CALIFICACIÓN LIA CARVAJAL MEDEROS: *9.0*
CALIFICACIÓN ALBERTO GÓMEZ DE ANDRÉS: *9.0*

DESARROLLO DE UN SISTEMA DE GESTIÓN DE INFORMACIÓN DE PACIENTES INGRESADOS EN UN HOSPITAL

DEVELOPMENT OF AN INFORMATION MANAGEMENT SYSTEM FOR PATIENTS ADMITTED TO A HOSPITAL

TRABAJO FIN DE GRADO
CURSO 2024-2025

AUTORES
LIA CARVAJAL MEDEROS
ALBERTO GÓMEZ DE ANDRÉS

DIRECTOR
ANTONIO SARASA CABEZUELO

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

DESARROLLO DE UN SISTEMA DE GESTIÓN DE INFORMACIÓN DE PACIENTES INGRESADOS EN UN HOSPITAL

DEVELOPMENT OF AN INFORMATION MANAGEMENT SYSTEM FOR PATIENTS ADMITTED TO A HOSPITAL

TRABAJO DE FIN DE GRADO DE INGENIERÍA INFORMÁTICA

AUTORES
LIA CARVAJAL MEDEROS
ALBERTO GÓMEZ DE ANDRÉS

DIRECTOR
ANTONIO SARASA CABEZUELO

CONVOCATORIA: JUNIO 2025

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

13 MAYO DE 2025

DEDICATORIA

A nuestra familia, por su apoyo constante
e incondicional, confiar en nosotros y
respaldarnos durante este proceso.

AGRADECIMIENTOS

Llegado este momento, queremos agradecer a nuestro tutor, Antonio Sarasa Cabezuelo, por habernos guiado y aconsejado desde el inicio de nuestro recorrido en este proyecto.

También queremos ampliar nuestros agradecimientos a nuestros familiares, amigos y compañeros de universidad por su paciencia y comprensión, haciéndonos este camino más llevadero.

RESUMEN

Desarrollo de un sistema de gestión de información de pacientes ingresados en un hospital

La administración de la información clínica de los pacientes en los hospitales es uno de los principales retos que tiene el personal sanitario debido al proceso manual de anotación y registro de tratamientos, constantes vitales, incidencias... Por ello mismo, para poder garantizar la seguridad de los datos del paciente y optimizar la labor de los sanitarios, es necesario adoptar ciertas soluciones que integren las nuevas tecnologías para llevar a cabo una gestión centralizada y en tiempo real de la información.

Este proyecto consiste en una aplicación móvil cuya finalidad es dar apoyo al personal sanitario a la hora de asistir a los pacientes ingresados. La solución presentada centraliza los datos clínicos, registra automáticamente los parámetros de seguimiento y envía notificaciones a los dispositivos de los sanitarios para que estén actualizados sobre cualquier cambio realizado.

La implementación de este sistema facilita la coordinación entre médicos y enfermeros, alivia la carga de trabajo y disminuye la posibilidad de errores en los procesos manuales. Además, representa un avance en la transformación digital de la gestión hospitalaria, ya que mejora la precisión y optimiza todos los procesos, promoviendo una atención médica más segura y eficiente.

Palabras clave

Centro sanitario, personal sanitario, médico, enfermero, paciente, tratamiento, registros clínicos, gestión de la información, coordinación sanitaria, aplicación móvil.

ABSTRACT

Development of an information management system for patients admitted to a hospital

The management of patient's clinical information in hospital is one of the main challenges for healthcare professionals due to the manual recording of treatments, vital signs, incidents… Therefore, in order to guarantee patient data safety and optimize the healthcare personnel's work, it is necessary to adopt certain solutions that integrate new technologies for centralized, real-time information management.

This project consists of a mobile application designed to support healthcare staff in the care of hospitalised patients. The solution presented centralises clinical data, automatically records monitoring parameters and sends notifications to the healthcare professional's devices so that they are kept up to date with any changes done.

The implementation of this system facilitates coordination between doctor and nurses, reduce workload and decreases possible errors in analogue processes. Furthermore, it represents a breakthrough in the digital transformation of hospital management, as it improves accuracy and optimises all processes, promoting safer and more efficient medical care.

Keywords

Healthcare centre, healthcare personnel, doctor, nurse, patient, treatment, clinical records, information management, healthcare coordination, mobile application.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

1.1 Motivación

En los centros sanitarios es fundamental una correcta gestión de la información clínica de los pacientes ingresados. Esto permite ofrecer una atención al paciente, que garantice la seguridad en el tratado de sus datos y una efectiva organización en los procedimientos internos del equipo sanitario para una mayor eficiencia en su tratamiento.

Los tratamientos, debido a los múltiples fármacos que se deben administrar y al ajuste de dosis individualizada, implican un incremento notorio en la dificultad y carga de trabajo sobre los profesionales sanitarios. Además, tradicionalmente, el registro del tratamiento y seguimiento de un paciente se realizan de manera manual, mediante el uso de libretas o plantillas que rellena el personal sanitario que posteriormente deben ser digitalizados. Este proceso está expuesto a grandes riesgos como puede ser la pérdida o deterioro de la información, la posible introducción de datos erróneos en la transcripción, la dificultad de una comunicación directa entre médicos y enfermeros, así como la imposibilidad de un acceso simultáneo a la información actualizada del paciente.

Actualmente, la implementación y el despliegue de tecnologías de la información en el ámbito sanitario ha ido en aumento con el desarrollo del HCE (historia clínica electrónica), la integración de la telemedicina y la IA. Estas nuevas tecnologías han supuesto grandes mejoras en la gestión de los pacientes respecto a la eficiencia y precisión de los procesos, la satisfacción del personal sanitario en su labor diaria y la experiencia percibida por los pacientes. Sin embargo, todavía en gran parte de los centros sanitarios la gestión del tratamiento está desactualizada y se sigue realizando de manera analógica.

Por todo ello, el desarrollo de una aplicación móvil orientada al manejo de la información de pacientes ingresados se convierte en una necesidad. Con esta herramienta el personal sanitario podrá consultar de forma instantánea las pautas de la

medicación del tratamiento, hacer un registro de las constantes vitales, anotar incidencias para hacer un seguimiento detallado y recibir cualquier tipo de notificación que les sea útil en el desarrollo de sus labores. De esta forma, el proyecto está orientado hacia la transformación y modernización digital de los servicios sanitarios, con el fin de mejorar la atención al paciente y la eficiencia operativa.

1.2 Objetivos

El objetivo principal del proyecto es el desarrollo de una aplicación móvil que dé apoyo a los profesionales sanitarios, facilitándoles tanto el seguimiento detallado de sus pacientes ingresados en tiempo real, como el registro de constantes vitales y anotaciones sobre su estado, y la prescripción y visionado del tratamiento del paciente.

Para lograr el objetivo principal, se exponen los siguientes objetivos específicos:

- Desarrollar una aplicación móvil intuitiva y accesible pensada para el personal sanitario sin necesidad de que tengan nociones ofimáticas para el uso de ella.
- Restringir el acceso de la aplicación móvil exclusivamente a los usuarios que estén registrados en la base de datos como personal sanitario.
- Crear funcionalidades prácticas para la gestión de la información del paciente ingresado.
- Facilitar la comunicación directa entre enfermero y médicos de una manera fluida y en tiempo real.
- Hacer más accesible la información del paciente en cualquier momento gracias a la portabilidad de una aplicación móvil.
- Digitalizar cualquier proceso y documentación relacionado con el ingreso del paciente que se hiciera previamente de manera analógica.

1.3 Plan de trabajo

Para cumplir el listado de objetivos enunciados en el apartado anterior, se fijaron unos plazos para realizar cada tarea. La planificación queda reflejada en la figura 1-1.

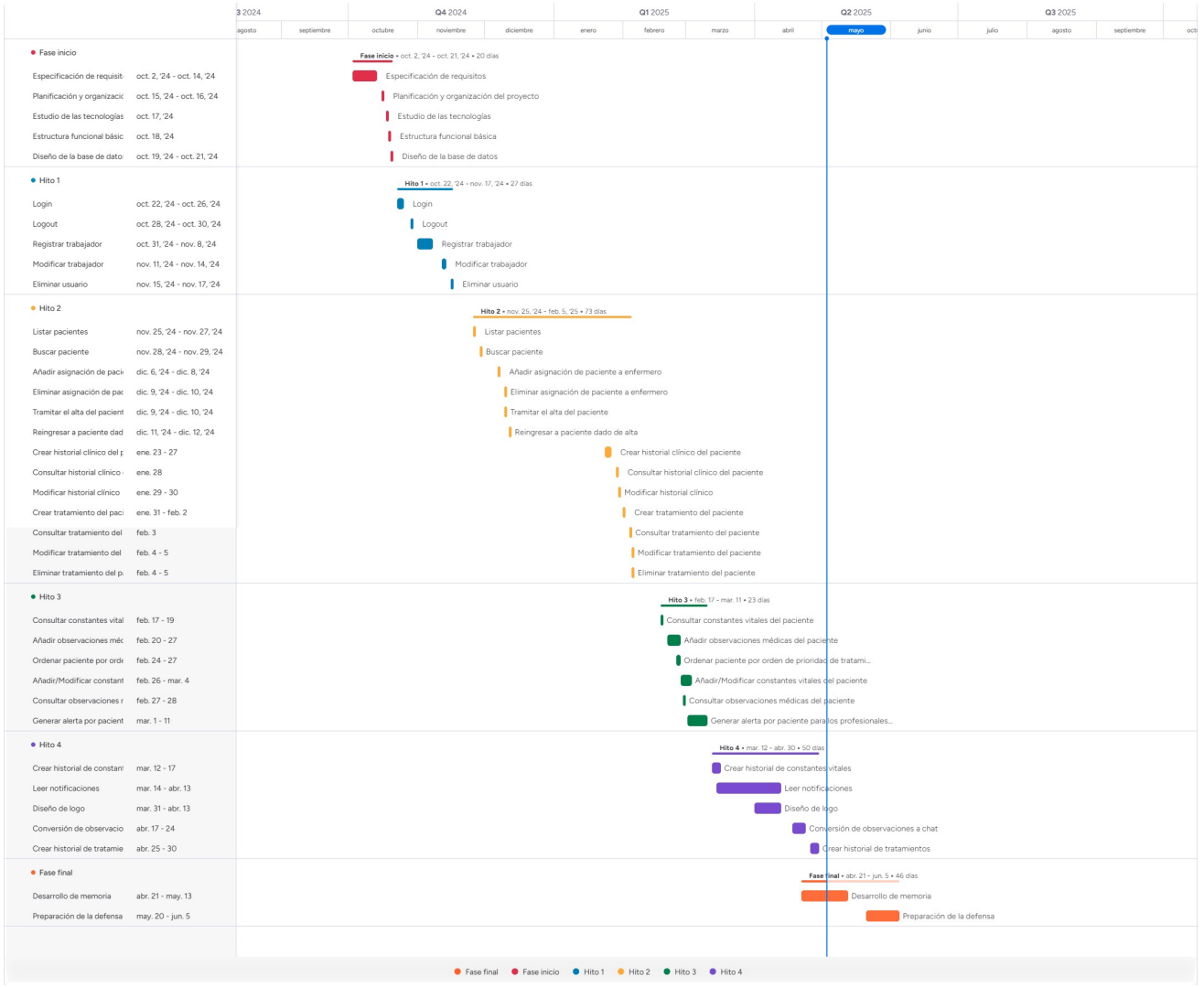


Figura 1-1. Diagrama de Gantt

Introduction

Motivation

In healthcare centres, the correct management of the clinical information of admitted patients is essential. This allows us to offer patient care that guarantees secure data processing and effective organization of the healthcare team's internal procedures to ensure the greatest possible efficiency in their treatment.

The treatments, due to the multiple drugs that must be administered to the patient and adjustment of each dose individually, implies an evident increase in the healthcare professionals' workloads and difficulty. Furthermore, traditionally, treatment's registration and patient's monitoring has been done manually, using notebooks or templates filled in by healthcare staff that has to be digitalize afterwards. This process is exposed to great risk, such as the loss or deterioration of the information, the possible introduction of erroneous data during the transcription process, the difficulty of direct communication between doctors and nurses, as well as the impossibility of simultaneous access to the patient's update information.

Nowadays, the implementation and deployment of information technology in the sanitary field has increased with the EHRs development (Electronic Health Record), the integration of telemedicine and the AI. These new technologies have entailed significant improvements in patient management in terms of the process's efficiency and accuracy, healthcare staff satisfaction in their daily work and the perceived patient experience. Nevertheless, in most healthcare centres, treatment handling is still outdated and carried out analogically.

Therefore, the development of a mobile application aimed on the admitted patient information management has become necessary. With this tool, healthcare personnel will be able to instantly consult treatment medication instructions, record vital signs, write down incidents for detailed monitoring and receive any type of notifications that may be useful in their work. As a result, the project is geared towards the digital

transformation and modernization of healthcare services, with the purpose of improving patient assistance and operational efficiency.

Goals

The main objective of the project is the development of a mobile application that supports healthcare professional by making easier the detailed, real-time monitoring of their inpatients, as well as the recording of vital signs and notes on their condition, and the creation and viewing of the patient's treatment.

To achieve the main objective, the following specific objectives are expounded:

- Develop an intuitive and accessible mobile application designed for healthcare staff without requiring office IT knowledge to use it.
- Restrict access to the mobile application exclusively to users registered as healthcare professionals in the database.
- Create practical functionalities for the management of admitted patient information.
- Make easier direct communication between nurses and doctors with fluency and in real time.
- Make patient information more accessible at any time thanks to the portability of a mobile application.
- Digitalise any process and documentation related to the patient's admission that was previously done analogically.

Work plan

In order to meet the list of objectives stated in the preceding section, deadlines were set for carrying each task. The schedule is shown in figure 1-1.

Capítulo 2 - Estado de la cuestión

2.1 Casiopea Mobility

Casiopea Mobility [1] es una aplicación móvil que el grupo Quirónsalud ha implantado en sus centros sanitarios. Con esta aplicación el personal sanitario puede acceder de forma sencilla, rápida y en tiempo real a la Historia Clínica Electrónica (HCE) desde un dispositivo móvil.

Esta aplicación facilita la visualización global del paciente, seguir su evolución, solicitar pruebas y visionar los resultados de estas, así como pautar la medicación del paciente y la frecuencia con la que se le debe administrar.

El grupo Quirónsalud ha ido implantando este nuevo sistema de manera progresiva en sus centros sanitarios por toda España para poder agilizar y mejorar la atención de los pacientes y dar más facilidades a los profesionales sanitarios.

2.2 Selene

Selene [2] es un programa informático desarrollado por el grupo GCM Clinical que trata de dar soporte a los profesionales sanitarios para que desde los ordenadores de los centros hospitalarios se puedan realizar las tareas correspondientes con la gestión de pacientes.

Actualmente está implantado en alrededor del 15% de hospitales públicos de España. Les facilita realizar el seguimiento del paciente, hacer una correcta planificación quirúrgica, gestión de Urgencias…

El mayor problema es la desactualización que sufren con este programa, lo poco intuitivo que es para los profesionales sanitarios y la poca flexibilidad respecto a la movilidad que proporciona debido a que está instalado en los ordenadores de sobremesa. Y aunque el grupo GCM sí que ha desarrollado una versión para dispositivos móviles llamada Selene Mobility [3], esta no se ha acabado de implantar por completo.

2.3 HCIS

HCIS [4] es un software que se está implantando en parte de los hospitales madrileños y que ha sido desarrollado por Dedalus. Ha sido diseñada para optimizar la gestión de información clínica y hospitalaria, permitiendo que todos los servicios sanitarios y hospitales de la región puedan compartir las historias clínicas de los pacientes.

La versión más extendida es para ordenadores de escritorio, y entre las principales funcionalidades que tiene más allá de esa conexión y compartición de las HCE de los pacientes están: la gestión de farmacia hospitalaria, planes de cuidados de enfermería, gestión de emergencias y hospitalizaciones, atención primaria y planificación de recursos entre otras muchas funcionalidades.

A diferencia de las soluciones digitales nombradas anteriormente, la aplicación desarrollada en este proyecto presenta una aplicación móvil innovadora en respuesta a las carencias de estas soluciones actuales.

Aunque Casiopea Mobility, Selene y HCIS han supuesto un gran avance en la gestión hospitalaria y digitalización de los datos clínicos de los pacientes, estas aplicaciones presentan ciertas limitaciones con respecto al manejo del paciente durante el ingreso, abriendo así la posibilidad a la creación de nuevas herramientas especializadas para optimizar estas funciones.

La aplicación móvil que hemos desarrollado tiene como objetivo principal optimizar la coordinación entre médicos y enfermeros durante el ingreso del paciente posibilitando la interacción en tiempo real, mejorando la lectura y el registro de los datos sobre el paciente y digitalizando procesos que se tenían que hacer de forma manual. Además, aprovechando la portabilidad de una aplicación móvil, hace que se adapte perfectamente al entorno de trabajo del personal sanitario ofreciéndoles la flexibilidad, movilidad y adaptabilidad que conlleva la atención en diferentes espacios del hospital sin depender de un sitio físico que cuente con un sistema informático.

