

Resultados esperados Se espera que el usuario se cree en la base de datos con sus respectivos datos y además al finalizar el registro el usuario, dicho usuario debe estar logueado en la aplicación.

Éxito ✓

Defectos encontrados Ninguno.

Registro de usuario con email y contraseña inválidos

Descripción El caso de prueba pretende mostrar lo que ocurre cuando un usuario ingresa ya sea un email mal formado o una contraseña inválida (menor a 8 caracteres).

Actores Usuarios sin cuenta registrada.

Resultados esperados Se espera que el sistema le muestre un error al usuario en los distintos casos.

Éxito ✓

Defectos encontrados Ninguno.

B.1.2. Logueo del usuario

Login del usuario con credenciales correctas

Descripción El caso de prueba pretende validar el inicio de sesión de un usuario previamente registrado en el sistema al ingresar las credenciales correctamente.

Actores Cualquier usuario con cuenta registrada.

Resultados esperados Se espera que el usuario sea redirigido a la página principal del sistema y quede logueado en el mismo.

Éxito ✓

Defectos encontrados Ninguno.

Login del usuario con credenciales incorrectas

Descripción El caso de prueba muestra lo que ocurre cuando un usuario quiere iniciar sesión con las credenciales incorrectas.

Actores Cualquier usuario con cuenta registrada.

Resultados esperados Se espera que el sistema le muestre un error común al usuario en ambos casos, ya sea para un mail no registrado en el sistema o para una contraseña incorrecta.

Éxito ✓

Defectos encontrados Ninguno.

B.1.3. Logout del usuario

Logout del usuario

Descripción El caso de prueba pretende mostrar lo que ocurre cuando un usuario cierra su sesión.

Actores Cualquier usuario con cuenta registrada.

Resultados esperados Se espera que la sesión del usuario se acabe y el mismo ya no esté logueado en la aplicación, además de ser redirigido a la página principal.

Éxito ✓

Defectos encontrados Se detectó un error en la configuración de la cookie del usuario. Dicho error también afectaba a otros casos como Cambio de contraseña del usuario.

B.1.4. Gestión de contraseña

Cambio de contraseña del usuario

Descripción El caso de prueba valida el cambio de contraseña de un usuario.

Actores Cualquier usuario con cuenta registrada.

Resultados esperados Se espera que se actualice la nueva contraseña del usuario en la base de datos y que el usuario sea capaz de iniciar sesión con la nueva contraseña la próxima vez que inicie sesión.

Éxito ✓

Defectos encontrados Ninguno. (se corrigieron en el caso de prueba Logout del usuario)

B.1.5. Casos de prueba para un usuario común con cuenta registrada

Agregar policlínica como favorito

Descripción El caso de prueba muestra como un usuario puede agregar policlínicas a favoritos para luego poder visualizarlas en su perfil.

Resultados esperados Se espera que el usuario tenga asignado como favorito a una o varias policlínicas y la que la próxima vez que acceda a su perfil, pueda visualizar cuales son sus policlínicas favoritas. Además también puede sacarla de favorito si lo desea directamente desde su perfil.

Éxito ✓

Defectos encontrados Ninguno.

Comparar policlínicas

Descripción El caso de prueba muestra la comparación de hasta 3 policlínicas del mismo prestador o de distintos.

Resultados esperados Se espera que el usuario pueda visualizar en el panel de comparación los detalles básicos de hasta 3 policlínicas sin importar el prestador al que pertenezcan y también pueda ver más detalles de las mismas. Finalmente puede limpiar el panel de comparación completo o solo quitar las policlínicas que desee.

Éxito ✓

Defectos encontrados Si no se está logueado y se hacen comparaciones, cuando se inicia sesión esas comparaciones siguen estando. Lo mismo si se está logueado y se desloguea para loguearse con otro usuario distinto, la comparación del usuario anterior queda guardada para ese otro usuario.

Ingresar una reseña sobre una policlínica

Descripción El caso muestra el ingreso de reseñas por parte del usuario hacia una policlínica.

Resultados esperados El usuario ingresa una reseña para una policlínica en cuestión y dicha reseña luego queda visible para todo el mundo. En la reseña se muestra el nombre del usuario que la realizó.

Éxito ✓

Defectos encontrados Ninguno.

Ingresar una nueva policlínica por parte de un usuario

Descripción El caso muestra cómo se ingresa un pedido para crear una nueva policlínica de un prestador de salud existente por parte del usuario.

Resultados esperados Se crea un pedido para crear una nueva policlínica y el pedido queda en espera hasta que un administrador del prestador del cual se agregó el pedido, acepte o rechace el mismo.

Éxito ✓

Defectos encontrados Ninguno.

Proponer edición a los datos de una policlínica por parte de un usuario

Descripción El caso muestra cómo se ingresa una propuesta por parte del usuario para cambiar los datos de una policlínica existente en el sistema.

Resultados esperados Se crea un pedido para cambiar los datos de una policlínica existente y el pedido queda en espera hasta que un administrador del prestador del cual se agregó el pedido, acepte o rechace el mismo.

Éxito ✓

Defectos encontrados En la visualización de detalles de la policlínica, al editar los input de nombre y dirección, cuando se clickea la ubicación en el mapa, se resetean los campos editados anteriormente con sus valores originales.

Proponer cierre de policlínica por parte de un usuario

Descripción El caso muestra cómo se ingresa una propuesta por parte del usuario para indicar el cierre de una policlínica existente en el sistema que ya no existe en la realidad.

Resultados esperados Se crea un pedido para eliminar una policlínica existente y el pedido queda en espera hasta que un administrador del prestador del cual se agregó el pedido, acepte o rechace el mismo.

Éxito ✓

Defectos encontrados Si se escribe algo en el formulario para Reportar Cierre de Policlínica, cuando se clickea afuera del formulario o se selecciona en cancelar, al volver a seleccionar el Reporte de cierre, aparece lo que estaba escrito antes

B.2. Testing Exploratorio Basado en Sesiones

Para realizar testing exploratorio basado en sesiones, se definieron cuatro sesiones, las cuales son descritas más adelante. Es de interés para todas las

sesiones determinar los siguientes tipos de errores:

- Errores de navegación.
- Errores de ortografía.
- Errores de tamaño de campos.

B.2.1. Sesión 1: Experiencia de Usuario invitado

Objetivo: Se desea probar la experiencia de un usuario invitado en la aplicación Salud en Cercanía. Esta sesión tiene como objetivo conocer todas las funcionalidades que tiene disponible la aplicación cuando no existe usuario logueado en el sistema.
Tester Todos.
Tiempo 10 minutos.
Defectos encontrados Ninguno
Acciones realizadas:

1. Se selecciona en Ingresar y se verifica que el usuario no está logueado
2. Se navega nuevamente al menú principal
3. Se selecciona en Buscar Prestadores
4. Se selecciona en Ver Policlínicas de un prestador de salud
5. Se selecciona en Comparar esa policlínica
6. Se selecciona en Ver Comparación
7. Se selecciona en Ver Detalles Completos
8. Se selecciona en Volver
9. Se selecciona en Ver Policlínicas de otro prestador de salud
10. Se selecciona en Comparar esa policlínica
11. Se selecciona en Ver Comparación
12. Se selecciona en Limpiar Comparación
13. Se selecciona en Ir a Visualizar
14. Se selecciona nuevamente en Ver Policlínicas de un prestador de salud seleccionado
15. Se selecciona en Ver Detalles de una policlínica
16. Se seleccionan sobre las Especialidades de esa policlínica
17. Se selecciona sobre los Servicios de esa policlínica

18. Se selecciona sobre las Reseñas de esa policlínica
19. Se selecciona en Volver para navegar nuevamente a la sección de buscar prestadores de salud
20. Fin de la sesión

B.2.2. Sesión 2: Experiencia de Usuario logueado

Objetivo: Se desea probar la experiencia de un usuario que se registra e ingresa en la aplicación Salud en Cercanía. Esta sesión tiene como objetivo conocer todas las funcionalidades que tiene disponible la aplicación cuando existe un usuario común logueado en el sistema.
Tester Todos.
Tiempo 15 minutos.
Defectos encontrados Ninguno
Acciones realizadas:

1. Se selecciona en Ingresar
2. Se selecciona en Registrarse
3. Se completan los datos solicitados, se aceptan los términos y condiciones y luego se clickea en Crear Cuenta.
4. Se accede a la página principal con el usuario logueado.
5. Se clickea en el icono del usuario logueado en la parte superior derecha y se selecciona en Cerrar Sesión
6. Se clickea en Ingresar y se vuelve a iniciar sesión con las credenciales recién creadas
7. Se clickea en el icono del usuario logueado en la parte superior derecha y se selecciona en Configuración.
8. Se selecciona en Cambiar Contraseña
9. Se ingresa la contraseña actual y la contraseña nueva y se selecciona en Cambiar Contraseña
10. Se clickea en el icono del usuario logueado en la parte superior derecha y se selecciona en Cerrar Sesión
11. Se selecciona Ingresar y se ingresa con la nueva contraseña
12. Se selecciona en Buscar Prestadores
13. Se selecciona en Ver Policlínicas de un prestador
14. Se marca como favorita una policlínica

15. Se selecciona en Mi Perfil y se verifica que la policlínica aparece en la lista de Policlínicas Favoritas

16. Se navega nuevamente al inicio

17. Se selecciona en Buscar Prestadores

18. Se selecciona en Ver Policlínicas de un prestador

19. Se selecciona en Agregar Nueva Policlínica, se ingresan todos los datos (Nombre, dirección y se marca la ubicación en el mapa) y finalmente se selecciona Enviar Propuesta

20. Se selecciona Volver a Visualizar

21. Se selecciona en Ver Policlínicas de un prestador

22. Se selecciona en Ver Detalles de una policlínica

23. Se selecciona en Proponer Edición, se completan los datos y se selecciona en Enviar Propuesta

24. Se selecciona en Volver

25. Se selecciona en Ver Policlínicas de un prestador

26. Se selecciona en Ver Detalles de una policlínica

27. Se selecciona en Reportar Cierre de Policlínica, se ingresa el motivo por el cual se está reportando su cierre y finalmente se selecciona en Confirmar Reporte

28. Se selecciona en Volver

29. Se selecciona en Ver Policlínicas de un prestador

30. Se selecciona en Ver Detalles de una policlínica

31. Se selecciona en Reseñas

32. Se selecciona en Escribir Reseña, se escribe una reseña para esa policlínica y finalmente se selecciona en Publicar Reseña

33. Se clickea en el icono del usuario logueado en la parte superior derecha y se selecciona en Cerrar Sesión

34. Fin de la sesión.

B.2.3. Sesión 3: Experiencia de un Administrador de Prestador

Objetivo: Se desea probar todas las funcionalidades que posee el sistema cuando existe un administrador de prestador logueado en el mismo.
Tester Todos.
Tiempo 15 minutos.
Defectos encontrados Ninguno
Acciones realizadas:

1. Se selecciona en Ingresar y se inicia sesión con las credenciales del Administrador de Prestador

2. Se clickea en el icono del usuario logueado en la parte superior derecha y se selecciona en Gestionar Policlínicas

3. Se selecciona en Propuestas y se Aprueba o Rechaza las propuestas del usuario según sean correctas o no

4. Se selecciona en Agregar Policlínica se ingresan los datos de la nueva policlínica a agregar y se selecciona en Agregar y se vuelve automáticamente a la sección de Panel de gestión

5. Se selecciona Ver Detalles en alguna de las policlínicas

6. Se selecciona en Servicios y se selecciona en Editar

7. Se ingresan los datos del nuevo servicio y se selecciona en Agregar

8. Se vuelve a seleccionar en Editar de servicios y se selecciona en la cruz para eliminar el servicio

9. Se selecciona en Volver

10. Se clickea en el icono del usuario logueado en la parte superior derecha y se selecciona en Gestionar Policlínicas

11. Se selecciona Editar en alguna de las policlínicas

12. Se editan los cambios y se selecciona en Guardar Cambios

13. Se selecciona Eliminar en alguna de las policlínicas

14. Se clickea en el icono del usuario logueado en la parte superior derecha y se selecciona en Cerrar Sesión

15. Fin de la sesión

B.2.4. Sesión 4: Experiencia de un administrador de plataforma

Objetivo: Se desea probar todas las funcionalidades que posee el sistema cuando existe un administrador de plataforma logueado en el mismo.
Tester Todos.
Tiempo 5 minutos.
Defectos encontrados Ninguno
Acciones realizadas:

1. Se selecciona en Ingresar y se inicia sesión con las credenciales del Administrador de Plataforma
2. Se clickea en el icono del usuario logueado en la parte superior derecha y se selecciona en Panel de Administración
3. Se selecciona en Reseñas y se selecciona en Eliminar la reseña que se desea moderar
4. Se clickea en el icono del usuario logueado en la parte superior derecha y se selecciona en Cerrar Sesión
5. Fin de la sesión

Anexo C

Manual de usuario

C.1. Introducción

El siguiente documento tiene como objetivo mostrar en detalle las funcionalidades de la web *Salud en Cercanía*. Se organiza en cinco secciones, la segunda sección describe las funcionalidades disponibles para usuarios invitados, la tercera sección corresponde a los usuarios comunes con cuenta registrada en la aplicación, la cuarta incluye las funcionalidades de los administradores de prestador, y finalmente la quinta sección muestra las funcionalidades disponibles para un administrador de plataforma.

C.2. Usuario invitado

Las funcionalidades disponibles para un usuario invitado son muy limitadas. Al ingresar al sitio, se accede a una página principal que se muestra en la figura C.1, dentro de la misma un usuario común puede clickear en *Buscar Prestadores* o en *Visualizar* como se observa en la figura y ambas lo redirigen a la sección para visualizar prestadores de salud.

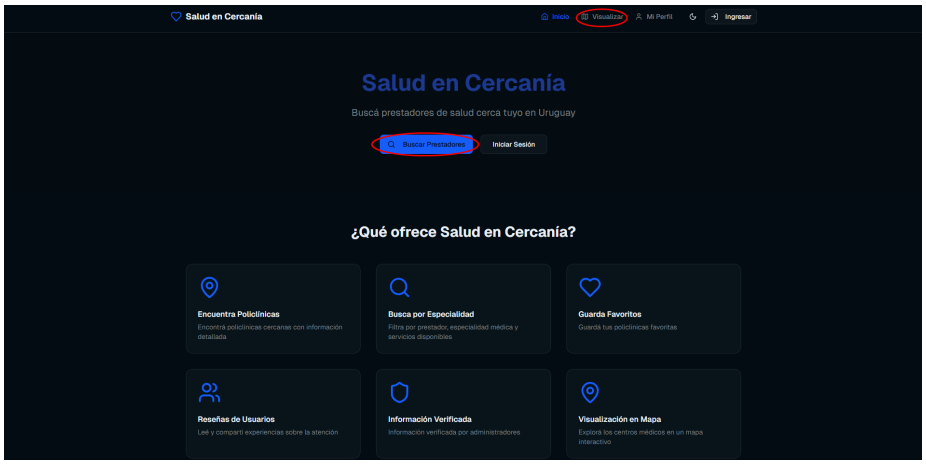


Figura C.1: Página principal de Salud en Cercanía

Dentro de la sección de Visualizar Prestadores se listan todos los prestadores de salud disponibles en el sistema, el usuario invitado puede seleccionar un prestador y visualizar todas sus policlínicas. Para realizar lo anterior, hay distintas formas de lograrlo, como se muestra en la figura C.2. Es posible visualizar todas las policlínicas de todos los prestadores en el sistema directamente desde el mapa accediendo a la sección *Ver en Mapa*, como se puede observar en la figura C.3.