La siguiente tabla comparativa analiza las principales funcionalidades claves de las soluciones actuales (Casiopea Mobility, Selene, HCIS) frente a la propuesto de la app VitaCare, destacando qué capacidades incorpora cada una y qué singularidades aporta nuestra aplicación.

Funcionalidad/Característica	**Casiopea Mobility**	**Selene**	**HCIS**	**App VitaCare**
Plataforma/Tipo de dispositivo	**Sí**: aplicación móvil diseñada para un acceso ágil a la HCE desde dispositivos móviles.	**No**: está orientada a entorno de escritorio, su versión móvil está en desarrollo y no resulta óptima para el uso diario.	**No**: solución pensad ara ordenadores, lo que limita su uso en ambientes de alta movilidad.	**Sí**: plataforma móvil creada específicamente para uso en planta.
Interfaz y usabilidad intuitiva	**Media**: optimizada para profesionales, pero muy genérica.	**Baja**: interfaz poco intuitiva con sobrecarga de PC.	**Media**: orientada a procesos clínicos y compleja.	**Alta**: diseño centrado en UX, sin conocimiento ofimáticos requeridos.
Portabilidad/movilidad	**Alta**: desarrollada para móvil.	**Baja**: desarrollada para ordenador escritorio.	**Baja**: desarrollada para ordenador escritorio.	**Alta**: desarrollada para móvil.

Actualización en tiempo real	**Sí**: permite la consulta y actualización instantánea de la HCE en móviles.	**Parcial**: algunas funciones se actualizan en tiempo real, aunque en la versión móvil se tiene que optimizar.	**Parcial**: su enfoque está en la integración de datos hospitalarios, pero no en actualizaciones instantáneas durante la atención.	**Sí**: implementa actualizaciones en tiempo real para el registro de constantes o cambios clínicos durante el ingreso entre otros.
Seguimiento digital del proceso de ingreso	**Parcial**: se centra en el acceso a la información clínica, sin una herramienta dedicada al seguimiento dinámico del ingreso.	**No**: no tiene un módulo específico para digitalizar y monitorizar el proceso de ingreso del paciente.	**No**: su uso está orientado a la administración general, sin cubrir el seguimiento del paciente.	**Sí**: está pensada para el seguimiento en planta, permitiendo registrar en tiempo real cualquier cambio del paciente durante el ingreso.

Sistema integrado de notificaciones/alertas	**No**: no tiene un sistema para alertar o notificar al personal de cambios o incidencias en la atención.	**No**: no dispone de funcionalidades para implementar notificaciones.	**No**: no está diseñado para gestionar alertar o notificaciones en tiempo real.	**Sí**: incorpora un sistema de notificaciones y alertar para comunicar en tiempo real cualquier cambio entre el personal sanitario.
Coordinación y comunicación interna	**Parcial**: se centra en la consulta de la HCE por lo que facilita el intercambio de datos, pero no tiene herramientas para coordinar la acción directa entre médicos y enfermeros.	**No**: no integra mecanismos que faciliten la comunicación directa entre profesionales.	**Parcial**: facilita el intercambio de datos, pero no dispone de canales dedicados para la comunicación en tiempo real entre sanitarios.	**Sí**: permita asignar enfermeros a paciente y cuenta con un chat interno para la comunicación directa entre médico y enfermeros, mejorando la coordinación.

Facilidad de uso para personal sanitario	**Sí**: orientada a sanitarios para el acceso a la HCE, aunque se centra en la consulta de datos únicamente.	**No**: requiere formación técnica para su manejo.	**No**: interfaz compleja.	**Sí**: diseñada para simplificar y facilitar su uso a sanitarios sin conocimientos avanzados, agilizando flujos de trabajo en situaciones de urgencia como con el envió de alertas inmediatas.
Interoperabilidad e integración con otros sistemas	**Sí**: se integra con la HCE existente, pero el alcance está limitado a la consulta de datos.	**Sí**: orientada a integrarse con sistemas hospitalarios.	**Sí**: orientada a integrarse con sistemas hospitalarios.	**Sí (futuro)**: aunque se centra en el seguimiento en planta, se tiene previsto evolucionar hacia la integración con otros sistemas hospitalarios y estándares.

Tabla 2-1. Comparación soluciones actuales

Capítulo 3 - Tecnologías empleadas

3.1 Android Studio

Android Studio [5] es el entorno de desarrollo integrado (IDE) oficial que se usa en el desarrollo de aplicaciones para Android. Ofrece un sistema de compilación flexible basado en Gradle, además de un entorno unificado donde se puede desarrollar para todo tipo de dispositivo Android, tablets, smartphones, relojes inteligentes. Entre sus principales funcionalidades destacan el editor de código inteligente con autocompletado y sugerencias, el emulador de dispositivos Android para probar la aplicación sin necesidad de utilizar un dispositivo físico, y herramientas avanzadas para la depuración y el análisis de rendimiento. Ofrece la integración con sistemas de control de versiones como GitHub, lo que facilita la colaboración grupal.

En nuestro caso, hemos utilizado la versión Koala Feature Drop 2024.1.2, la más actualizada en el momento en el que comenzamos el desarrollo de nuestra aplicación.

3.2 Github

Github [6] es una plataforma basada en la nube que permite almacenar, compartir y gestionar código de forma segura entre diferentes usuarios. Se basa en el sistema de control de versiones Git, lo que permite registrar todos los cambios que se realizan en el código a lo largo del tiempo. Nos ha permitido organizarnos mejor como equipo, dividir las tareas de forma estructurada y mantener siempre una copia de seguridad actualizada del estado de la aplicación.

3.3 Firebase

Firebase [7] es una plataforma que facilita el desarrollo tanto de aplicaciones web como móviles. Se encuentra alojada en la nube y está integrada con Google Cloud Platform, ofreciendo diversas herramientas para crear, sincronizar y gestionar aplicaciones.

3.3.1 Cloud Firestore

Cloud Firestore [8] es una base de datos flexible y escalable para el desarrollo en servidores, dispositivos móviles y aplicaciones web, perteneciente a Firebase y Google Cloud. Los datos se almacenan en documentos que contienen campos de diferentes tipos de valores, organizados dentro de colecciones. Esta estructura permite crear modelos de datos complejos de manera sencilla y adaptarlos a las necesidades específicas de la aplicación.

3.3.2 Cloud Messaging

Firebase Cloud Messaging (FCM) [9] es una solución de mensajería multiplataforma que permite enviar mensajes de forma segura y eficiente. Esto ayuda a captar la atención de los usuarios, mejorar su experiencia y fomentar su retención dentro de la aplicación.

Para la implementación de FCM se incluyen dos componentes principales: por un lado, un entorno de confianza, en nuestro caso Cloud Functions for Firebase, encargado de compilar, segmentar y enviar los mensajes; y por otro, la propia aplicación, encargada de recibir los mensajes a través del servicio de transporte correspondiente de cada plataforma.

3.3.3 Cloud Functions

Cloud Functions [10] es un framework sin servidores que te permite ejecutar automáticamente el código de backend en respuesta a diferentes tipos de eventos. El código, en nuestro proyecto escrito en JavaScript, se almacena y ejecuta en la infraestructura de Google Cloud.

Una vez escrita e implementada la función, los servidores de Google comienzan a administrarla de manera automática, sin necesidad de tener que gestionar o escalar servidores.

3.4 Java

Java [11] es un lenguaje de programación multiplataforma orientado a objetos. Sustenta aplicaciones, sistemas operativos de smartphones, software empresarial y muchos programas conocidos. A pesar de haberse inventado hace más de 20 años, Java es actualmente el lenguaje de programación más popular para el desarrollo de aplicaciones.

Su integración nativa, versión 8, en Android Studio, junto con su amplia documentación y la gran cantidad de librerías disponibles, nos ha permitido implementar de manera eficiente tanto la lógica de negocio como la gestión de la interfaz de usuario.

3.5 JavaScript

JavaScript [12] es un lenguaje de programación muy flexible, utilizado para crear funcionalidades, automatizar tareas, modificar la estructura de una página web, etc. Es conocido por su flexibilidad, su capacidad para ejecutarse tanto en el navegador como en el servidor, y su facilidad para integrar funcionalidades interactivas en las aplicaciones.

Como mencionamos anteriormente, en nuestro proyecto utilizamos JavaScript para el desarrollo de las funciones de backend en Cloud Functions, lo que nos permite automatizar el envío de mensajes.

3.6 XML

El lenguaje de marcado extensible (XML) [13] permite definir y almacenar datos de forma estructurada, facilitando el intercambio de información entre sistemas de computación. A diferencia de otros lenguajes de programación, XML no puede realizar operaciones de computación por sí mismo. Para definir los layouts de nuestra aplicación, utilizamos XML, estructurando los elementos de la interfaz de usuario, como botones, textos, imágenes, entre otros.

3.7 GIMP

GIMP [14] es una aplicación gráfica que permite crear, editar y retocar imágenes pixelares fijas o animadas. Además, ofrece múltiples funcionalidades adicionales, como utilizar una gran variedad de herramientas (pinceles, filtros y efectos), o realizar montajes, cartografías e infografías. Utilizamos GIMP para diseñar los logotipos, tanto el que aparece en la pantalla de carga, como el icono principal de la aplicación.

Capítulo 4 - Especificación de requisitos

Este capítulo está dedicado a la descripción de actores y la especificación de requisitos del proyecto.

4.1 Actores

A continuación, se identifican los principales actores del sistema y se realiza una descripción adecuada del papel que tienen cada uno de ellos:

- Administrador: usuario destinado a la gestión de cuentas de usuario. Podrá dar de alta a nuevos trabajadores del hospital (médicos, enfermeros), así como modificar los datos de estos y dar de baja a cualquier usuario (profesional sanitario y pacientes). Además, podrá administrar los datos almacenados en la aplicación.
- Médico: usuario destinado al profesional sanitario que sea médico. Podrá realizar todas las funcionalidades relacionadas con la gestión de sus pacientes. Tales como hacer un seguimiento de estos, gestionar el tratamiento, el historial clínico…
- Enfermero: usuario destinado al profesional sanitario que sea enfermero. Podrá realizar funcionalidades relacionadas con la ayuda en el tratamiento al médico, seguimiento del paciente y su estado.

4.2 Casos de uso

Los diagramas de casos de uso correspondientes a los actores administrador, médico y enfermero se muestran en las figuras 4-1, 4-2 y 4-3, respectivamente.

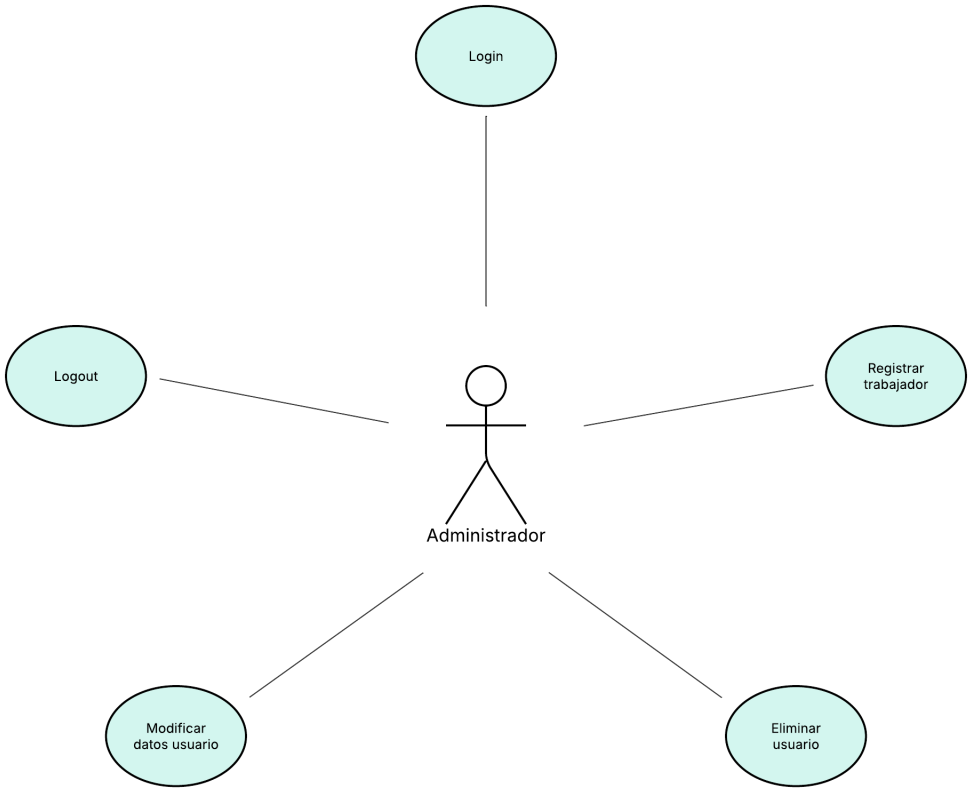


Figura 4-1. Diagrama de casos de uso actor administrador

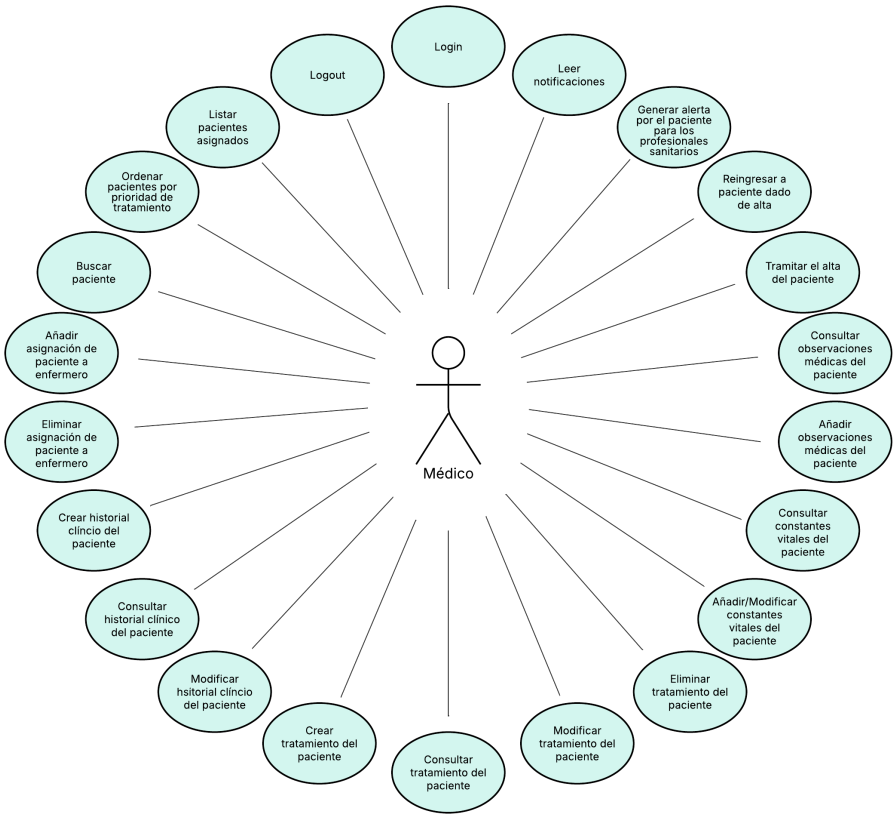


Figura 4-2. Diagrama de casos de uso actor médico

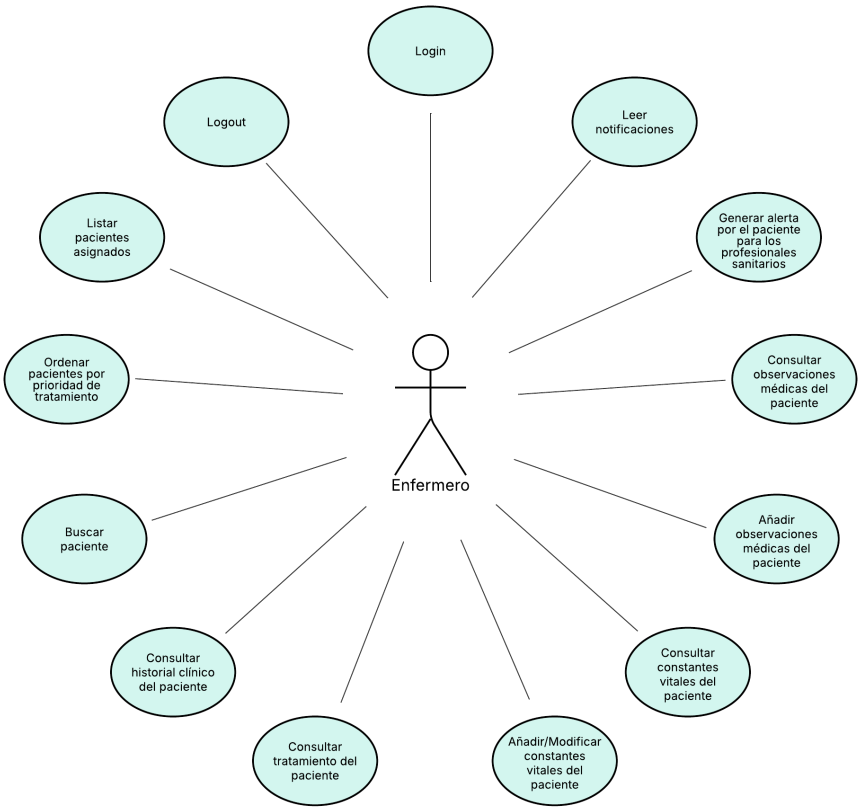


Figura 4-3. Diagrama de casos de uso actor enfermero

Se procede a detallar los casos de uso para cada funcionalidad de la aplicación.

4.2.1 Gestión de cuenta

Requisito	**Login**	
Identificador	1.1	
Prioridad	Alta	
Precondición	El usuario debe estar previamente registrado en el sistema.	
Descripción	El usuario inicia sesión para acceder a la aplicación.	
Entrada	Datos del usuario.	
Salida	Página principal.	
Secuencia normal	Paso	Acción
	1	Al entrar en la aplicación, se le pide al usuario los datos necesarios para el inicio de sesión.
	2	El usuario rellena todos los campos del formulario y pulsa el botón "Login".
	3	El sistema valida la información y se muestra la pantalla principal.
Postcondición	El usuario inicia sesión y logra acceder a la aplicación.	
Excepciones	Paso	Acción
	3	El sistema no encuentra un usuario con los datos introducidos y muestra un mensaje de error.
Comentarios	NA	
Actores	Administrador, Médico, Enfermero.	

Tabla 4-1. Login

Requisito	**Logout**	
Identificador	1.2	
Prioridad	Alta	
Precondición	El usuario debe haber iniciado sesión previamente.	
Descripción	El usuario cierra la sesión en la aplicación.	
Entrada	NA	
Salida	Página de inicio de sesión.	
Secuencia normal	Paso	Acción
	1	El usuario accede al menú para cerrar sesión.
	2	El usuario pulsa el botón de "Cerrar sesión".
	3	El sistema cierra la sesión del usuario y le muestra el formulario de inicio de sesión.
Postcondición	El usuario ha cerrado sesión.	
Excepciones	Paso	Acción
	3	El sistema no ha podido cerrar sesión y muestra un mensaje de error
Comentarios	NA	
Actores	Administrador, Médico, Enfermero	

Tabla 4-2. Logout

4.2.2 Gestión de administrador

Requisito	**Registrar trabajador**	
Identificador	2.1	
Prioridad	Alta	
Precondición	El administrador debe haber iniciado sesión.	
Descripción	Se registra un nuevo trabajador sanitario que no existe en el sistema.	
Entrada	Datos del trabajador.	
Salida	Mensaje: "Trabajador añadido.".	
Secuencia normal	Paso	Acción
	1	El administrador entra en el listado de trabajadores pulsa en "Añadir nuevo trabajador" y aparece en pantalla el formulario de registro.
	2	Rellena los datos del nuevo usuario y pulsa "Añadir trabajador".
	3	El sistema pide confirmación y pulsa "Confirmar".
	4	El sistema valida la información y muestra la pantalla con el nuevo listado de trabajadores.
Postcondición	Se ha creado correctamente la cuenta del nuevo usuario.	
Excepciones	Paso	Acción
	3	El administrador pulsa el botón "Cancelar".
	4	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Administrador	

Tabla 4-3. Registrar trabajador

Requisito	**Modificar datos de trabajador**	
Identificador	2.2	
Prioridad	Alta	
Precondición	El administrador debe tener una sesión iniciada y el usuario al que quiere modificar los datos debe estar registrado en el sistema.	
Descripción	El administrador puede modificar los datos de otras cuentas de usuario.	
Entrada	Datos del trabajador.	
Salida	Mensaje: "Trabajador modificado.".	
Secuencia normal	Paso	Acción
	1	El administrador busca el usuario en el listado de trabajadores y desliza hacia la derecha sobre el trabajador para modificar sus datos.
	2	Modifica los datos necesarios y se pulsa "Modificar trabajador".
	3	El sistema pide confirmación y pulsa "Confirmar".
	4	El sistema valida la información y muestra el listado de trabajadores con la información actualizada.
Postcondición	Se han modificado correctamente los datos del usuario.	
Excepciones	Paso	Acción
	3	El administrador pulsa el botón "Cancelar".
	4	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Administrador	

Tabla 4-4. Modificar datos del trabajador

Requisito	**Eliminar usuario**	
Identificador	2.3	
Prioridad	Alta	
Precondición	El administrador debe haber iniciado sesión y el usuario al que quiere dar de baja debe estar registrado en el sistema.	
Descripción	Se elimina un usuario que existe en el sistema.	
Entrada	Datos del usuario.	
Salida	Mensaje: "Usuario eliminado.".	
Secuencia normal	Paso	Acción
	1	El administrador busca el usuario en el listado de trabajadores o de pacientes y desliza hacia la izquierda sobre el usuario para eliminarlo.
	2	El sistema pide confirmación y pulsa "Confirmar".
	3	Se elimina el usuario especificado y muestra la pantalla actualizada con el nuevo listado de usuarios.
Postcondición	Se ha eliminado correctamente al usuario.	
Excepciones	Paso	Acción
	2	El administrador pulsa el botón "Cancelar".
	3	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Administrador	

Tabla 4-5. Eliminar usuario

4.2.3 Gestión de pacientes común a médico y enfermero

Requisito	**Listar pacientes asignados**	
Identificador	3.1	
Prioridad	Media	
Precondición	El profesional sanitario debe haber iniciado sesión y tener algún paciente asignado.	
Descripción	El profesional sanitario puede ver el listado de pacientes que tiene asignados.	
Entrada	NA	
Salida	Lista de pacientes.	
Secuencia normal	Paso	Acción
	1	Al entrar a la aplicación el sistema le mostrará el listado de los pacientes que le han sido asignados.
Postcondición	Se visiona correctamente el listado de pacientes.	
Excepciones	Paso	Acción
	1	Si al profesional sanitario no se le han asignado pacientes todavía, no se mostrará nada por pantalla.
Comentarios	NA	
Actores	Médico-Enfermero (profesional sanitario)	

Tabla 4-6. Listar pacientes asignados

Requisito	**Ordenar pacientes por prioridad de tratamiento.**	
Identificador	3.2	
Prioridad	Baja	
Precondición	El profesional sanitario debe iniciar sesión y tener un listado de pacientes asignados con tratamiento.	
Descripción	Al profesional sanitario se le muestran ordenados los pacientes según la prioridad de su tratamiento.	
Entrada	NA	
Salida	Lista ordenada de pacientes.	
Secuencia normal	Paso	Acción
	1	Al entrar a la aplicación el sistema el sistema le mostrará al profesional sanitario el listado de pacientes ordenados por prioridad de tratamiento.
Postcondición	Se ordenan correctamente todos los pacientes por prioridad de tratamiento.	
Excepciones	Paso	Acción
	1	El profesional sanitario no tiene asignado ningún paciente con tratamiento.
Comentarios	NA	
Actores	Médico-Enfermero (profesionales sanitarios)	

Tabla 4-7. Ordenar pacientes por prioridad de tratamiento

Requisito	**Buscar paciente**	
Identificador	3.3	
Prioridad	Media	
Precondición	El médico debe haber iniciado sesión y el paciente debe estar registrado en el sistema.	
Descripción	Médico busca el paciente.	
Entrada	Datos del usuario.	
Salida	Paciente buscado.	
Secuencia normal	Paso	Acción
	1	En la pantalla de inicio, el médico introduce los datos del usuario en la barra de búsqueda "Buscar paciente...".
	2	Al pulsar sobre la lupa, se muestra por pantalla al paciente que coincide con los datos introducidos.
Postcondición	Se actualiza la lista de pacientes y aparece solo el paciente buscado.	
Excepciones	Paso	Acción
	2	No hay ningún paciente con los datos introducidos.
Comentarios	NA	
Actores	Médico	

Tabla 4-8. Buscar paciente

Requisito	**Consultar historial clínico del paciente**	
Identificador	3.4	
Prioridad	Media	
Precondición	El médico debe haber iniciado sesión y el paciente debe estar registrado en el sistema.	
Descripción	Médico consulta el historial clínico del paciente.	
Entrada	Datos del usuario.	
Salida	Historial clínico del paciente.	
Secuencia normal	Paso	Acción
	1	El médico busca al paciente en su lista "Mis pacientes" y se desplaza hasta la sección historial clínico.
	2	Aparece en pantalla el historial clínico de este.
Postcondición	Se muestra el historial clínico del paciente.	
Excepciones	Paso	Acción
	1	No hay ningún paciente con los datos introducidos.
Comentarios	NA	
Actores	Médico	

Tabla 4-9. Consultar historial clínico del paciente

Requisito	**Consultar tratamiento del paciente**	
Identificador	3.5	
Prioridad	Media	
Precondición	El médico debe haber iniciado sesión, el paciente debe estar registrado en el sistema y tener algún tratamiento.	
Descripción	El médico consulta el tratamiento del paciente.	
Entrada	Datos del usuario.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El médico busca al paciente en su lista "Mis pacientes", y se desplaza hasta la sección tratamiento.
	2	Aparece en pantalla el historial clínico de este, y también se puede ver el historial con tratamientos pasados.
Postcondición	Se muestra el tratamiento del paciente.	
Excepciones	Paso	Acción
	2	El paciente no tiene ningún tratamiento y muestra un mensaje.
Comentarios	NA	
Actores	Médico	

Tabla 4-10. Consultar tratamiento del paciente

Requisito	**Añadir/Modificar constantes vitales del paciente**	
Identificador	3.6	
Prioridad	Alta	
Precondición	El profesional sanitario debe tener una sesión iniciada, y el paciente debe estar registrado en el sistema.	
Descripción	El profesional sanitario añade las constantes vitales del paciente.	
Entrada	Datos de las constantes vitales.	
Salida	Mensaje: "Constantes vitales actualizadas.".	
Secuencia normal	Paso	Acción
	1	El profesional sanitario busca al paciente en su lista "Mis pacientes", se desplaza hasta la sección historial clínico y pulsa sobre las constantes vitales.
	2	Se introducen las nuevas métricas de las constantes vitales a modificar y pulsa en "Guardar cambios".
	3	El sistema pide confirmación y pulsa "Confirmar".
	4	El sistema valida los datos cambiados y se muestra por pantalla las nuevas métricas de las constantes vitales.
Postcondición	Se guardan en el sistema las constantes vitales les paciente.	
Excepciones	Paso	Acción
	3	Pulsa el botón "Cancelar".
	4	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico-Enfermero (profesional sanitario)	

Tabla 4-11. Añadir/Modificar constantes vitales del paciente

Requisito	**Consultar constantes vitales del paciente**	
Identificador	3.7	
Prioridad	Alta	
Precondición	El profesional sanitario debe tener una sesión iniciada, el paciente debe estar registrado en el sistema, que se le haya tomado las constantes vitales y estén guardadas.	
Descripción	El profesional sanitario consulta las constantes vitales del paciente.	
Entrada	Datos del usuario.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El profesional sanitario busca al paciente en su lista "Mis pacientes", se desplaza hasta la sección historial clínico y pulsa sobre las constantes vitales.
	2	Aparece en pantalla las constantes vitales de este, y también se puede ver el historial de las constantes vitales que se le han tomado anteriormente.
Postcondición	Se muestra las constantes vitales del paciente.	
Excepciones	Paso	Acción
	2	El paciente no tiene ninguna constante vital tomada.
Comentarios	NA	
Actores	Médico-Enfermero (profesional sanitario)	

Tabla 4-12. Consultar constantes vitales del paciente

Requisito	**Añadir observaciones médicas del paciente**	
Identificador	3.8	
Prioridad	Alta	
Precondición	El profesional sanitario debe tener una sesión iniciada, y el paciente debe estar registrado en el sistema.	
Descripción	El profesional sanitario añade observaciones médicas del paciente.	
Entrada	Datos de las observaciones médicas.	
Salida	Mensaje: "Observaciones enviadas.".	
Secuencia normal	Paso	Acción
	1	El profesional sanitario busca al paciente en su lista "Mis pacientes", se desplaza hasta la sección datos personales y pulsa sobre el chat de observaciones médicas.
	2	Introduce las observaciones médicas en la caja de mensaje y le da al botón de enviar.
	3	El sistema valida la información enviada y se muestra por pantalla el mensaje con las nuevas observaciones médicas en el chat.
Postcondición	Se añade correctamente en el sistema las observaciones médicas del paciente.	
Excepciones	Paso	Acción
	3	El paciente no tiene ninguna observación médica guardada.
Comentarios	NA	
Actores	Médico-Enfermero (profesional sanitario)	

Tabla 4-13. Añadir observaciones médicas del paciente

Requisito	**Consultar observaciones médicas del paciente**	
Identificador	3.9	
Prioridad	Alta	
Precondición	El profesional sanitario debe tener una sesión iniciada, y el paciente debe estar registrado en el sistema.	
Descripción	El profesional sanitario consulta las observaciones médicas del paciente para hacer un seguimiento.	
Entrada	Datos del usuario.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El profesional sanitario busca al paciente en su lista "Mis pacientes", se desplaza hasta la sección datos personales y pulsa sobre el chat de observaciones médicas.
	2	Se muestra por pantalla el chat con las observaciones médicas guardades del paciente tomadas durante su ingreso.
Postcondición	Se muestras las observaciones médicas del paciente.	
Excepciones	Paso	Acción
	3	En caso de errores, se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico-Enfermero (profesional sanitario)	

Tabla 4-14. Consultar observaciones médicas del paciente

Requisito	**Generar alerta por el paciente para los profesionales sanitarios**	
Identificador	3.10	
Prioridad	Media	
Precondición	El profesional sanitario debe tener una sesión iniciada y el paciente debe tener un enfermero asignado.	
Descripción	El profesional sanitario genera una notificación de alerta del paciente para el enfermero asignado.	
Entrada	Datos del usuario.	
Salida	Mensaje: "Nueva alerta generada.".	
Secuencia normal	Paso	Acción
	1	El profesional sanitario busca al paciente en su lista "Mis pacientes", y pulsa sobre el símbolo de la campana roja.
	2	El sistema pide confirmación y pulsa "Continuar".
	3	El sistema envía la alerta al profesional sanitario asignado.
Postcondición	Se envía correctamente la alerta al profesional sanitario.	
Excepciones	Paso	Acción
	3	Pulsa el botón "Cancelar".
	3	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico-Enfermero (profesional sanitario)	

Tabla 4-15. Generar alerta por el paciente para los profesionales sanitarios

Requisito	**Leer notificaciones**	
Identificador	3.11	
Prioridad	Baja	
Precondición	El profesional sanitario debe haber iniciado sesión y ha recibido notificaciones.	
Descripción	El profesional sanitario puede leer las notificaciones recibidas.	
Entrada	Datos del usuario.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El profesional sanitario entra en el menú y pulsa sobre "Notificaciones" para poder ver las notificaciones recibidas recientemente.
	2	El sistema le mostrará las notificaciones recibidas.
	3	Si pulsa sobre alguna notificación, el sistema le redireccionará a ese apartado para que pueda ver la información relacionada con la notificación.
Postcondición	Se muestran correctamente las notificaciones.	
Excepciones	Paso	Acción
	2	Si el profesional sanitario no ha recibido ninguna notificación anteriormente, no se mostrará nada por pantalla.
Comentarios	NA	
Actores	Médico-Enfermero (profesional sanitario)	

Tabla 4-16. Leer notificaciones

4.2.4 Gestión de pacientes específico de médico

Requisito	**Crear historial clínico del paciente**	
Identificador	4.1	
Prioridad	Alta	
Precondición	El médico debe haber iniciado sesión y el paciente no debe estar registrado en el sistema.	
Descripción	El médico añade un nuevo paciente y le abre un historial clínico.	
Entrada	Datos del historial clínico.	
Salida	Mensaje: "Nuevo historial clínico creado.".	
Secuencia normal	Paso	Acción
	1	El médico añade un nuevo paciente pulsando en "Nuevo historial clínico".
	2	Se introducen todos los datos necesarios y pulsa en "Crear nuevo historial clínico".
	3	El sistema pide confirmación y pulsa "Confirmar".
	4	El sistema valida la información proporcionada y se muestra por pantalla la nueva "Lista pacientes".
Postcondición	Se ha creado correctamente el historial clínico del paciente.	
Excepciones	Paso	Acción
	3	El médico pulsa el botón "Cancelar".
	4	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico	

Tabla 4-17. Crear historial clínico del paciente

Requisito	**Modificar historial clínico del paciente**	
Identificador	4.2	
Prioridad	Alta	
Precondición	El médico debe haber iniciado sesión y el paciente debe estar registrado en el sistema.	
Descripción	El médico modifica/rellena algunos campos del historial clínico del paciente.	
Entrada	Datos del historial clínico.	
Salida	Mensaje: "Historial clínico modificado.".	
Secuencia normal	Paso	Acción
	1	El médico busca al paciente en su lista "Mis pacientes", se desplaza hasta la sección historial clínico y pulsa el botón "Modificar historial clínico".
	2	Se introducen los datos a modificar del historial clínico y pulsa en "Modificar historial clínico".
	3	El sistema pide confirmación y pulsa "Confirmar".
	4	El sistema valida los datos cambiados y se muestra por pantalla el nuevo historial clínico actualizado.
Postcondición	Se ha modificado correctamente el historial clínico del paciente.	
Excepciones	Paso	Acción
	3	Pulsa el botón "Cancelar".
	4	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico	