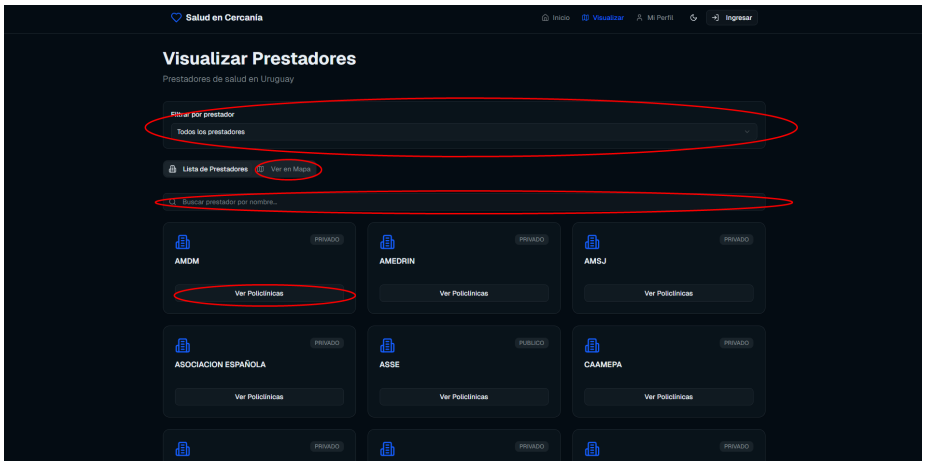


Figura C.2: Salud en Cercanía - Visualizar Prestadores

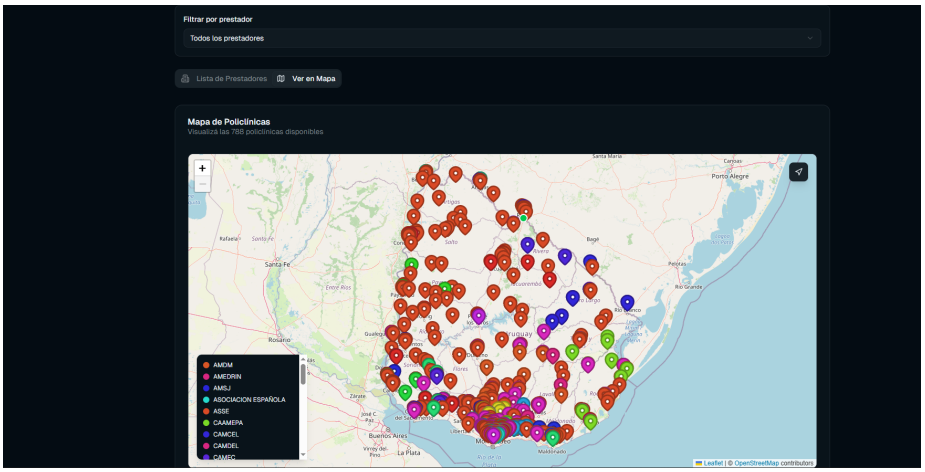


Figura C.3: Salud en Cercanía - Visualizar Policlínicas en el Mapa

Volviendo a la figura C.2, se observa que es posible poder filtrar por prestador o directamente escribir en el buscador de la aplicación el nombre del prestador, en caso de no estar disponible a la vista para el usuario.

Al seleccionar en *Ver Policlínicas* de algún prestador listado, o clickear directamente en algún prestador en el filtro por prestador, se accede a la vista de las policlínicas del prestador que fue seleccionado como se puede observar en la figura C.4.

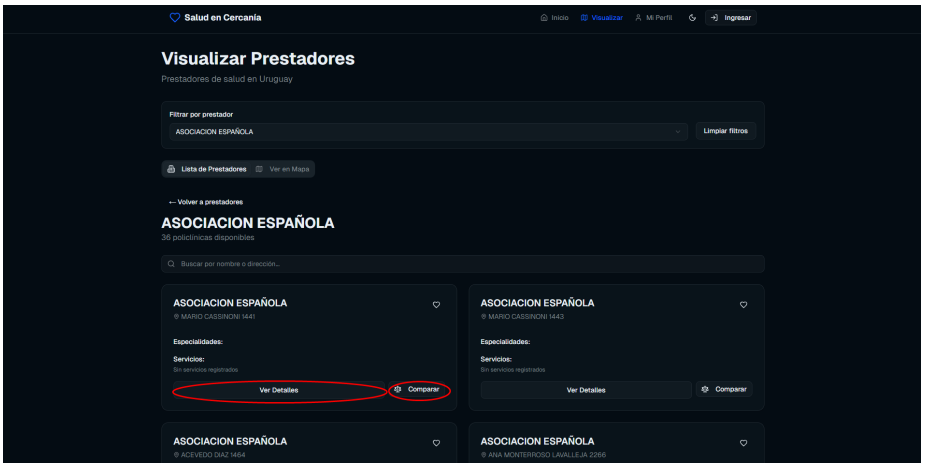


Figura C.4: Salud en Cercanía - Visualizar Policlínicas del Prestador Seleccionado

Dentro de esta sección un usuario puede clickear en *Ver en el mapa* para visualizar las policlínicas del prestador directamente en el mapa (figura C.5). También puede seleccionar en *Ver Detalles* de alguna policlínica para ver la información detallada de la misma, como se puede ver en la figura C.6.

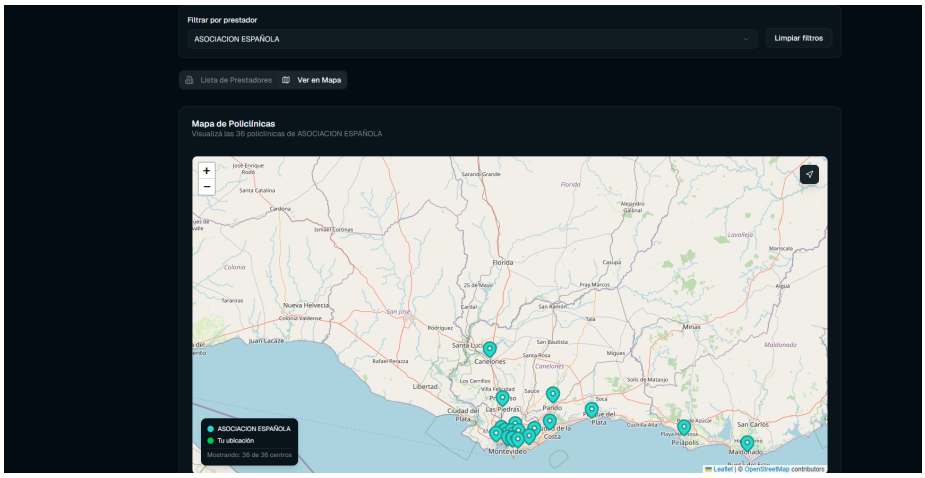


Figura C.5: Salud en Cercanía - Visualizar Policlínicas del Prestador Seleccionado en el Mapa

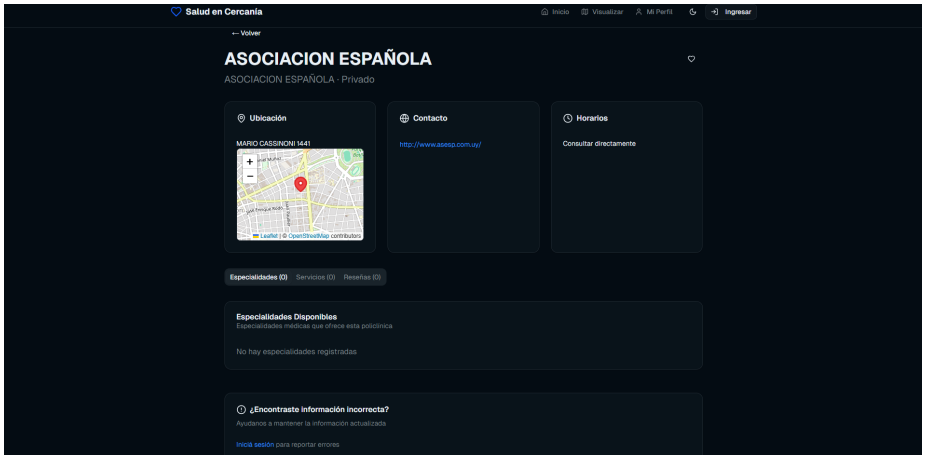


Figura C.6: Salud en Cercanía - Ver Detalles Policlínica

Finalmente un usuario puede seleccionar en *Comparar* alguna policlínica para poder comparar hasta tres policlínicas de cualquier prestador y poder visualizar los detalles en paralelo, esto se puede ver en la figura C.7 y figura C.8,