Tabla 4-18. Modificar historial clínico del paciente

Requisito	**Crear tratamiento del paciente**	
Identificador	4.3	
Prioridad	Alta	
Precondición	El médico debe haber iniciado sesión y el paciente debe estar registrado en el sistema.	
Descripción	El médico crea un nuevo tratamiento para el paciente.	
Entrada	Datos del tratamiento.	
Salida	Mensaje: "Nuevo tratamiento creado.".	
Secuencia normal	Paso	Acción
	1	El médico busca al paciente en su lista "Mis pacientes", se desplaza hasta la sección tratamiento y pulsa el botón "Crear nuevo tratamiento".
	2	Se introducen todos los datos necesarios y pulsa en "Crear nuevo tratamiento".
	3	El sistema pide confirmación y pulsa "Confirmar".
	4	El sistema valida la información proporcionada y se muestra por pantalla el nuevo tratamiento creado.
Postcondición	Se ha creado correctamente el nuevo tratamiento.	
Excepciones	Paso	Acción
	3	Pulsa el botón "Cancelar".
	4	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico	

Tabla 4-19. Crear tratamiento del paciente

Requisito	**Modificar tratamiento del paciente**	
Identificador	4.4	
Prioridad	Alta	
Precondición	El médico debe haber iniciado sesión, el paciente debe estar registrado en el sistema y tener algún tratamiento.	
Descripción	El médico modifica el tratamiento que tiene el paciente.	
Entrada	Datos del tratamiento.	
Salida	Mensaje: "Tratamiento modificado.".	
Secuencia normal	Paso	Acción
	1	El médico busca al paciente en su lista "Mis pacientes", se desplaza hasta la sección tratamiento y pulsa el botón "Modificar tratamiento".
	2	Se introducen los datos a modificar del tratamiento y pulsa en "Modificar tratamiento".
	3	El sistema pide confirmación y pulsa "Confirmar".
	4	El sistema valida los datos cambiados y se muestra por pantalla el nuevo historial clínico actualizado.
Postcondición	Se ha modificado correctamente el tratamiento del paciente.	
Excepciones	Paso	Acción
	3	Pulsa el botón "Cancelar".
	4	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico	

Tabla 4-20. Modificar tratamiento del paciente

Requisito	**Eliminar tratamiento del paciente**	
Identificador	4.5	
Prioridad	Alta	
Precondición	El médico debe haber iniciado sesión, el paciente debe estar registrado en el sistema y tener algún tratamiento.	
Descripción	El médico elimina el tratamiento del paciente.	
Entrada	Datos del usuario.	
Salida	Mensaje: "Tratamiento eliminado.".	
Secuencia normal	Paso	Acción
	1	El médico busca al paciente en su lista "Mis pacientes", se desplaza hasta la sección tratamiento y pulsa sobre la papelera.
	2	El sistema pide confirmación y pulsa "Confirmar".
	3	El sistema elimina el tratamiento y actualiza la pantalla quitando el tratamiento.
Postcondición	Se elimina correctamente el tratamiento.	
Excepciones	Paso	Acción
	3	Pulsa el botón "Cancelar".
	3	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico	

Tabla 4-21. Eliminar tratamiento del paciente

Requisito	**Añadir asignación de paciente a enfermero**	
Identificador	4.6	
Prioridad	Alta	
Precondición	El médico debe tener una sesión iniciada, y el paciente y enfermero deben estar registrados en el sistema.	
Descripción	El médico asigna un paciente a un enfermero.	
Entrada	Datos del enfermero.	
Salida	Mensaje: "Enfermero asignado correctamente.".	
Secuencia normal	Paso	Acción
	1	El médico busca al paciente en su lista "Mis pacientes", se desplaza hasta la sección datos personal y pulsa sobre el botón "Asignar".
	2	Selecciona uno de los enfermeros disponibles y pulsa "Confirmar".
	3	El sistema realiza la asignación y se muestra por pantalla el nombre del enfermero asignado al paciente.
Postcondición	Se asigna correctamente el paciente al enfermero.	
Excepciones	Paso	Acción
	3	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico	

Tabla 4-22. Añadir asignación de paciente a enfermero

Requisito	**Eliminar asignación de paciente a enfermero**	
Identificador	4.7	
Prioridad	Alta	
Precondición	El médico debe tener una sesión iniciada, y el paciente y enfermero deben estar registrados en el sistema.	
Descripción	El médico asigna un paciente a un enfermero.	
Entrada	Datos del usuario.	
Salida	Mensaje: "Enfermero asignado correctamente.".	
Secuencia normal	Paso	Acción
	1	El médico busca al paciente en su lista "Mis pacientes", se desplaza hasta la sección datos personal y pulsa sobre el botón "Asignar".
	2	Selecciona uno de los enfermeros disponibles y pulsa "Confirmar".
	3	El sistema realiza la asignación y se muestra por pantalla el nombre del enfermero asignado al paciente.
Postcondición	Se elimina correctamente la asignación.	
Excepciones	Paso	Acción
	3	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico	

Tabla 4-23. Eliminar asignación de paciente a enfermero

Requisito	**Tramitar el alta del paciente**	
Identificador	4.8	
Prioridad	Alta	
Precondición	El médico debe iniciar sesión y el paciente debe estar registrado en el sistema.	
Descripción	El médico le da el alta al paciente.	
Entrada	Datos del usuario.	
Salida	Mensaje: "Paciente dado de alta.".	
Secuencia normal	Paso	Acción
	1	El médico busca al paciente en su lista "Mis pacientes", se desplaza hasta la sección datos personales y pulsa sobre "Tramitar el alta".
	2	El sistema pide confirmación y pulsa "Continuar".
	3	Se muestra por pantalla la lista "Mis pacientes" del médico con el paciente dado de alta deshabilitado.
Postcondición	Se tramita correctamente el alta del paciente.	
Excepciones	Paso	Acción
	2	Pulsa el botón "Cancelar".
	2	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico	

Tabla 4-24. Tramitar el alta del paciente

Requisito	**Reingresar a paciente dado de alta**	
Identificador	4.9	
Prioridad	Alta	
Precondición	El médico debe iniciar sesión y el paciente además de estar registrado en el sistema debe haber sido dado de alta anteriormente.	
Descripción	El médico readmite a un paciente.	
Entrada	Datos del usuario.	
Salida	Mensaje: "Paciente readmitido.".	
Secuencia normal	Paso	Acción
	1	El médico busca al paciente en su lista "Mis pacientes" y pulsa sobre él.
	2	El sistema pide confirmación y pulsa "Continuar".
	3	Se muestra por pantalla la información del paciente y sus secciones.
Postcondición	Se tramita correctamente el reingreso del paciente.	
Excepciones	Paso	Acción
	2	Pulsa el botón "Cancelar".
	2	En caso de error se muestra por pantalla un mensaje especificando el problema.
Comentarios	NA	
Actores	Médico	

Tabla 4-25. Reingresar a paciente dado de alta

Capítulo 5 - Arquitectura

La arquitectura seguida para implementar la aplicación móvil del proyecto ha sido el modelo cliente-servidor. En dicho modelo se identifica el cliente y servidor como entidades distintas que establecen una comunicación entre ellas para interactuar y realizar las tareas que conlleven peticiones y respuestas.

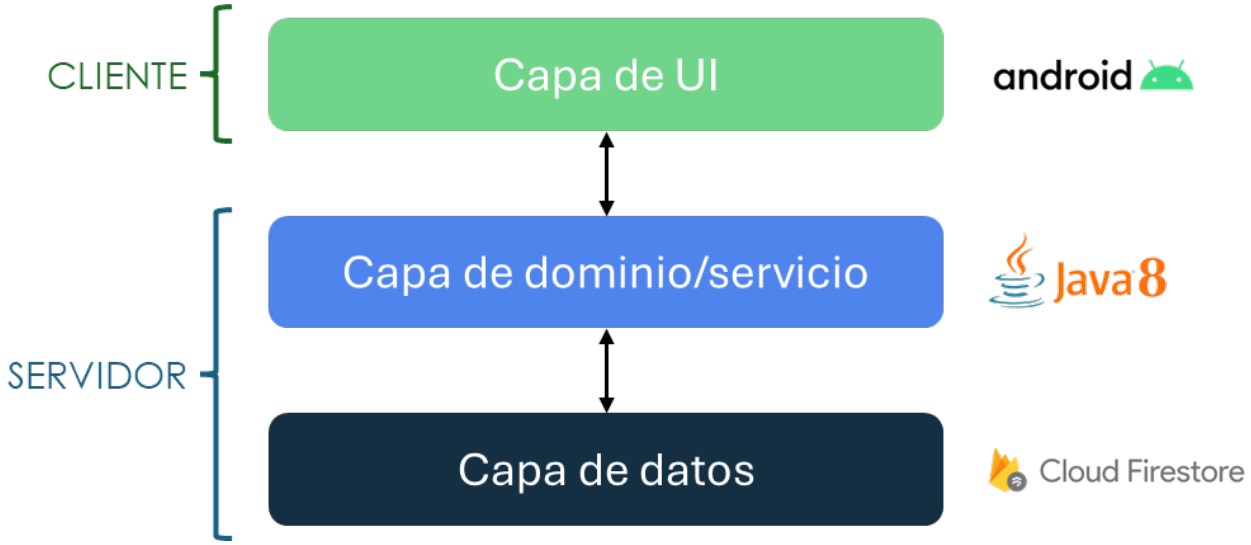


Figura 5-1. Diagrama de la arquitectura de la aplicación

Tal y como se muestra en la figura 5-1 tenemos una arquitectura de tres capas. Dos de las cuales forman parte del servidor y la tercera que está destinada al cliente.

El cliente interactúa directamente con el usuario. El cliente está formado por la capa de UI, que está desarrollada en Android usando Java. La capa de UI recoge las entradas/peticiones del usuario y muestra la información en la pantalla. El cliente tiene una capa intermedia que le deniega el acceso directo a la base de datos, esta capa forma parte del servidor y es la capa de servicio que se encarga de ejecutar la lógica de negocio.

Por el otro lado, tenemos el servidor que está formado por dos capas internas, la capa de servicio y la capa de datos. La capa de servicio es la encargada de centralizar la lógica de negocio, las validaciones y casos de uso de la aplicación y está

implementado en Java 8. Esta capa es la intermediaria entre la presentación y la persistencia de datos, se ocupa de dar respuesta a las peticiones que viene de la capa UI y hacer las peticiones a la capa de datos.

Y dentro del servidor la capa encargada de la gestión del almacenamiento de los datos y la consulta de estos es la capa de datos. Para el almacenaje de la información hemos utilizado Cloud Firestore de Firebase. Esta capa tiene como único objetivo realizar las operaciones de lectura, escritura, actualización y eliminación de la base de datos, sin que se mezcle con la lógica de negocio.

Capítulo 6 - Modelo de datos

En este capítulo se explica el modelo de datos que ha sido utilizado en el proyecto.

Se ha empleado Firestore de Firebase [8], un sistema de gestión de bases de datos no relacionales (NoSQL) orientado a documentos. Firestore almacena los datos en documentos que se agrupan en colecciones. Cada documento es un objeto similar a un tipo JSON, sin estar obligado a seguir un esquema predefinido con anterioridad y donde los campos pueden tener datos de diferentes tipos.

La base de datos está alojada en la nube, creada en la infraestructura de Google Cloud. Entre las características que más destacan de este sistema y por las que decidimos elegir este soporte es debido a que ofrece una alta flexibilidad que nos permitía construir estructuras de datos más flexibles además de contener objetos anidados y subcolecciones. También brinda el contar con actualizaciones en tiempo real y asistencia sin conexión almacenando en caché los datos que usa la aplicación y cuando el dispositivo vuelva a tener conexión Cloud Firestore sincroniza los cambios locales.

Seguidamente describiremos las colecciones que han sido creadas y utilizadas para el proyecto con su correspondiente diagrama de entidad relación de la base de datos, quedando reflejado en la figura 6-1.

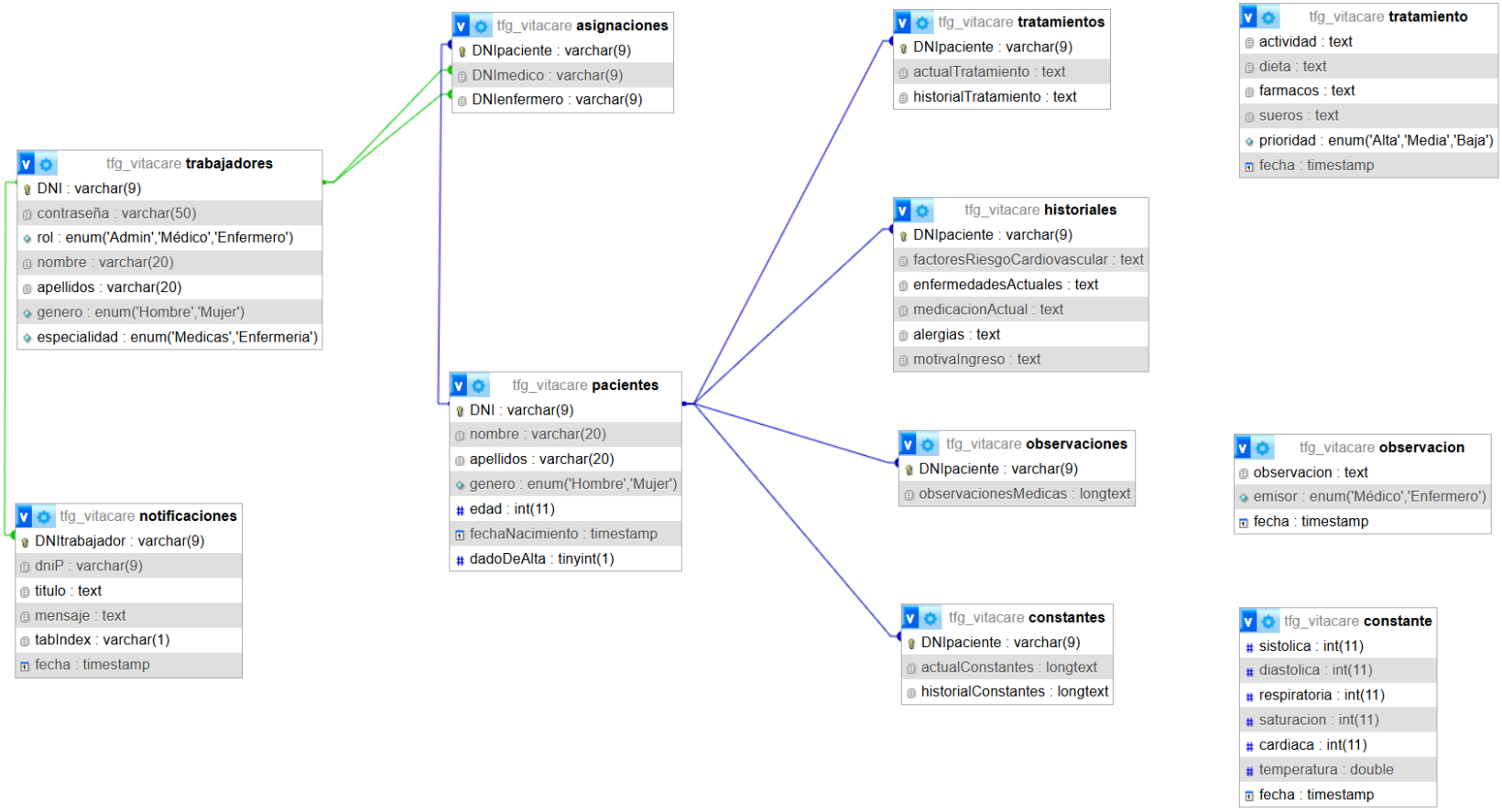


Figura 6-1. Diagrama entidad-relación base de datos

Dado que Firestore es un sistema de gestión de bases de datos no relacional no se puede exportar la estructura que tiene, por lo que optamos por utilizar MySQL [15] para poder realizar el diagrama de entidad-relación que hemos seguido, intentando representarlo lo más parecido posible con el tipo de cada campo, pero en algún caso no fue posible, cómo es con las colecciones tratamiento, observaciones y constantes.

Más adelante detallaremos los tipos de los campos de cada colección, pero en el caso de las colecciones nombradas anteriormente, estaban estructuradas con mapas y arrays que contenían otra estructura internamente. Estas estructuras son las especificadas en las tablas próximas a cada una de ellas.

6.1 Colección Trabajadores

Descripción: almacena la información de todos los trabajadores registrados en la aplicación.

Campos del documento:

- DNI (PK) [string]: identificador del trabajador.
- contraseña [string]: contraseña hasheada del trabajador.
- rol [enum {admin, médico, enfermero}]: rol que tiene el trabajador en la aplicación.
- nombre [string]: nombre del trabajador.
- apellidos [string]: apellidos del trabajador.
- genero [enum {hombre, mujer}]: género del trabajador.
- especialidad [array:
    - medicina[Alergología, Análisis Clínicos, Anatomía Patológica, Anestesiología y Reanimación, Angiología y Cirugía Vascular, Aparato Digestivo, Bioquímica Clínica, Cardiología, Cirugía Cardiovascular, Cirugía General y del Aparato Digestivo, Cirugía Oral y Maxilofacial, Cirugía Ortopédica y Traumatología, Cirugía Pediátrica, Cirugía Plástica, Estética y Reparadora, Cirugía Torácica, Dermatología

Médico-Quirúrgica, Endocrinología y Nutrición, Farmacología Clínica, Geriatría, Hematología y Hemoterapia, Inmunología, Medicina del Trabajo, Medicina Familiar y Comunitaria, Medicina Física y Rehabilitación, Medicina Intensiva, Medicina Interna, Medicina Legal y Forense, Medicina Nuclear, Medicina Preventiva y Salud Pública, Microbiología y Parasitología, Nefrología, Neumología, Neurocirugía, Neurofisiología Clínica, Neurología, Obstetricia y Ginecología, Oftalmología, Oncología Médica, Oncología Radioterápica, Otorrinolaringología, Pediatría y Áreas Específicas, Psiquiatría, Psiquiatría Infantil y de la Adolescencia, Radiodiagnóstico, Reumatología, Urología]

  - enfermería [Enfermería Obstétrico-Ginecológica, Enfermería de Salud Mental, Enfermería Geriátrica, Enfermería del Trabajo, Enfermería Familiar y Comunitaria, Enfermería Pediátrica]:

especialidad del profesional sanitario.

6.2 Colección Pacientes

Descripción: contiene la información de todos los pacientes que han sido registrados en la aplicación por un ingreso.

Campos del documento:

- DNI (PK) [string]: identificador del paciente.
- nombre [string]: nombre del paciente.
- apellidos [string]: apellidos del paciente.
- genero [enum {hombre, mujer}]: género del paciente.
- edad [number]: edad del paciente.
- fechaNacimiento [timestamp]: fecha de nacimiento del paciente.
- dadoDeAlta [boolean]: indica si el paciente ha sido dado de alta o no.

6.3 Colección Asignaciones

Descripción: gestiona las asignaciones entre médico-paciente-enfermero que se hacen durante el ingreso del paciente.

Campos del documento:

- DNIpaciente (PK/FK) [string]: identificador del paciente para asociarlo a las asignaciones de médico y enfermero que le tratan.
- DNImedico (FK) [string]: identificador del médico que trata la paciente.
- DNIenfermero (FK) [string]: identificador del enfermero que trata al paciente.

6.4 Colección Historiales

Descripción: almacena la información correspondiente al historial clínico de cada paciente.

Campos del documento:

- DNIpaciente (PK/FK) [string]: identificador del paciente para asociarlo a su historial clínico.
- factoresRiesgoCardiovascular [string]: factores de riesgos cardiovascular que haya sufrido el paciente o si sus antepasados han tenido algún problema cardiovascular.
- enfermedadesActuales [string]: enfermedades que padezca el paciente.
- medicacionActual [string]: medicación que toma el paciente.
- alergias [string]: cualquier alergia que tenga el paciente que se deba conocer.
- motivoIngreso [string]: motivo de ingreso por el cual el paciente está hospitalizado.

6.5 Colección Tratamientos

Descripción: contiene los tratamientos que han sido creados para cada paciente durante su ingreso.

Campos del documento:

- DNIpaciente (PK/FK) [string]: identificador del paciente para asociarlo a su tratamiento.
- actualTratamiento [mapa]: mapa del tratamiento actual que tiene el paciente, contiene la información de cada tratamiento y una marca temporal de cuando se hicieron los cambios. A continuación, se detallan los campos que contiene el mapa:
    - sueros [string]: sueros que se deben administrar al paciente.
    - farmacos [string]: fármacos que se deben administrar al paciente y su pauta.
    - dieta [string]: dieta que debe seguir el paciente mientras esté hospitalizado.
    - actividad [string]: actividad que puede o debe hacer el paciente (inmovilizado, encamado, estar sentado, andar…)
    - prioridad [enum {alta, media, baja)]: prioridad del tratamiento del paciente.
    - fecha [timestamp]: fecha de cuando se han realizado un cambio en el tratamiento.
- historialTratamiento [array]: array de mapas de los tratamientos que ha tenido el paciente, en donde se ve reflejado los distintos tratamientos que ha tenido y así poder hacer un seguimiento más controlado de ello. Cada elemento del array es un mapa que contiene los mismos campos que el mapa actualTratamiento.

6.6 Colección Constantes

Descripción: almacena los valores de las constantes vitales cada vez que han sido tomadas al paciente.

Campos del documento:

- DNIpaciente (PK/FK) [string]: identificador del paciente para asociarlo a sus constantes vitales.
- actualConstantes [mapa]: mapa que contiene los valores de cada constante vital y una marca temporal de cuando se hicieron los cambios. A continuación, se detallan los campos que contiene el mapa:
    - tensionArterialSistolica [number]: refleja la presión de la sangre en las paredes arteriales cuando el corazón se contrae. Parámetros normales: 110-140 mm Hg. Representación en aplicación: SYS mmHg.
    - tensionArterialDiastolica [number]: refleja la presión de la sangre en las paredes arteriales cuando el corazón se relaja. Parámetros normales: 70-90 mm Hg. Representación en aplicación: DIA mmHg.
    - frecuenciaCardiaca [number]: indica cuántas pulsaciones hace el corazón por minuto. Parámetros normales: 60-100 lpm (latidos por minuto). Representación en aplicación: PR/min (Pulse Rate).
    - frecuenciaRespiratoria [number]: indica las respiraciones por minuto. Parámetros normales: 12-20 respiraciones por minuto. Representación en aplicación: RR/min (Respiration Rate).
    - saturacionOxigeno [number]: refleja cuánto oxígeno tenemos en la sangre. Parámetros normales. 95-100%. Representación en aplicación: %Sp02.
    - temperaturaCorporal [number]: indica la temperatura corporal. Parámetros normales: 35'8-37 ºC. Representación en aplicación: Temp ºC.

    - fecha [timestamp]: fecha de cuando se ha realizado un cambio en las constantes vitales.

- historialConstantes [array]: array de mapas de las constates vitales del paciente en donde se ve reflejado el progreso del paciente. Cada elemento del array es un mapa que contiene los mismos campos que el mapa actualConstantes.

6.7 Colección Observaciones

Descripción: contiene la información de las observaciones que han sido hechas por el médico o el enfermero acerca del paciente.

Campos del documento:

- DNIpaciente (PK/FK) [string]: identificador del paciente para asociarlo a sus observaciones médicas.
- observacionesMedicas [array]: array de mapas de las observaciones médicas que se van haciendo sobre el paciente. A continuación, se detallan los campos que contiene cada mapa:
    - emisor [enum {Médico, Enfermero}]: indica el profesional sanitario que ha anotado dicha observación médica.
    - observación [string]: descripción de la observación médica que ha sido anotada.
    - fecha [timestamp]: fecha de cuando se anotó la observación médica.

6.8 Colección Notificaciones

Descripción: gestiona las notificaciones que llegan a los profesionales sanitarios.

Campos del documento:

- DNItrabajador (PK/FK) [string]: identificador del trabajador para asociarlo a sus notificaciones.

- idMapa [mapa]: mapa que contiene la información de cada notificación que ha recibido el profesional sanitario y una marca temporal de cuando lo recibió. A continuación, se detallan los campos que contiene cada mapa de la notificación:
    - dniP [string]: identificador del paciente del que proviene la notificación.
    - titulo [string]: título que indica qué tipo de notificación es. Puede ser cualquiera de las siguientes, "*ALERTA*", "*NUEVO TRATAMIENTO*", "*TRATAMIENTO EDITADO*", "*NUEVAS OBSERVACIONES*", "*NUEVAS CONSTNTES VITALES*" y "*NUEVO PACIENTE*".
    - mensaje [string]: información de interés relacionado con la notificación.
    - tabIndex [string]: indicativo del fragmento de la información del paciente al que debe redireccionar la notificación cuando se pulse sobre ella.
    - fecha [timestamp]: fecha de cuando se creó la notificación.

Capítulo 7 - Diseño

VitaCare es una aplicación móvil destinada a la gestión de información de pacientes ingresados en un hospital. Facilita el intercambio de datos entre médicos y enfermeros, permitiendo que ambos profesionales puedan estar informados en todo momento de la evolución de sus pacientes. Desde el inicio del proyecto, se estableció como prioridad crear un diseño usable, intuitivo y accesible, pensando en que los usuarios no tendrían necesariamente conocimientos avanzados de informática.

Durante el proceso de diseño, se buscó un color que reflejara los valores esenciales de un entorno hospitalario. Por este motivo, se eligió como color principal el verde en diferentes tonos, el cual se asocia a naturaleza, vida y crecimiento, trasmitiendo tranquilidad, reduciendo el estrés y la ansiedad; combinado con el blanco que refleja limpieza y claridad visual.

Verde principal: #0F666D

Otros verdes: #D3F6EF, #1CBAC5, #13838B

Blanco: #FFFFFFFF

En cuanto a la imagen de nuestra aplicación, se diseñó un logo sencillo pero representativo. Para el icono principal se utilizaron las iniciales VitaCare, buscando que fuera simple y fácil de reconocer, figura 7-1. Para la pantalla de carga, se diseñó un logotipo en el que se muestra el nombre completo como se ve en la figura 7-2.

Figura 7-1. Logo aplicación

Figura 7-2. Pantalla de carga aplicación

VitaCare está concebida para que los administradores, médicos y enfermeros pueden utilizarla de forma sencilla, sin necesidad de formación previa ni manuales de uso. En el módulo destinado al personal sanitario, la pantalla principal muestra por orden

de prioridad de tratamiento todos los pacientes asignados, de manera que, al pulsar en alguno de ellos, se visualizan los datos personales, el historial clínico, los tratamientos o las constantes vitales. Además, disponen de una sección de notificaciones donde se almacenan todas las actualizaciones y cambios relativos a los pacientes, permitiéndoles estar actualizados con respecto a su evolución.

En el caso del administrador, la navegación es igual de sencilla, desde su pantalla de inicio, a través de dos botones bien diferenciados, puede acceder a las funcionalidades principales.

Los formularios que se encuentran en la aplicación están correctamente identificados con su título correspondiente, y cuando es necesario, se especifica el formato de los datos a introducir. Algunos campos del formulario permiten seleccionar entre opciones predeterminadas o utiliza sistemas de autocompletado basados en las letras que el usuario introduce, lo que facilita el proceso de entrada de información.

Los usuarios tienen a su disposición un menú lateral que les muestra todas las opciones disponibles a la hora de navegar por la aplicación. Además, tienen la opción de cerrar sesión de forma rápida y segura.

Con el objetivo de minimizar errores y mejorar la experiencia de uso del usuario, las acciones que conllevan crear, modificar o eliminar datos de la aplicación, requieren una confirmación previa mediante un cuadro de diálogo. Si se produjese algún error al introducir datos, se muestran mensajes informativos que permiten al usuario corregirlos fácilmente. Gracias a estas medidas, nuestro proyecto demuestra ser una aplicación accesible, intuitiva y segura, diseñada para funcionar de manera óptima en un entorno hospitalario donde la fiabilidad, la rapidez y la facilidad de uso son imprescindibles.

Capítulo 8 - Implementación

A continuación, explicaremos las funcionalidades que hemos implementado en nuestro proyecto, organizado en diferentes módulos para una mayor claridad.

8.1 Gestión de cuentas

8.1.1 Login

La funcionalidad de Login permite a los usuarios que se encuentren previamente en nuestra base de datos registrados, acceder a todas las herramientas de VitaCare. Para iniciar sesión, el usuario debe introducir su NIF, identificador único de cada uno dentro del sistema, junto a la contraseña previamente establecida. Esto garantiza que cada usuario tenga un acceso único y seguro. Si los datos introducidos son incorrectos, el sistema muestra una serie de mensajes para ayudar al usuario a saber cuál es el problema y corregirlo. Por ejemplo, si alguno de los campos se ha dejado vacío, el NIF no se encuentra en nuestra base de datos, si su formato no es correcto o la contraseña no coincide con la que está asociado al NIF del trabajador. Dichos mensajes informativos mejoran la experiencia del usuario ofreciendo una retroalimentación clara y comprensible.

Como vemos en la figura 8-1, a la derecha del campo de la contraseña se encuentra un botón que permite al usuario alternar entre mostrar u ocultar la contraseña mientras escribe.

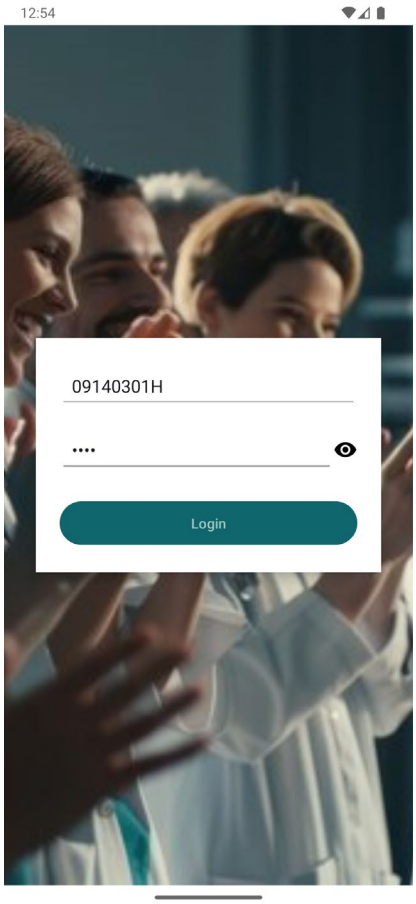


Figura 8-1. Pantalla de login

En las figuras 8-2 y 8-3, se muestra la funcionalidad del botón para ocultar y mostrar el campo de contraseña.

```
showPasswordLoginButton.setOnClickListener(v -> {
    isPasswordLoginVisible = !isPasswordLoginVisible;
    PasswordVisibility.togglePasswordVisibility(passwordText, showPasswordLoginButton, isPasswordLoginVisible);
});
```

Figura 8-2. Implementación para ocultar contraseña 1

```
public static void togglePasswordVisibility(EditText passwordField, ImageButton toggleButton, boolean isVisible) {
    if (isVisible) { //Mostrar contraseña
        passwordField.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
        toggleButton.setImageResource(R.drawable.baseline_visibility_off_24); //Ojo abierto
    } else { //Ocultar contraseña
        passwordField.setTransformationMethod(PasswordTransformationMethod.getInstance());
        toggleButton.setImageResource(R.drawable.baseline_visibility_24); //Ojo cerrado
    }
    passwordField.setSelection(passwordField.getText().length());
}
```

Figura 8-3. Implementación para ocultar contraseña 2

Una vez introducidos los datos, el sistema los valida consultando la base de datos para localizar al trabajador mediante su NIF como se observa en la figura 8-4. Si el trabajador existe y la contraseña coincide con la almacenada se procede a guardar los datos del profesional. Para ello, utilizamos el sistema de SharedPreferences, una herramienta propia de Android que permite guardar información en formato "clave-valor" de forma persistente. La sesión permanece abierta, aunque se cierre la aplicación o se apague el dispositivo móvil. Además, esta herramienta nos proporciona la ventaja de poder consultar los datos desde cualquier parte del proyecto, por ejemplo, el rol, el nombre y los apellidos del usuario, entre otros.

```java
public void verificarCredenciales(String DNI, String password, Callbacks.OnLoginResultCallback callback) {

    DocumentReference trabajadores = fdb.collection( collectionPath: "trabajadores").document(DNI);

    trabajadores.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            DocumentSnapshot trabajador = task.getResult();
            if (trabajador.exists()) {
                String passwordHashed = trabajador.getString( field: "contraseña");

                if (HashUtils.checkPassword(password, passwordHashed)) {
                    // Guarda el estado de sesión y el rol en SharedPreferences
                    editor.putBoolean("isLoggedIn", true);
                    editor.putString("userRole", trabajador.getString( field: "rol"));
                    editor.putString("userDNI", DNI);

                    editor.putString("userNombre", trabajador.getString( field: "nombre"));
                    editor.putString("userApellidos", trabajador.getString( field: "apellidos"));
                    editor.putString("userEspecialidad", trabajador.getString( field: "especialidad"));
                    editor.putString("userGénero", trabajador.getString( field: "genero"));
                    editor.apply();

                    // Notifica el éxito de la autenticación
                    callback.onSuccess(trabajador.getString( field: "rol"));
                } else {
                    callback.onError("Contraseña incorrecta");
                }
            } else {
                callback.onError("No existe el trabajador");
            }
        } else {
            callback.onError("Error al obtener los datos");
        }
    });
}
```

Figura 8-4. Implementación para verificar datos login

Cuando los datos ya han sido validados, se realiza una redirección automática a la pantalla correspondiente dependiendo de su rol como se muestra en la figura 8-5.

```
private void redirigirPorRol(String role) {  2 usages
    Intent intent;

    if ("Admin".equals(role)) {
        intent = new Intent( packageContext: Login.this, AdminActivity.class);
    } else {
        createNotificationChannel();
        suscribeToTopicUser(fbd.getLoggedInWorker().getDNI());
        intent = new Intent( packageContext: Login.this, DoctorActivity.class);
    }
    startActivity(intent);
}
```

Figura 8-5. Implementación para redirigir por rol

En el caso de ser un profesional sanitario, se lleva a cabo todo lo relacionado con el sistema de notificaciones. Primeramente, como se observa en la figura 8-6, se crea un canal de notificaciones que permite clasificar los distintos tipos de mensajes que puede recibir el usuario y, al mismo tiempo, ofrece al propio usuario el control sobre cómo quiere recibir esas notificaciones. Se utiliza para gestionar los mensajes que el médico puede enviar al enfermero asignado y viceversa, facilitando una comunicación ágil y efectiva entre profesionales sanitarios.

```
private void createNotificationChannel() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        NotificationChannel channel = new NotificationChannel(
                CHANNEL_ID,
                name: "VitaCare Notifications",
                NotificationManager.IMPORTANCE_HIGH);
        channel.setDescription("Canal para notificaciones de VitaCare");

        NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
        if (notificationManager != null) {
            notificationManager.createNotificationChannel(channel);
        }
    }
}
```

Figura 8-6. Implementación para crear canal de notificaciones

Además, se realiza una suscripción automática a un tema de Firebase Cloud Messaging (FCM) utilizando el NIF del trabajador como identificador del tema. Esto permite que las notificaciones enviadas desde el servidor lleguen exclusivamente al destinatario deseado. Es una implementación sencilla, reflejada en la figura 8.7, que nos permite la comunicación entre los usuarios de VitaCare.

```
private void suscribeToTopicUser(String DNI) {  1 usage
    FirebaseMessaging.getInstance().subscribeToTopic("user_" + DNI)
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    Log.d( tag: "Suscripción",  msg: "Suscrito al tema user_" + DNI);
                } else {
                    Log.e( tag: "Suscripción",  msg: "Error al suscribirse al tema user_" + DNI, task.getException());
                }
            });
}
```

Figura 8-7. Implementación para suscribirse a tema

Una vez el inicio de sesión se ha completado con éxito, el usuario no necesita realizar este proceso cada vez que entre en la aplicación. Mientras no se cierre sesión de forma manual, se mantendrá activa.