donde se comparan dos policlínicas de la Asociación Española y una de CASMU

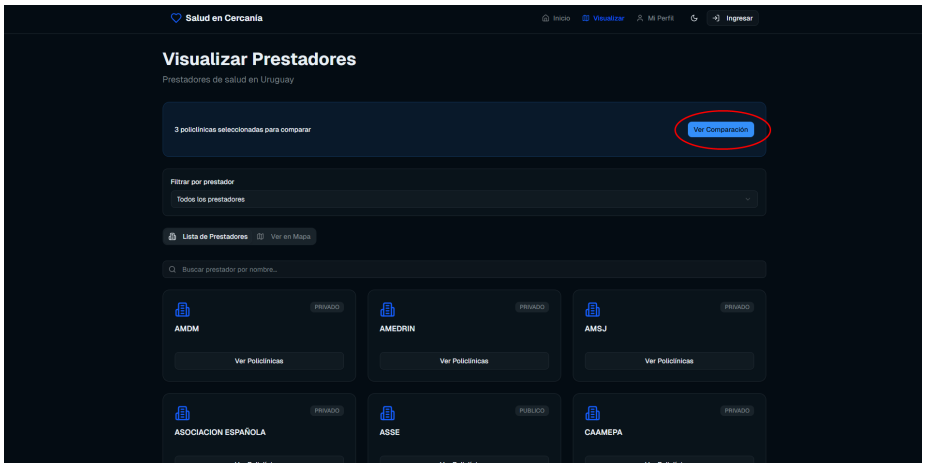


Figura C.7: Salud en Cercanía - Ver Comparación

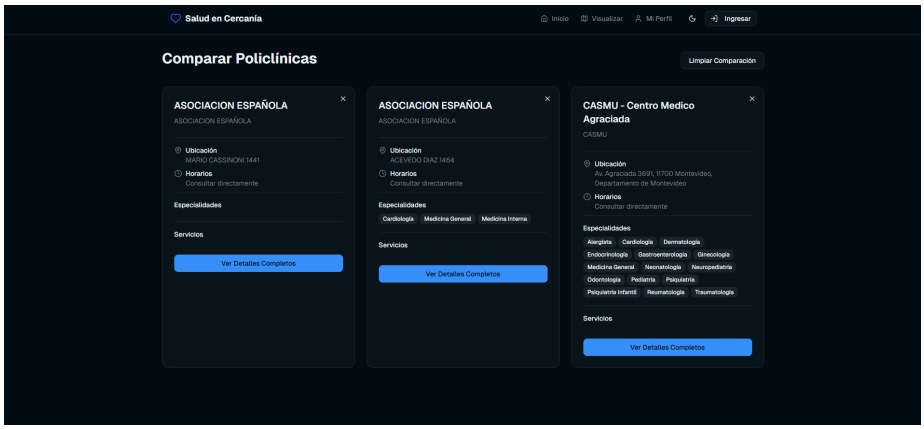


Figura C.8: Salud en Cercanía - Comparar Policlínicas

C.3. Usuario común registrado en la aplicación

Para poder ingresar con un usuario registrado, se debe clickear en *Ingresar* desde la página principal de la aplicación (figura C.9).

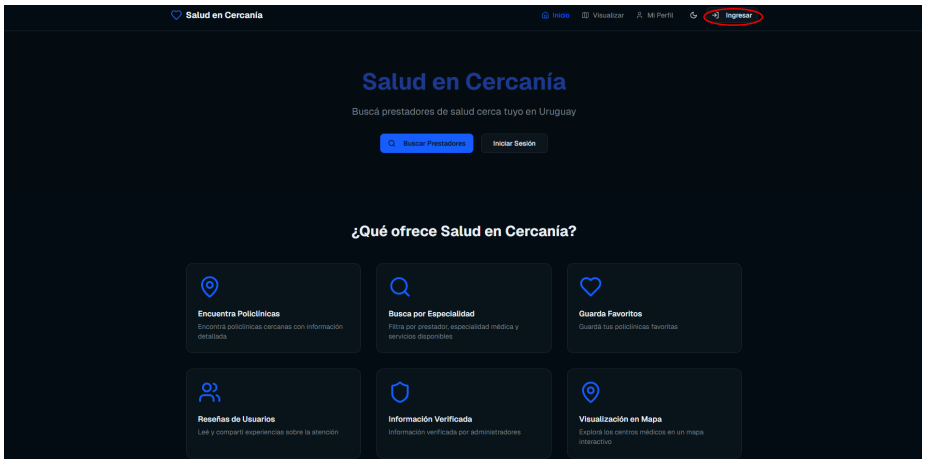


Figura C.9: Salud en Cercanía - Ingresar

Al ingresar se accede directamente al formulario para iniciar sesión como se observa en la figura C.10, en caso de no estar registrado también se puede acceder al formulario de registro, el cual se muestra en la figura C.11.

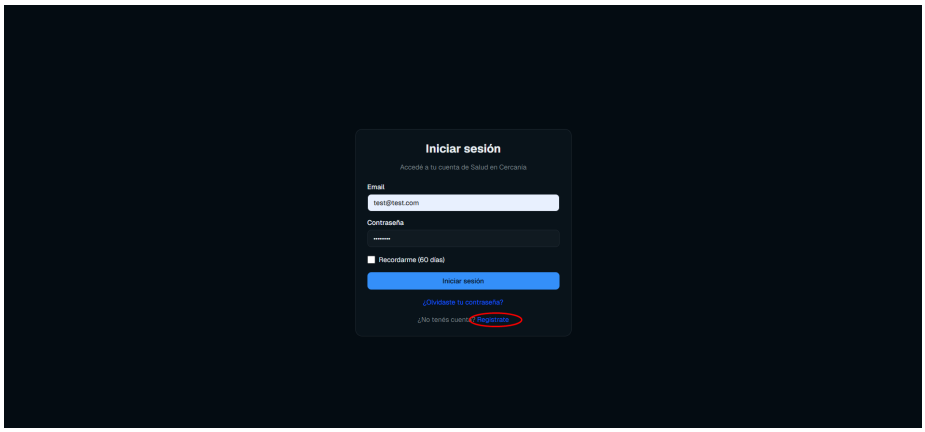


Figura C.10: Salud en Cercanía - Iniciar Sesión

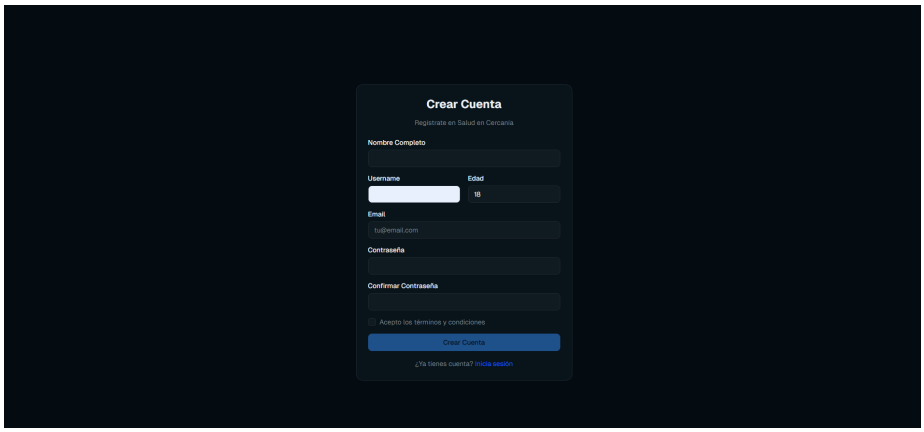


Figura C.11: Salud en Cercanía - Crear Cuenta

Al iniciar sesión, un usuario registrado cuenta con muchas más funcionalidades nuevas, además de contar con todas las anteriores de un usuario invitado.

Una de las funcionalidades que se desbloquean al iniciar sesión con una cuenta es la de tener un perfil y un panel de configuración (figura C.12).