8.1.2 Logout

Cuando el usuario desea cerrar sesión, debe acceder al menú lateral, donde se encuentra la opción correspondiente como se observa en la figura 8-8. Al pulsar en ella, aparece por pantalla un diálogo de confirmación mostrado en la figura 8-9, que solicita al usuario la validación de su decisión. Directamente, el sistema redirige al usuario a la pantalla de Login.

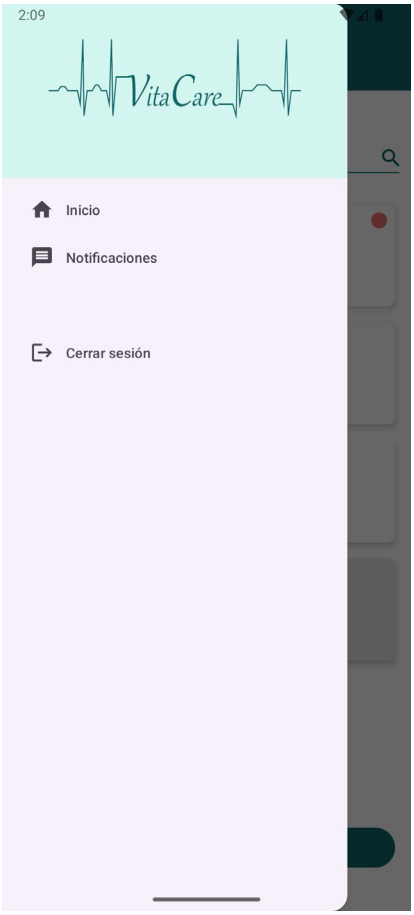


Figura 8-8. Menú lateral del profesional sanitario

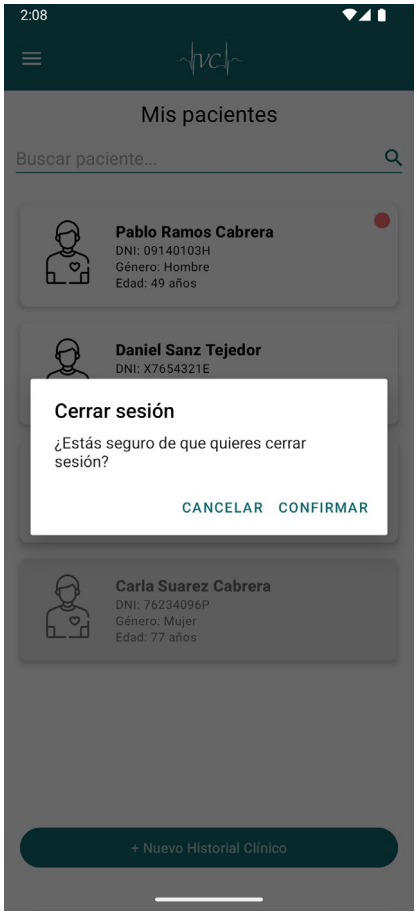


Figura 8-9. Confirmación para cerrar sesión

Tal y como se muestra en la figura 8-10, el sistema procede a desuscribir al usuario del tema al que fue asociado durante el inicio de sesión. Esto asegura que el usuario no reciba notificaciones destinadas a su cuenta cuando no esté autenticado, lo cual protege su privacidad y evita posibles inconsistencias en el sistema de notificaciones.

Luego, el sistema limpia todos los datos de sesión almacenados localmente a través de SharedPreferences. Esta acción elimina cualquier rastro del usuario anterior y garantiza que, si la aplicación se vuelve a abrir posteriormente, no se acceda automáticamente sin una nueva autenticación. Es decir, la pantalla de inicio de sesión se vuelve a mostrar como punto de partida.

```
} else if (id == R.id.logout) {
    String userDNI = preferences.getString( key: "userDNI",  defValue: "");

    // Desuscribirse al usuario del topic
    FirebaseMessaging.getInstance().unsubscribeFromTopic("user_" + userDNI)
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    Log.d( tag: "Desuscripción",  msg: "Desuscrito al tema user_"+ userDNI);
                } else {
                    Log.e( tag: "Desuscripción",  msg: "Error al desuscribirse al tema user_"+ userDNI, task.getException());
                }
            });

    // Limpiar sesión
    SharedPreferences.Editor editor = preferences.edit();
    editor.clear();
    editor.apply();

    // Redirigir al login
    Intent intent = new Intent( packageContext: this, Login.class);
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(intent);

    Toast.makeText( context: this,  text: "Sesión cerrada", Toast.LENGTH_SHORT).show();
}
```

Figura 8-10. Implementación para logout

8.2 Gestión del administrador

Cuando el Admin inicia sesión, en su pantalla principal encuentra las opciones de visualizar todos los trabajadores y pacientes al igual que en su menú lateral como se muestra en las figuras 8-11 y 8-12.

Figura 8-11. Pantalla principal del admin

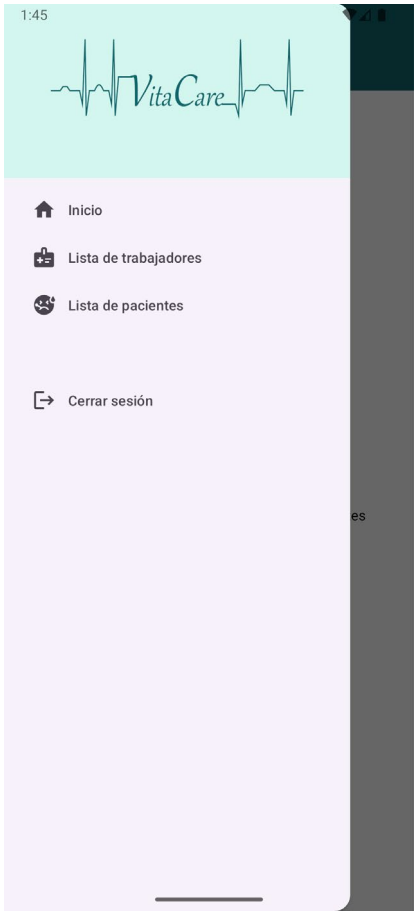


Figura 8-12. Menú lateral del admin

8.2.1 Registrar trabajador

El administrador, como responsable de registrar nuevos trabajadores, debe acceder a la sección de lista de trabajadores y pulsar el botón de "Añadir nuevo trabajador". A continuación, se le muestra un formulario que debe completar con los datos del nuevo profesional sanitario, tal como se observa en las figuras 8-13 y 8-14.

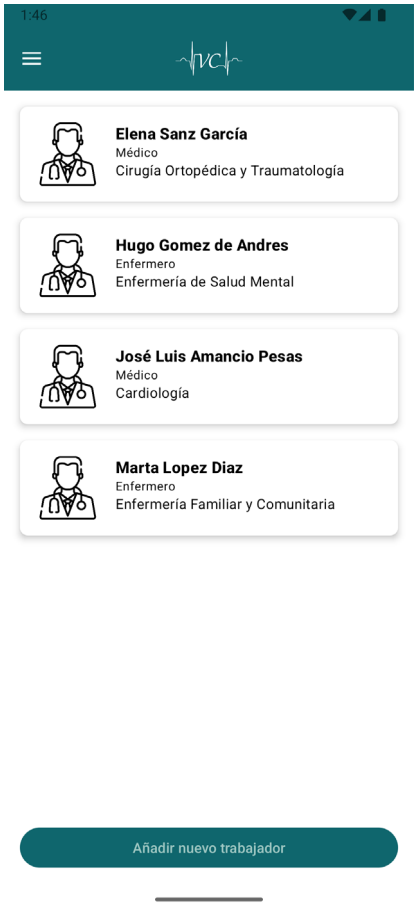


Figura 8-13. Lista de trabajadores

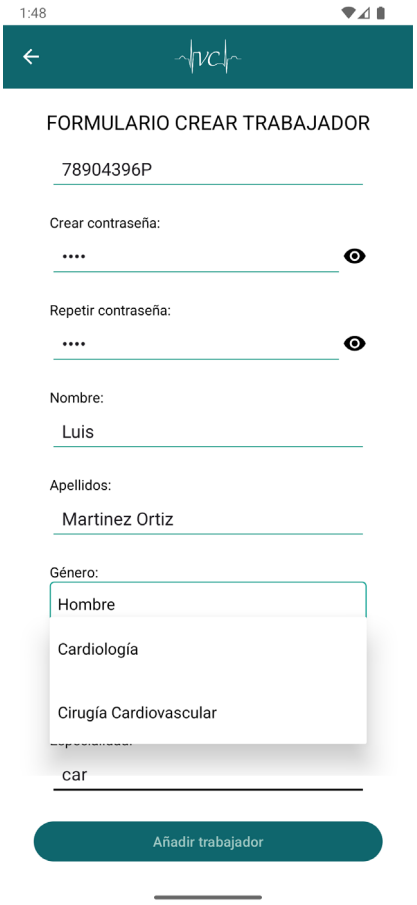


Figura 8-14. Formulario para añadir trabajador

Una vez rellenados todos los campos del formulario, el sistema procede a validar los datos introducidos. Es importante destacar que todos los campos son obligatorios, por lo que, en caso de dejar alguno vacío, se muestra un mensaje de error informativo que ayuda al administrador a corregirlo. El primer paso en la validación consiste en comprobar que no exista otro trabajador con el mismo NIF, ya que actúa como identificador único de cada usuario de nuestro sistema. Esta lógica queda reflejada en la figura 8-15.

```
public void workerExists(String DNI, Callbacks.WorkerExistenceCallback callback) {  2 usages
    fdb.collection( collectionPath: "trabajadores").document(DNI) DocumentReference
            .get() Task<DocumentSnapshot>
            .addOnSuccessListener(documentSnapshot -> {
                if (documentSnapshot.exists()) {
                    callback.onResult( exists: true);
                } else {
                    callback.onResult( exists: false);
                }
            })
            .addOnFailureListener(e -> {
                callback.onResult( exists: false);
            });
}
```

Figura 8-15. Implementación para comprobar trabajador existente

Por otro lado, los campos de las contraseñas, uno para introducirla y otro para confirmarla, cuentan también con los botones laterales de mostrar o esconder el texto como vimos anteriormente en la pantalla de Login. Durante la validación, se verifica que ambas contraseñas coincidan, y si es el caso, se guarda cifrada en nuestra base de datos, con el fin de garantizar la seguridad y confidencialidad de los datos sensibles del trabajador.

La sección de la especialidad viene condicionada por el rol del usuario, médico o enfermero, mostrando para cada uno las especialidades correspondientes. Todas ellas han sido obtenidas desde una lista oficial obtenida de fuentes institucionales disponibles en internet y almacenadas en nuestro sistema para el uso interno [16].

En caso de que se produzca algún error durante la comprobación de los datos, el sistema informa al admin mediante mensajes claros y específicos, permitiéndole corregir los errores antes de continuar.

Por último, el administrador debe pulsar en "Añadir trabajador", que le muestra un diálogo para la doble confirmación. Una vez aceptado, el nuevo trabajador se añade correctamente a la base de datos con el método mostrado en la figura 8-16 y aparece junto a los demás trabajadores de la lista.

```
public void createWorker(Worker newWorker, Callbacks.CreateCallback callback){  2 usages
    Map<String, Object> dataWorker = new HashMap<>();
    dataWorker.put("contraseña", newWorker.getContraseña());
    dataWorker.put("nombre", newWorker.getNombre());
    dataWorker.put("apellidos", newWorker.getApellidos());
    dataWorker.put("genero", newWorker.getGenero());
    dataWorker.put("rol", newWorker.getRol());
    dataWorker.put("especialidad", newWorker.getEspecialidad());

    fdb.collection( collectionPath: "trabajadores").document(newWorker.getDNI()) DocumentReference
            .set(dataWorker) Task<Void>
            .addOnSuccessListener(aVoid -> {
               callback.onSuccess();
            })
            .addOnFailureListener(e -> {
                callback.onError( errorMessage: "Error al añadir trabajador: " + e.getMessage());
            });
}
```

Figura 8-16. Implementación para crear trabajador

8.2.2 Modificar trabajador

Para editar los datos de un trabajador, el administrador debe deslizar hacia la derecha alguno de elementos de la lista de trabajadores como se observa en la figura 8-17. Esto redirige al admin al formulario de edición donde los campos aparecen previamente completados con la información actual del trabajador seleccionado que se consigue mediante el método de la figura 8-19.

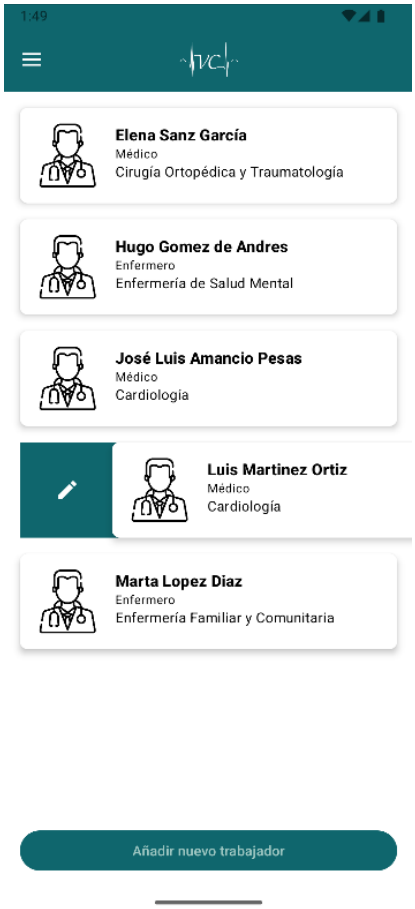


Figura 8-17. Movimiento de deslizar para editar

Como se observa en la figura 8-18, este formulario permite modificar cualquier dato asociado al trabajador, con la excepción de la contraseña, que no se muestra incialmente por motivos de seguridad. Si desea cambiarla, el sistema solicita al admin que la introduzca en dos campos, como en el proceso de registro, asegurando así que ambas coinciden antes de guardar los cambios.

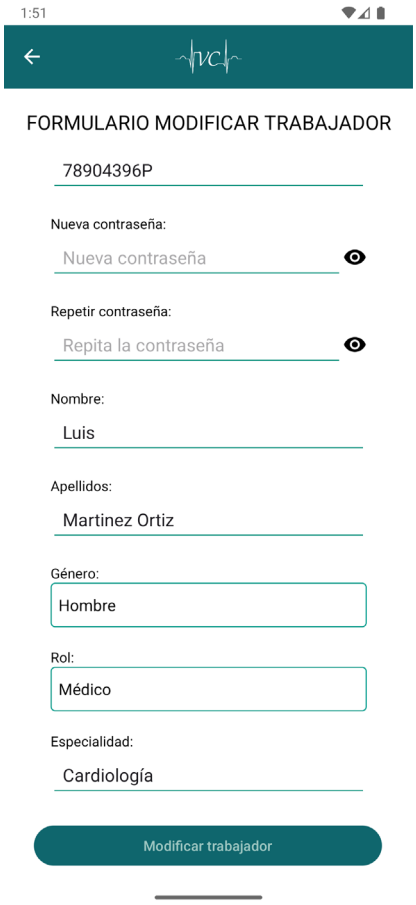


Figura 8-18. Formulario para editar trabajador

```
public void getDataWorker(String DNI, Callbacks.WorkerDataCallback callback) {  1 usage
    fdb.collection( collectionPath: "trabajadores").document(DNI) DocumentReference
            .get() Task<DocumentSnapshot>
            .addOnSuccessListener(documentSnapshot -> {
                if(documentSnapshot.exists()){
                    Worker worker = documentSnapshot.toObject(Worker.class);
                    if (worker != null){
                        worker.setDNI(documentSnapshot.getId());
                        callback.onWorkerDataLoaded(worker);
                    }
                    else{
                        callback.onError( errorMessage: "Error al cargar los datos del trabajador.");
                    }
                }
                else{
                    callback.onError( errorMessage: "No se encontró el trabajador.");
                }
            })
            .addOnFailureListener(e -> {
                callback.onError( errorMessage: "Error al obtener los datos: " + e.getMessage());
            });
}
```

Figura 8-19. Implementación para obtener datos del trabajador

En cuanto a la implementacion de la lógica, el proceso de edición sigue una estructura similar a la del registro. Antes de guardar cualquier cambio, se validan todos los datos introducidos, comprobando que se ajusten al formato esperado y que no haya inconsistencias. Si el dato a modificar es el NIF del trabajador, el sistema primero compara este con los almacenados en la base de datos, en caso de que no exista elimina el documento con el NIF antiguo y crea un nuevo trabajador y su correspondiente documento en la colección trabajadores con el nuevo identificador. Al pulsar "Modificar trabajador", el sistema pide confirmación para proceder a actualizar la base de datos.

8.2.3 Eliminar usuario

Para eliminar un paciente o trabajador, el administrador debe acceder a la lista correspondiente y realizar el movimiento de desplazamiento hacia la izquierda, reflejado en figura 8-21, sobre el elemento que desea eliminar.

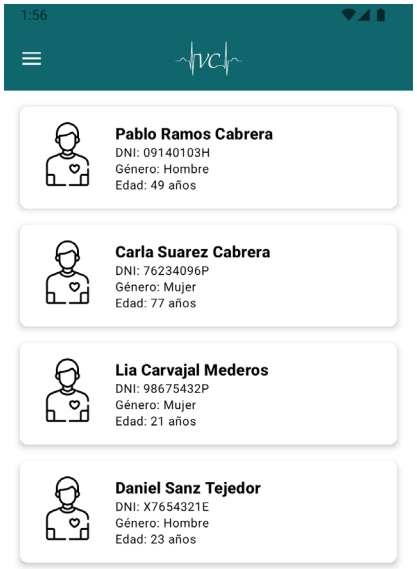


Figura 8-20. Lista de pacientes

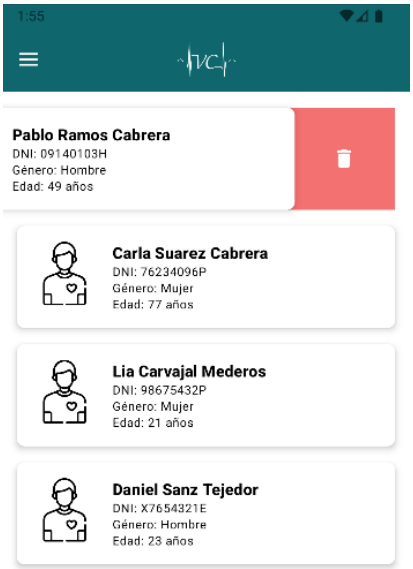


Figura 8-21. Movimiento para eliminar usuario

Antes de eliminarlo de la base de datos permanentemente, el sistema pide una confirmación al administrador, con el fin del evitar eliminaciones accidentales, como se observa en la figura 8-22.

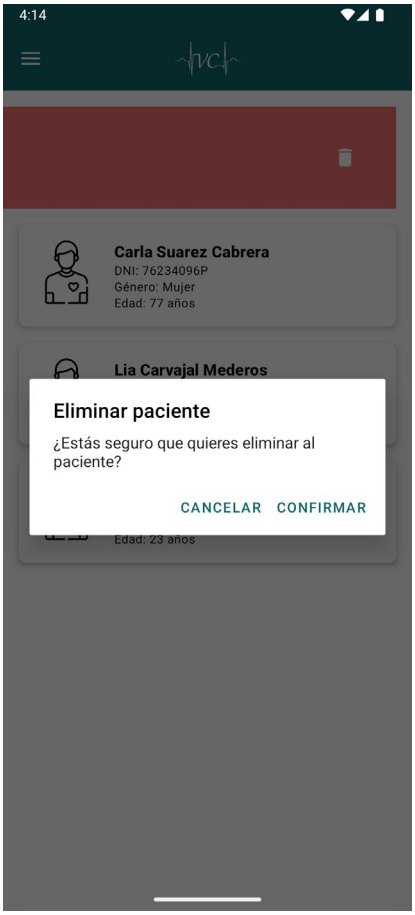


Figura 8-22. Confirmación para eliminar usuario

Si el elemento a eliminar es un paciente, el sistema no solo borra su registro principal, sino que también se encarga de eliminar todos los datos asociados a él, tales como historiales clínicos, tratamientos, constantes vitales y cualquier otra información vinculada tal y como se muestra en la figura 8-23.

```
public void deletePatientAdapter(String patientId, PatientAdapter adapter, List<Patient> patientList, int position){  1 usage
    fdb.collection( collectionPath: "historiales").document(patientId) DocumentReference
            .delete() Task < Void >
            .addOnSuccessListener(aVoid -> {})
            .addOnFailureListener(e -> {
                Toast.makeText(context, text: "Error al eliminar historial clínico del paciente", Toast.LENGTH_SHORT).show();
                adapter.notifyItemChanged(position); // Restablece el elemento en caso de error
            });

    fdb.collection( collectionPath: "asignaciones").document(patientId) DocumentReference
            .delete() Task < Void >
            .addOnSuccessListener(aVoid -> {})
            .addOnFailureListener(e -> {
                Toast.makeText(context, text: "Error al eliminar asignaciones del paciente", Toast.LENGTH_SHORT).show();
                adapter.notifyItemChanged(position); // Restablece el elemento en caso de error
            });

    fdb.collection( collectionPath: "tratamientos").document(patientId) DocumentReference
            .delete() Task < Void >
            .addOnSuccessListener(aVoid -> {})
            .addOnFailureListener(e -> {
                Toast.makeText(context, text: "Error al eliminar tratamiento del paciente", Toast.LENGTH_SHORT).show();
                adapter.notifyItemChanged(position); // Restablece el elemento en caso de error
            });

    fdb.collection( collectionPath: "constantes").document(patientId) DocumentReference
            .delete() Task < Void >
            .addOnSuccessListener(aVoid -> {})
            .addOnFailureListener(e -> {
                Toast.makeText(context, text: "Error al eliminar constantes vitales del paciente", Toast.LENGTH_SHORT).show();
                adapter.notifyItemChanged(position); // Restablece el elemento en caso de error
            });

    fdb.collection( collectionPath: "observaciones").document(patientId) DocumentReference
            .delete() Task < Void >
            .addOnSuccessListener(aVoid -> {})
            .addOnFailureListener(e -> {
                Toast.makeText(context, text: "Error al eliminar observaciones del paciente", Toast.LENGTH_SHORT).show();
                adapter.notifyItemChanged(position); // Restablece el elemento en caso de error
            });

    fdb.collection( collectionPath: "pacientes").document(patientId) DocumentReference
            .delete() Task < Void >
            .addOnSuccessListener(aVoid -> {
                patientList.remove(position);
                adapter.notifyItemRemoved(position);
                Toast.makeText(context, text: "Paciente eliminado", Toast.LENGTH_SHORT).show();
            })
            .addOnFailureListener(e -> {
                Toast.makeText(context, text: "Error al eliminar paciente", Toast.LENGTH_SHORT).show();
                adapter.notifyItemChanged(position); // Restablece el elemento en caso de error
            });
}
```

Figura 8-23. Implementación para eliminar pacientes

Por lo contrario, si es un médico o enfermero, el sistema lo elimina de la colección de trabajadores, siempre y cuando, no tenga pacientes asignados, figura 8-24.

```
public void deleteWorkerAdapter(String workerId, WorkerAdapter adapter, List<Worker> workerList, int position) {  1 usage
    CollectionReference asignacionesRef = fdb.collection( collectionPath: "asignaciones");

    asignacionesRef.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            boolean asignado = false;
            for (DocumentSnapshot asignacion : task.getResult()) {
                if(asignacion.getString( field: "DNImedico").equals(workerId) || asignacion.getString( field: "DNIenfermero").equals(workerId)){
                    asignado = true;
                }
            }
            if(asignado){
                Toast.makeText(context,  text: "El trabajador tiene pacientes asignados", Toast.LENGTH_SHORT).show();
            }
            else{
                fdb.collection( collectionPath: "trabajadores").document(workerId) DocumentReference
                        .delete() Task<Void>
                        .addOnSuccessListener(aVoid -> {
                            workerList.remove(position);
                            adapter.notifyItemRemoved(position);
                            Toast.makeText(context,  text: "Trabajador eliminado", Toast.LENGTH_SHORT).show();
                        })
                        .addOnFailureListener(e -> {
                            Toast.makeText(context,  text: "Error al eliminar trabajador", Toast.LENGTH_SHORT).show();
                            adapter.notifyItemChanged(position); // Restablece el elemento en caso de error
                        });
            }
        } else {
            Toast.makeText(context,  text: "Error al obtener asignaciones", Toast.LENGTH_SHORT).show();
        }
    });
}
```

Figura 8-24. Implementación para eliminar trabajador

8.3 Gestión de pacientes

8.3.1 Listar pacientes asignados

El listar los pacientes asignados es una funcionalidad implícita en el sistema que permite a los sanitarios ver los pacientes. Cuando el profesional sanitario inicia sesión le lleva a la pantalla principal de la aplicación en donde se muestra un listado en el que están todos los pacientes que tiene asignados como se muestra en la figura 8-25.

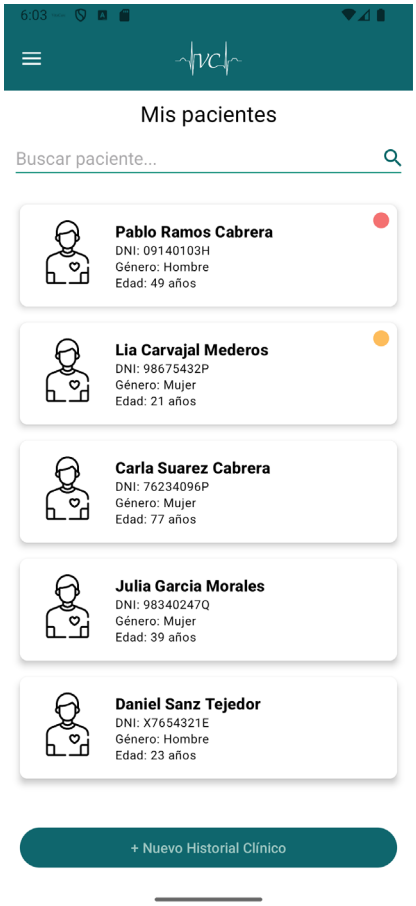


Figura 8-25. Listado de pacientes asignados

La implementación de la lógica, como se ve en la figura 8-26, consulta la base de datos para buscar el NIF del sanitario en la colección de asignaciones y si tiene algún paciente asignado se recopila la información de estos por sus NIFs.

```
public void getPatientsListForWorker(String worker, String DNIworker, PatientAdapter adapter, List<Patient> patientList) {
    CollectionReference asignaciones = fdb.collection( collectionPath: "asignaciones");
    CollectionReference pacientes = fdb.collection( collectionPath: "pacientes");
    CollectionReference tratamientos = fdb.collection( collectionPath: "tratamientos"); // Referencia a la colección de tr

    // Buscamos asignaciones en las que esté el DNI del trabajador
    asignaciones.whereEqualTo(worker, DNIworker).get().addOnCompleteListener(task -> {
        if (task.isSuccessful() && task.getResult() != null) {
            patientList.clear();
            adapter.notifyDataSetChanged();

            // Obtenemos lista de DNI de pacientes asignados al trabajador
            List<String> DNIpacientes = new ArrayList<>();
            for (DocumentSnapshot asignacion : task.getResult()) {
                String DNIpaciente = asignacion.getId();
                if (DNIpaciente != null) {
                    DNIpacientes.add(DNIpaciente);
                }
            }

            if (DNIpacientes.isEmpty()) {
                adapter.notifyDataSetChanged();
                return;
            }

            // Obtener pacientes de la colección de pacientes
            pacientes.whereIn(FieldPath.documentId(), DNIpacientes).get().addOnCompleteListener(patientTask -> {
                if (patientTask.isSuccessful() && patientTask.getResult() != null) {
                    for (DocumentSnapshot paciente : patientTask.getResult()) {
                        Patient patient = paciente.toObject(Patient.class);
                        if (patient != null) {
                            patient.setDNI(paciente.getId());

                            // Comprobamos si el paciente tiene un tratamiento asignado
                            tratamientos.document(patient.getDNI()).get()
                                    .addOnSuccessListener(tratamientoSnapshot -> {...})
                                    .addOnFailureListener(e -> {
                                        Log.e( tag: "Error", msg: "Error al obtener el tratamiento: " + e.getMessage());
                                        patient.setTratamiento(new Treatment( prioridad: null, sueros: "", farmacos:
                                        patientList.add(patient);
                                    });
                        }
                    }
                } else {
                    Toast.makeText(context, text: "Error al obtener pacientes", Toast.LENGTH_SHORT).show();
                }
            });

        } else {
            Toast.makeText(context, text: "Error al obtener asignaciones del trabajador", Toast.LENGTH_SHORT).show();
        }
    });
}
```

Figura 8-26. Implementación para obtener lista de pacientes asignados

8.3.2 Buscar paciente

Los profesionales sanitarios pueden buscar a los pacientes mediante la barra de búsqueda que está localizada en el extremo superior del listado de pacientes como se muestra en la figura 8-27. Para ello, introducen parte del nombre o apellidos del paciente y de forma dinámica se actualiza la pantalla y se muestran los pacientes que coinciden con los resultados de la búsqueda.

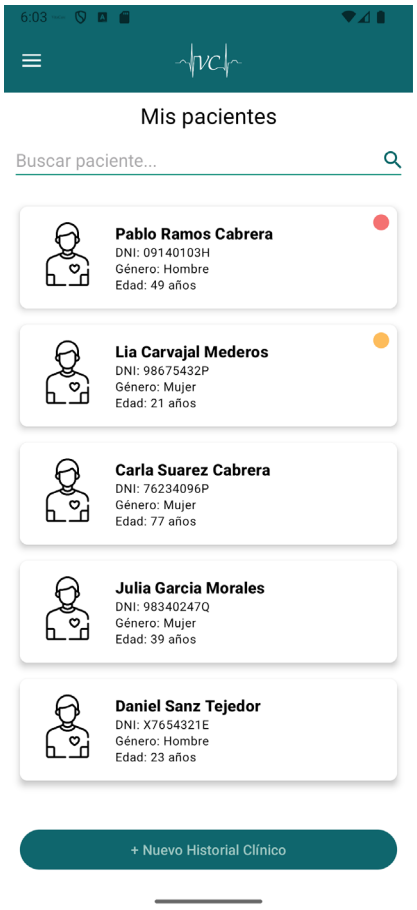


Figura 8-27. Listado de pacientes

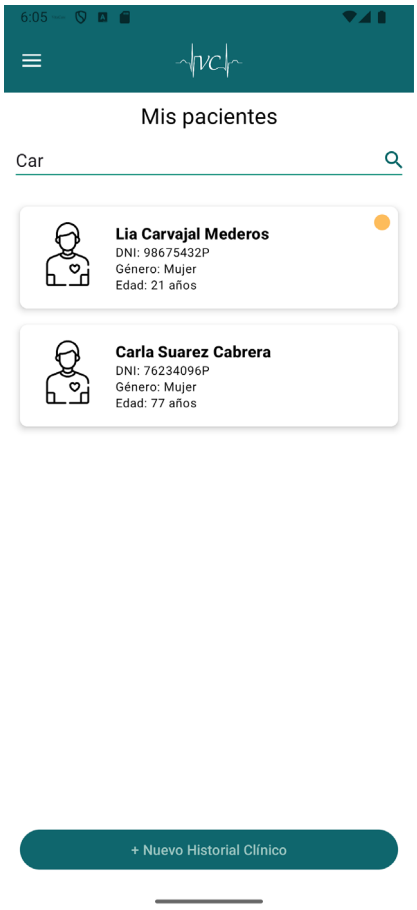


Figura 8-28. Listado de pacientes actualizado tras búsqueda

Como se puede observar en la figura 8-28, con cada cambio que se produzca en el espacio de texto reservado para la búsqueda del paciente, hay un listener que detecta los cambios y llama a un método externo. Este método que queda reflejado en

la figura 8-29, se encarga de hacer una búsqueda en la lista de los pacientes que tiene asignados el profesional sanitario.