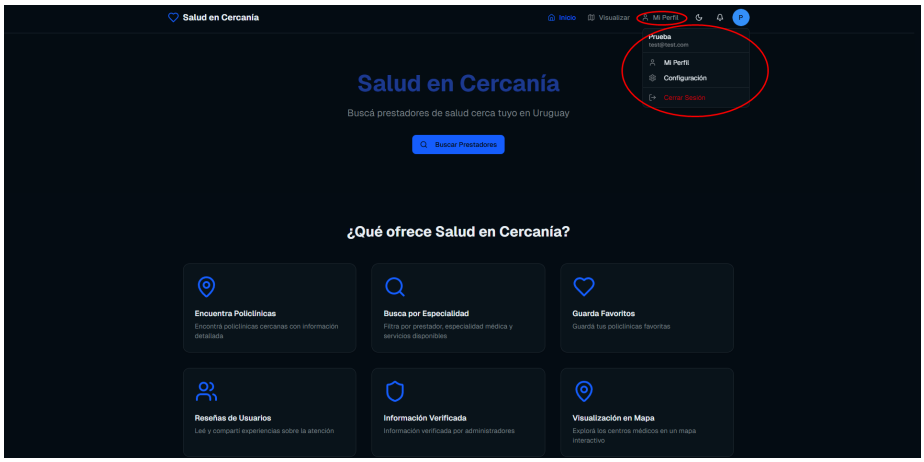


Figura C.12: Salud en Cercanía - Panel de Usuario

Al acceder a *Mi Perfil* se muestra el perfil del usuario, donde se muestran algunas de sus estadísticas como sus reseñas o las policlínicas que el usuario añadió como favorito, esto se puede ver en la figura C.13.

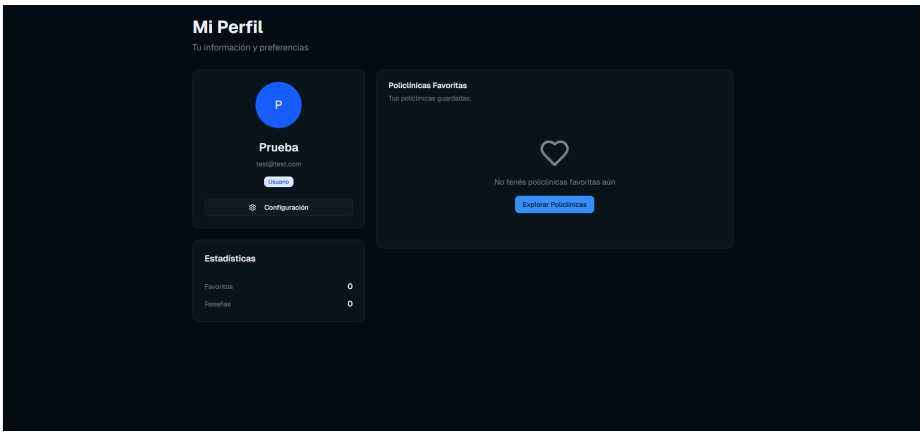


Figura C.13: Salud en Cercanía - Perfil del Usuario

Además, en la figura C.14 se observa el panel de configuración del usuario donde se muestran sus datos y también es posible cambiar la contraseña.

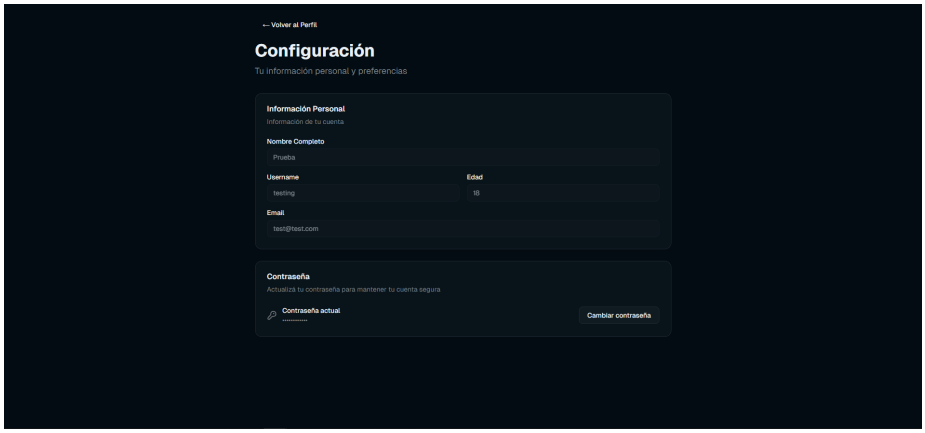


Figura C.14: Salud en Cercanía - Panel de Configuración

Otras funcionalidades que desbloquea un usuario registrado son las de VGI (agregar una nueva policlínica, editar los datos de una policlínica existente y eliminar una policlínica, las mismas se muestran en las figuras C.15, C.16 y C.17 respectivamente), las mismas deben pasar antes por un período de aprobación por parte de un administrador de prestador.

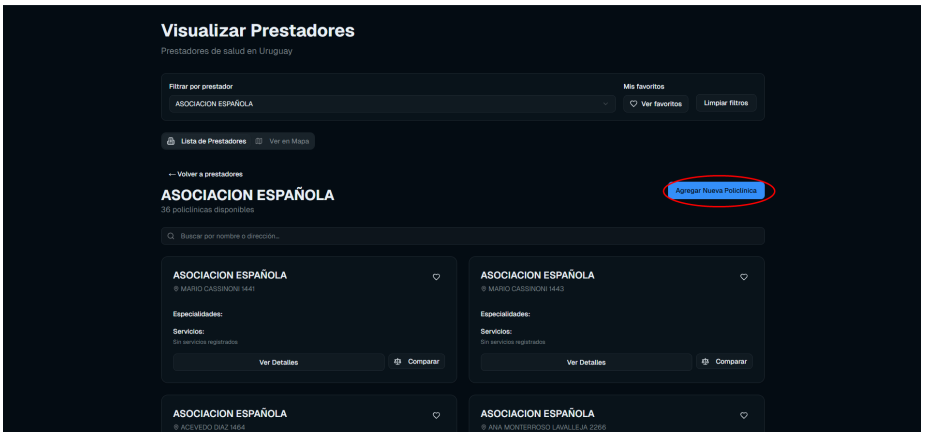


Figura C.15: Salud en Cercanía - Agregar Nueva Policlínica

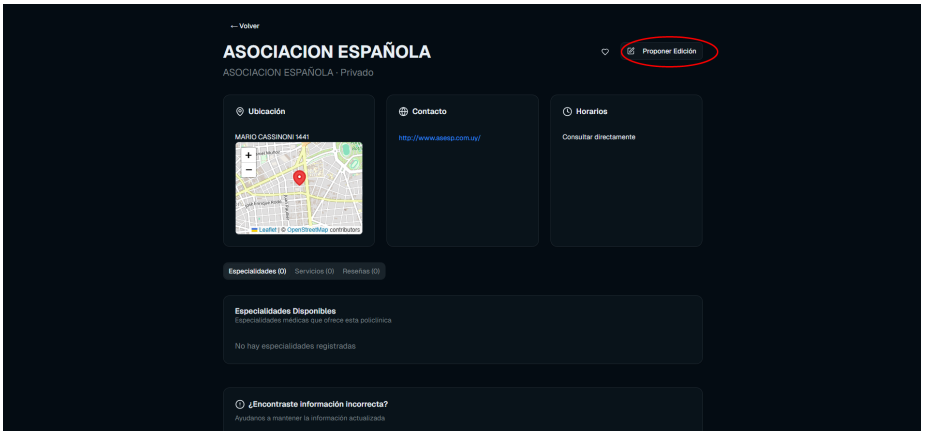


Figura C.16: Salud en Cercanía - Proponer Edición

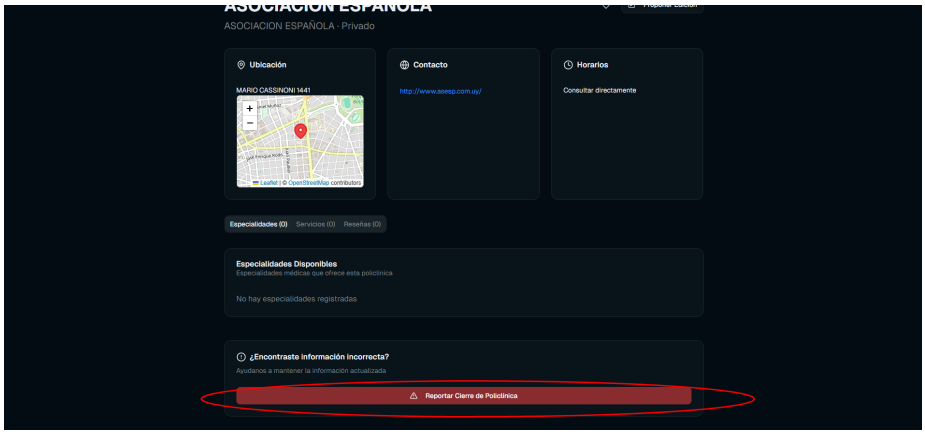


Figura C.17: Salud en Cercanía - Reportar Cierre Policlínica

En las figuras C.18, C.19 y C.20 se muestran los formularios para agregar una nueva policlínica, editar los datos de una policlínica existente y eliminar una policlínica respectivamente.

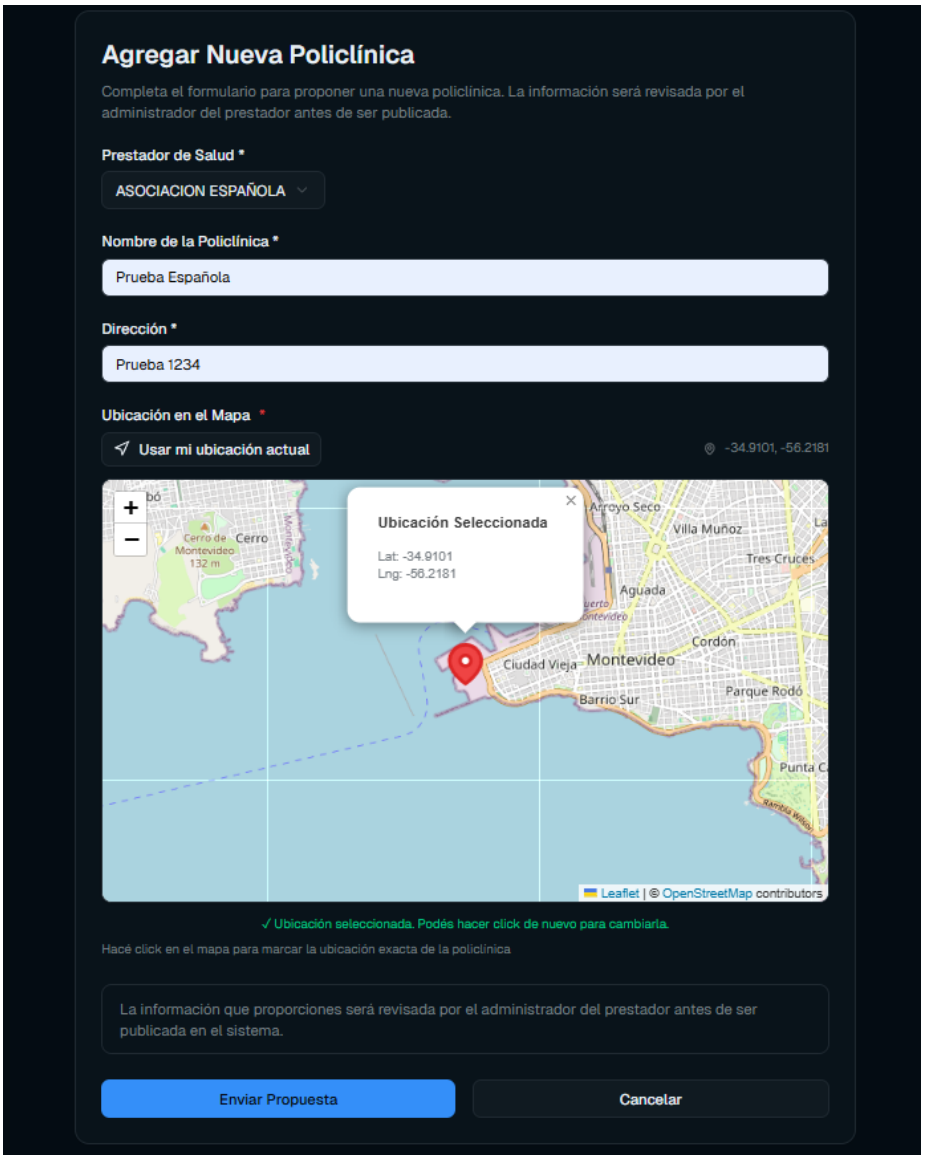


Figura C.18: Salud en Cercanía - Formulario Agregar Nueva Policlínica

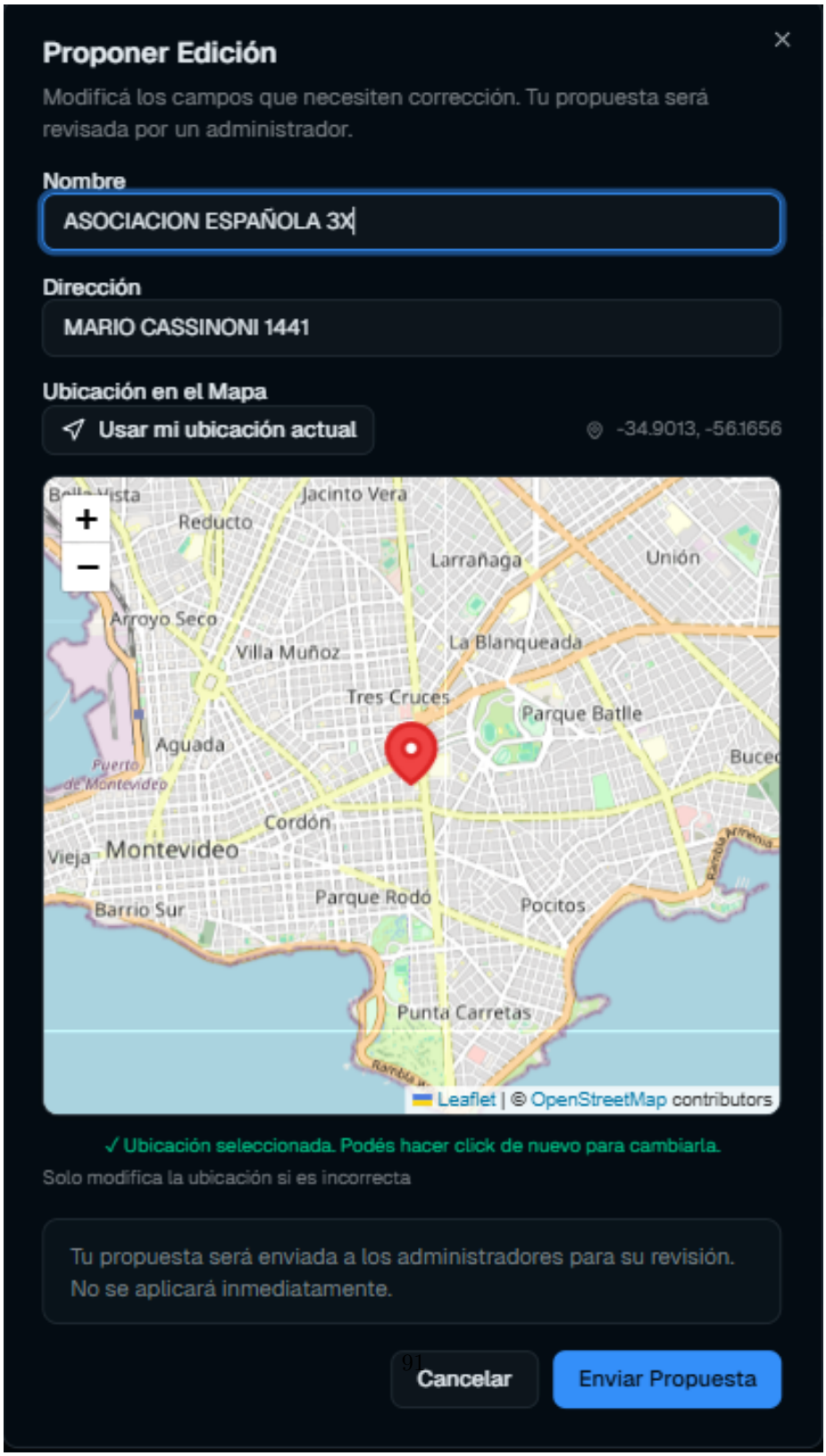


Figura C.19: Salud en Cercanía - Formulario Proponer Edición



Figura C.20: Salud en Cercanía - Formulario Reportar Cierre Policlínica

Finalmente, en las figuras C.21 y C.22 se observa como un usuario puede realizar una reseña para una policlínica.

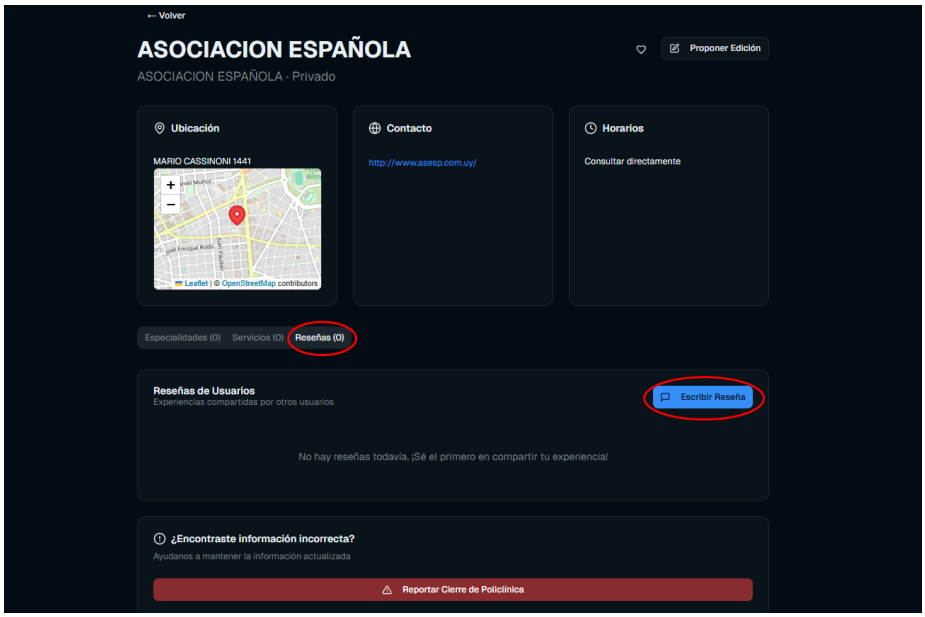


Figura C.21: Salud en Cercanía - Escribir Reseña

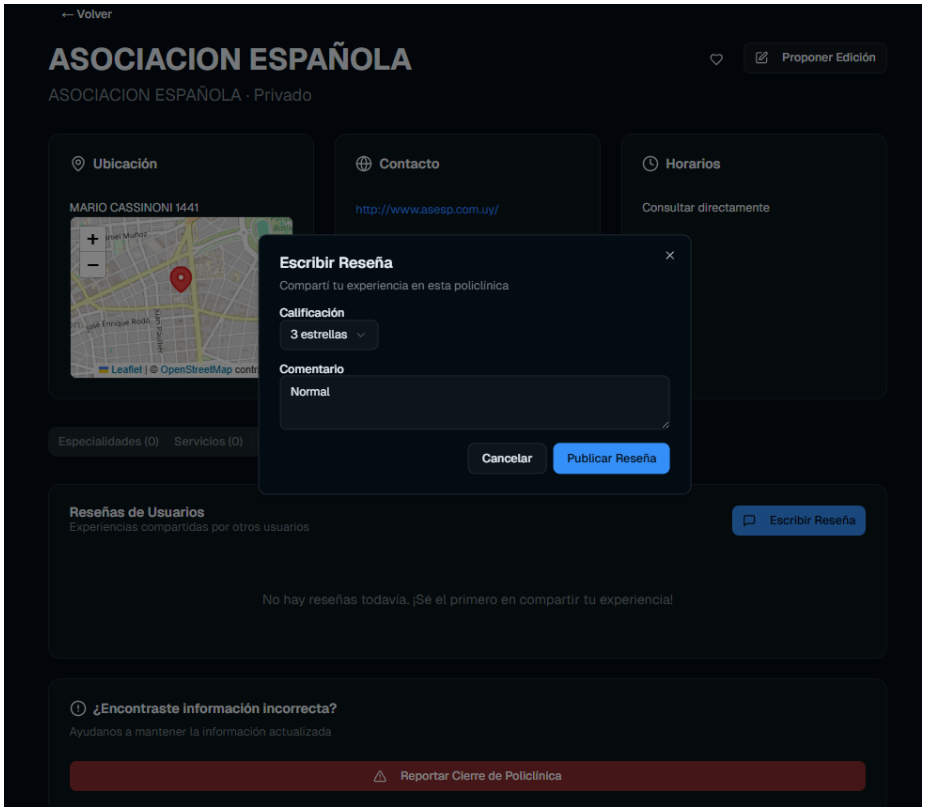


Figura C.22: Salud en Cercanía - Formulario Escribir Reseña

C.4. Administrador de Prestador

Un Administrador de Prestador, a diferencia de los demás usuarios tiene un panel para gestionar policlínicas, al cual se puede acceder desde su perfil (figura C.23).

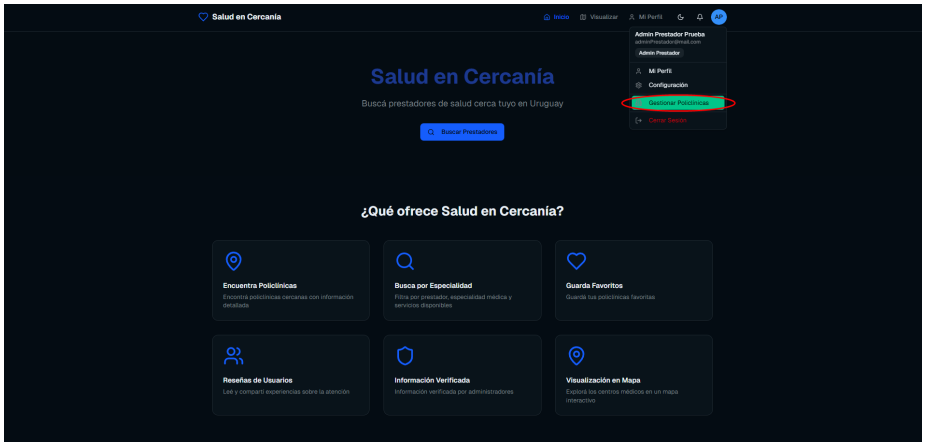


Figura C.23: Salud en Cercanía - Gestionar Policlínicas

Dentro de este panel, el Administrador de Prestador puede dar de alta, de baja o editar los datos de una policlínica directamente, sin tener que pasar por ningún período de aprobación. En la figura C.24 se puede observar el panel de administración, también se puede ver la sección de Propuestas, la cual muestra todos los pedidos VGI por parte de los usuarios comunes que deben ser aprobados o rechazados, esto se observa con más detalle en la figura C.25.

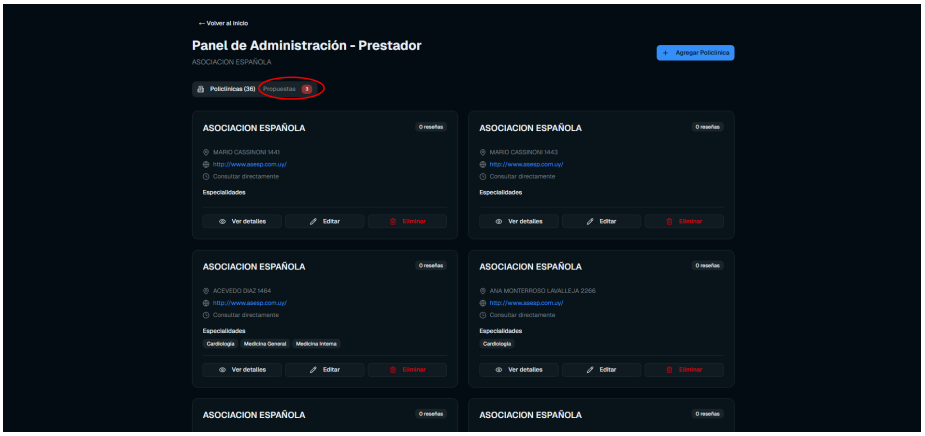


Figura C.24: Salud en Cercanía - Panel de Administrador de Prestador

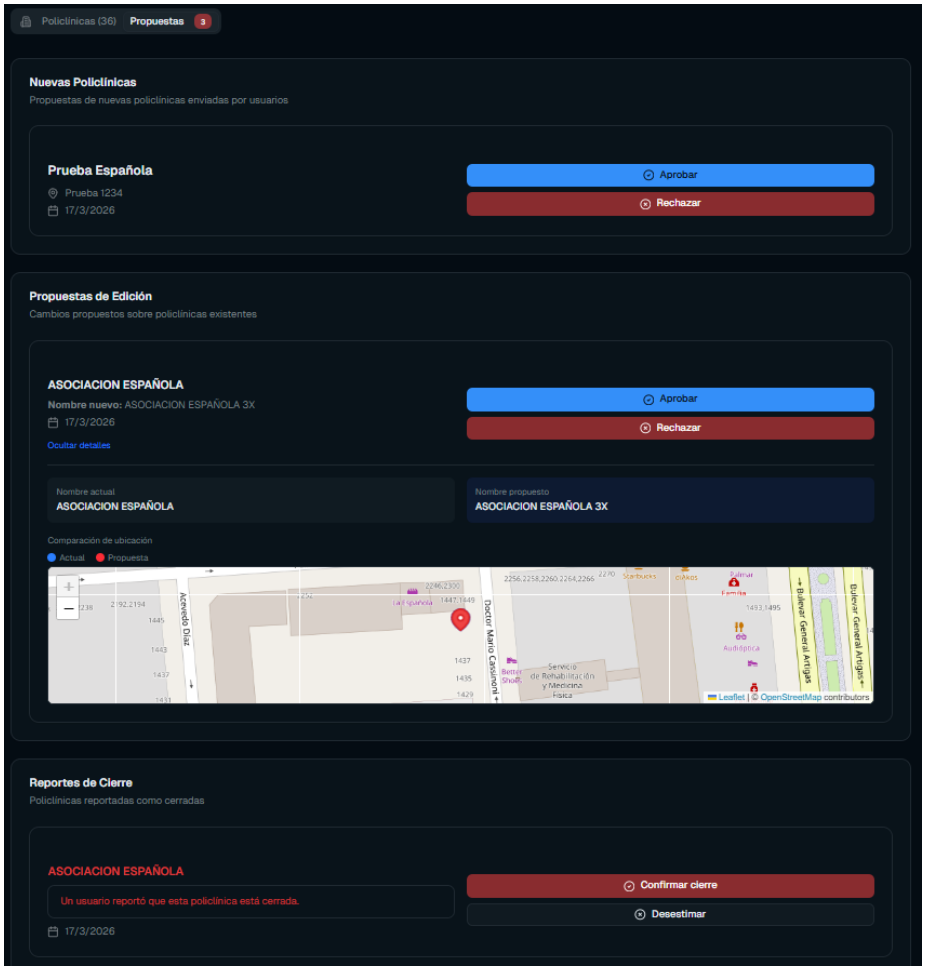


Figura C.25: Salud en Cercanía - Propuestas VGI por parte de los usuarios

C.5. Administrador de Plataforma

Finalmente un Administrador de Plataforma, de forma similar al Administrador de Prestador, cuenta con un panel de administración (figura C.26).

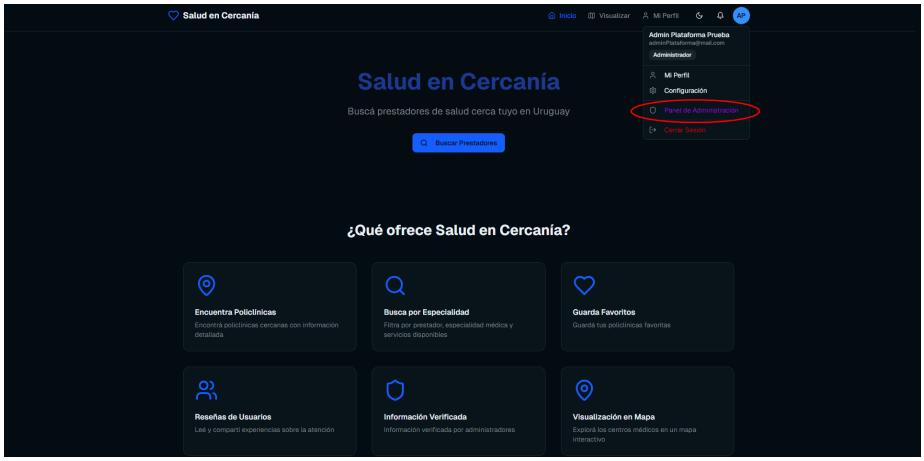


Figura C.26: Salud en Cercanía - Panel de Administración

Dentro del panel de administración se pueden ver algunas estadísticas como se observa en la figura C.27, además es posible acceder a la sección de Reseñas para poder moderar las reseñas ingresadas por los usuarios. En la figura C.28 se observa la sección de reseñas por parte de los usuarios.

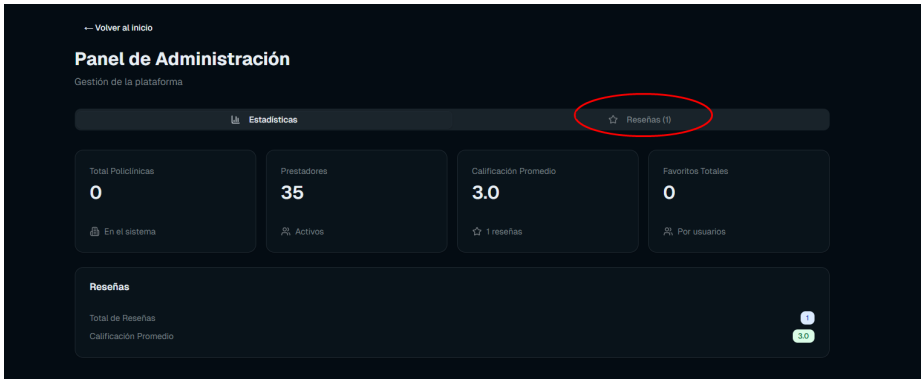


Figura C.27: Salud en Cercanía - Estadísticas Generales de la Plataforma

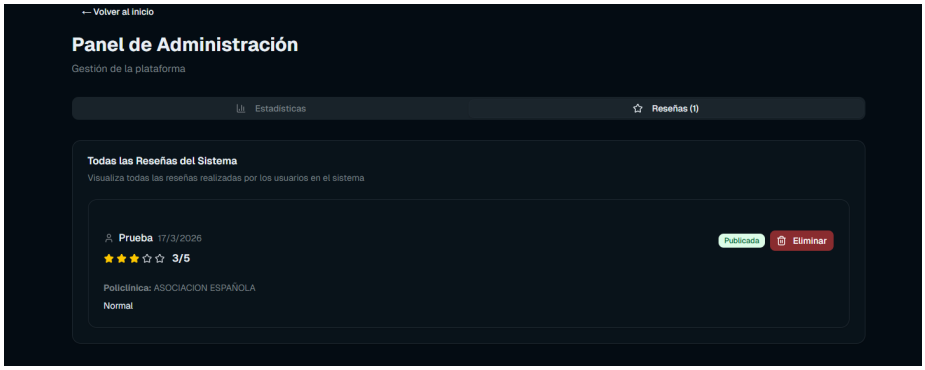


Figura C.28: Salud en Cercanía - Moderar Reseñar de Usuarios