Los pacientes con los que hay cualquier coincidencia entre los caracteres introducidos y los nombres y/o apellidos de estos se muestran por pantalla. De esta forma, cada vez que se modifica el campo de búsqueda, la lista de pacientes mostrada por pantalla se actualiza reflejando los resultados, figura 8-30.

```
EditText searchEditText = findViewById(R.id.nombrePaciente);
searchEditText.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        filterPatients(s.toString());
    }

    @Override
    public void afterTextChanged(Editable s) {}
});
```

Figura 8-29. Implementación para buscar paciente en tiempo real

```
private void filterPatients(String query) {  1 usage
    List<Patient> filteredList = new ArrayList<>();
    String lowerCaseQuery = query.toLowerCase();

    for (Patient patient : patientListForWorker) {
        String fullName = (patient.getNombre() + " " + patient.getApellidos()).toLowerCase(); // Combinar nombre y apellidos
        if (fullName.contains(lowerCaseQuery)) {
            filteredList.add(patient);
        }
    }

    patientAdapter.updateList(filteredList);
}
```

Figura 8-30. Implementación para actualizar listado con resultados búsqueda

8.3.3 Crear historial clínico del paciente

Cuando un médico recibe un nuevo paciente, el primer paso para iniciar su seguimiento clínico es crear un nuevo historial clínico donde ponga todo lo relacionado con él. En la figura 8-31, se muestra la lista de los pacientes que tiene asignados y, además, dispone del botón "Nuevo Historial Clínico", desde el cual se accede al formulario correspondiente para registrar un nuevo historial.

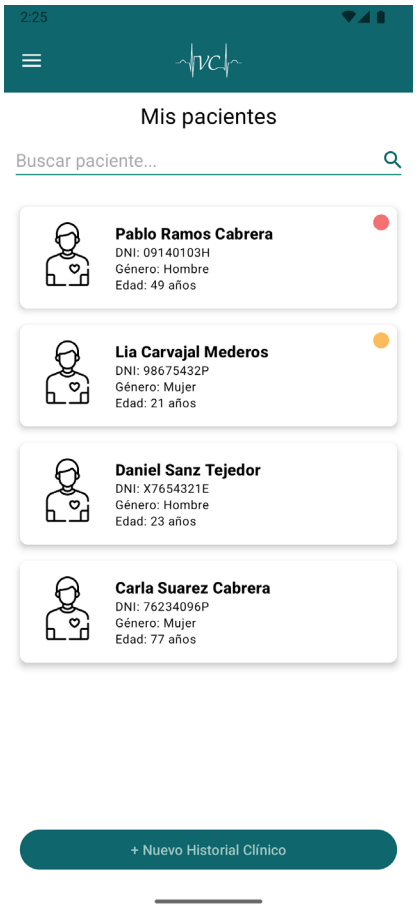


Figura 8-31. Pantalla principal del médico

Al pulsarlo, aparece en pantalla un formulario, figuras 8-32 y 8-33, para crear el nuevo historial, el cual solicita tanto los datos personales como cierta información médica relevante.

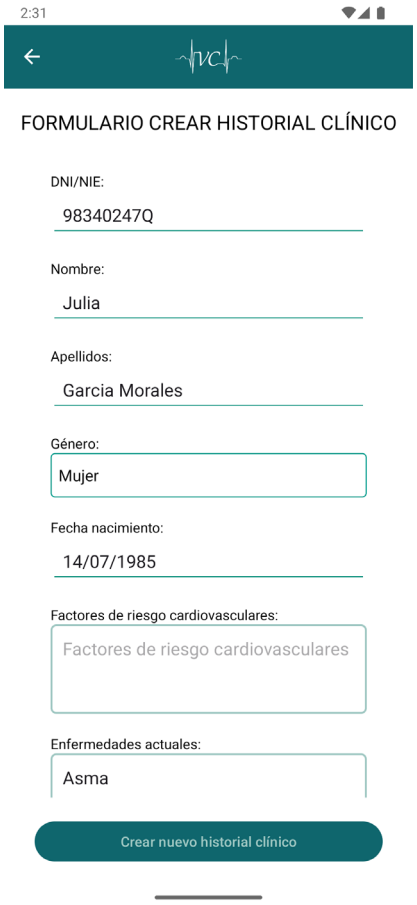


Figura 8-32. Formulario para crear historial clínico 1

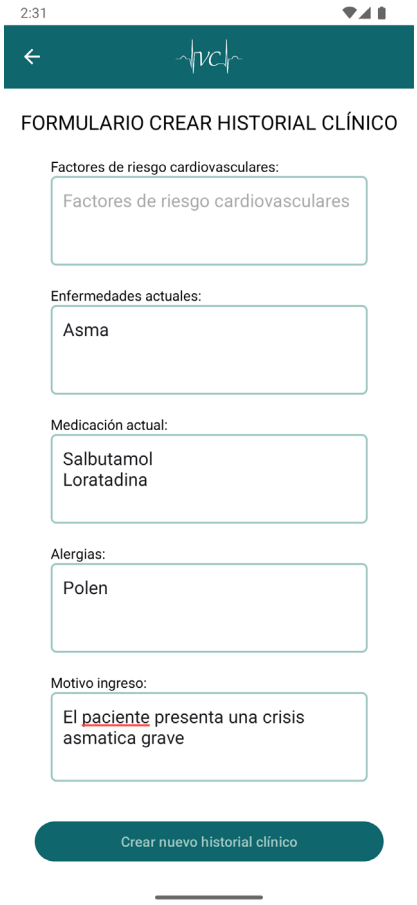


Figura 8-33. Formulario para crear historial clínico 2

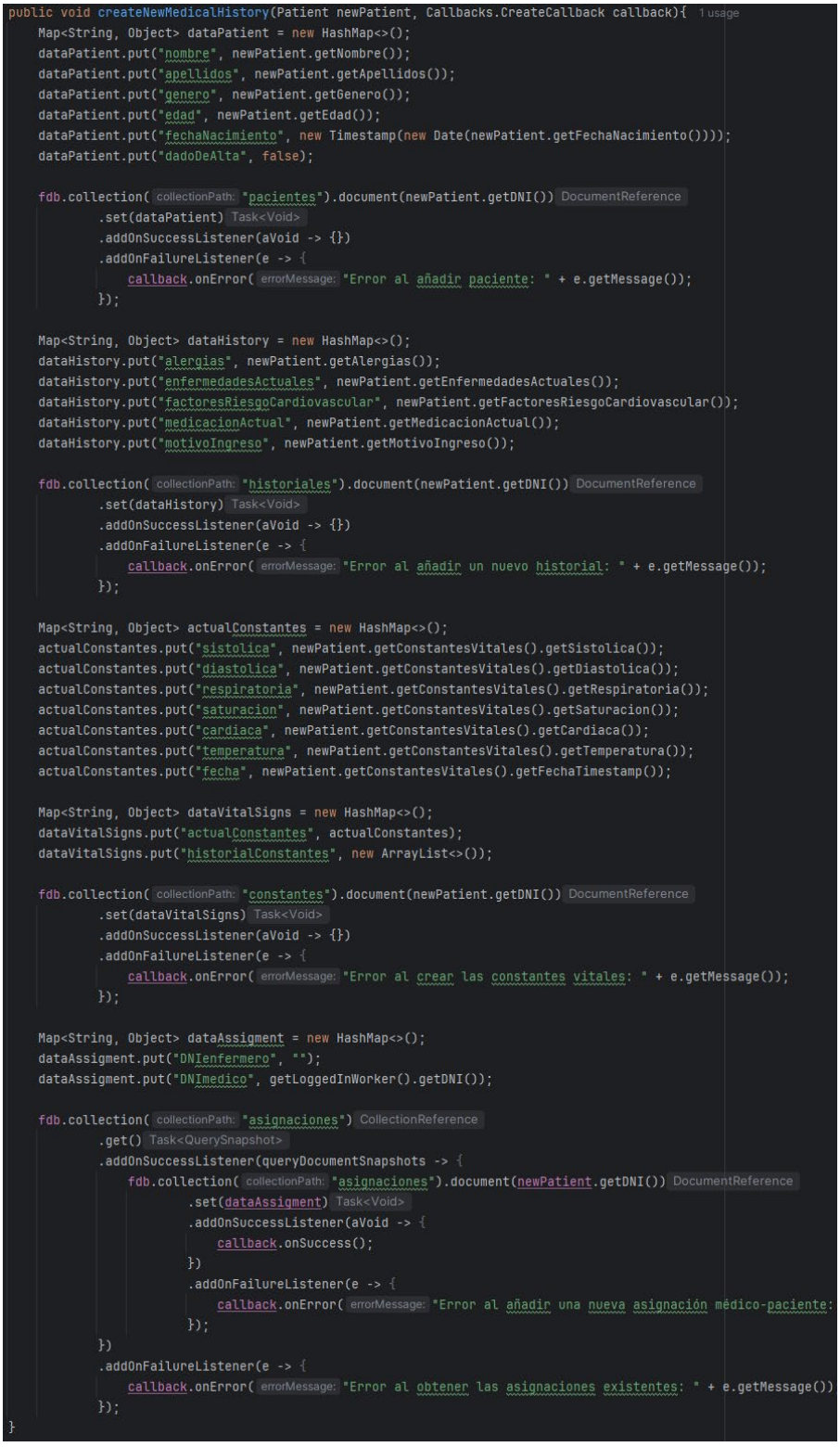
Dentro de los campos obligatorios, se incluyen todos los datos personales (nombre, apellidos, DNI, fecha de nacimiento…), así como el motivo de ingreso, que es el único campo médico requerido obligatoriamente en esta etapa.

Una vez completado el formulario, al pulsar el botón de "Crear nuevo historial clínico", el sistema se encarga de validar todos los datos introducidos. En el caso de la fecha de nacimiento, para comprobar que es válida utilizamos el método que mostramos en la figura 8-34, el cual se encarga de verificar que el formato sea correcto; que el día se corresponda con el mes seleccionado (teniendo en cuenta meses con 30 y 31 días, así como años bisiestos en el caso de febrero), y que el año esté dentro de un rango razonable y no sea posterior al actual.

```java
private Date FechaEsValida(String fechaStr) {  1 usage
    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd/MM/yyyy", Locale.getDefault());
    sdf.setLenient(false);

    try {
        Date fechaIngresada = sdf.parse(fechaStr);
        Calendar fechaNacimiento = Calendar.getInstance();
        fechaNacimiento.setTime(fechaIngresada);

        int dia = fechaNacimiento.get(Calendar.DAY_OF_MONTH);
        int mes = fechaNacimiento.get(Calendar.MONTH) + 1; // Los meses en Calendar van de 0 a 11
        int anio = fechaNacimiento.get(Calendar.YEAR);

        // Obtener fecha actual
        Calendar fechaActual = Calendar.getInstance();
        fechaActual.set(Calendar.HOUR_OF_DAY, 0);
        fechaActual.set(Calendar.MINUTE, 0);
        fechaActual.set(Calendar.SECOND, 0);
        fechaActual.set(Calendar.MILLISECOND, 0);

        // Comprobar que la fecha no es mayor a la actual
        if (fechaNacimiento.after(fechaActual)) {
            return null;
        }

        //Comprobar que el año no muy antiguo
        Calendar hace100Anios = (Calendar) fechaActual.clone();
        hace100Anios.add(Calendar.YEAR,  amount: -100);

        if (fechaNacimiento.before(hace100Anios)) {
            return null;
        }

        // Comprobación de días según el mes y si es año bisiesto
        int[] diasPorMes = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

        // Ajustar febrero para años bisiestos
        if((anio % 4 == 0 && anio % 100 != 0) || (anio % 400 == 0)) {
            diasPorMes[2] = 29;
        }

        // Comprobar que el mes y día son válidos
        if(mes < 1 || mes > 12 || dia < 1 || dia > diasPorMes[mes]) {
            return null;
        }

        return fechaIngresada;

    } catch (ParseException e) {
        return null;
    }
}
```

Figura 8-34. Implementación para validar la fecha

En cuanto a los campos médicos, como se ha mencionado anteriormente, el único obligatorio es el motivo de ingreso. Una vez superado el proceso de validación, el historial clínico del paciente es almacenado en la base de datos, quedando registrado correctamente en el sistema como se muestra en el método de la figura 8-35.



```
public void createNewMedicalHistory(Patient newPatient, Callbacks.CreateCallback callback){  1 usage
    Map<String, Object> dataPatient = new HashMap<>();
    dataPatient.put("nombre", newPatient.getNombre());
    dataPatient.put("apellidos", newPatient.getApellidos());
    dataPatient.put("genero", newPatient.getGenero());
    dataPatient.put("edad", newPatient.getEdad());
    dataPatient.put("fechaNacimiento", new Timestamp(new Date(newPatient.getFechaNacimiento())));
    dataPatient.put("dadoDeAlta", false);

    fdb.collection( collectionPath: "pacientes").document(newPatient.getDNI()) DocumentReference
            .set(dataPatient) Task<Void>
            .addOnSuccessListener(aVoid -> {})
            .addOnFailureListener(e -> {
                callback.onError( errorMessage: "Error al añadir paciente: " + e.getMessage());
            });

    Map<String, Object> dataHistory = new HashMap<>();
    dataHistory.put("alergias", newPatient.getAlergias());
    dataHistory.put("enfermedadesActuales", newPatient.getEnfermedadesActuales());
    dataHistory.put("factoresRiesgoCardiovascular", newPatient.getFactoresRiesgoCardiovascular());
    dataHistory.put("medicacionActual", newPatient.getMedicacionActual());
    dataHistory.put("motivoIngreso", newPatient.getMotivoIngreso());

    fdb.collection( collectionPath: "historiales").document(newPatient.getDNI()) DocumentReference
            .set(dataHistory) Task<Void>
            .addOnSuccessListener(aVoid -> {})
            .addOnFailureListener(e -> {
                callback.onError( errorMessage: "Error al añadir un nuevo historial: " + e.getMessage());
            });

    Map<String, Object> actualConstantes = new HashMap<>();
    actualConstantes.put("sistolica", newPatient.getConstantesVitales().getSistolica());
    actualConstantes.put("diastolica", newPatient.getConstantesVitales().getDiastolica());
    actualConstantes.put("respiratoria", newPatient.getConstantesVitales().getRespiratoria());
    actualConstantes.put("saturacion", newPatient.getConstantesVitales().getSaturacion());
    actualConstantes.put("cardiaca", newPatient.getConstantesVitales().getCardiaca());
    actualConstantes.put("temperatura", newPatient.getConstantesVitales().getTemperatura());
    actualConstantes.put("fecha", newPatient.getConstantesVitales().getFechaTimestamp());

    Map<String, Object> dataVitalSigns = new HashMap<>();
    dataVitalSigns.put("actualConstantes", actualConstantes);
    dataVitalSigns.put("historialConstantes", new ArrayList<>());

    fdb.collection( collectionPath: "constantes").document(newPatient.getDNI()) DocumentReference
            .set(dataVitalSigns) Task<Void>
            .addOnSuccessListener(aVoid -> {})
            .addOnFailureListener(e -> {
                callback.onError( errorMessage: "Error al crear las constantes vitales: " + e.getMessage());
            });

    Map<String, Object> dataAssigment = new HashMap<>();
    dataAssigment.put("DNIenfermero", "");
    dataAssigment.put("DNImedico", getLoggedInWorker().getDNI());

    fdb.collection( collectionPath: "asignaciones") CollectionReference
            .get() Task<QuerySnapshot>
            .addOnSuccessListener(queryDocumentSnapshots -> {
                fdb.collection( collectionPath: "asignaciones").document(newPatient.getDNI()) DocumentReference
                        .set(dataAssigment) Task<Void>
                        .addOnSuccessListener(aVoid -> {
                            callback.onSuccess();
                        })
                        .addOnFailureListener(e -> {
                            callback.onError( errorMessage: "Error al añadir una nueva asignación médico-paciente:
                        });
            })
            .addOnFailureListener(e -> {
                callback.onError( errorMessage: "Error al obtener las asignaciones existentes: " + e.getMessage())
            });
}
```

Figura 8-35. Implementación para crear el nuevo historial clínico

8.3.4 Consultar historial clínico del paciente

Para consultar los datos de un paciente, por ejemplo, Julia, a quien acabamos de añadir, únicamente es necesario pulsar sobre el cardView correspondiente dentro de la lista de pacientes. El sistema seguidamente nos redirige a una nueva interfaz donde se despliega toda la información detallada del paciente. Esta vista está estructurada en diferentes secciones, lo que facilita la navegación y lectura de los datos. En primer lugar, se muestran los datos personales del paciente, y si el médico pulsa en la pestaña correspondiente al historial clínico, se accede a la información médica del paciente. Podemos observar en las figuras 8-36 y 8-37, que esta organización facilita al médico un acceso rápido y ordenado.

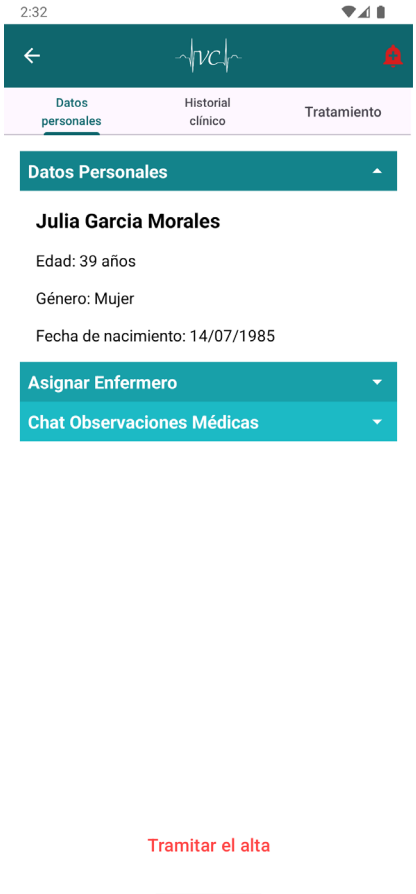


Figura 8-36. Sección para ver los datos personales

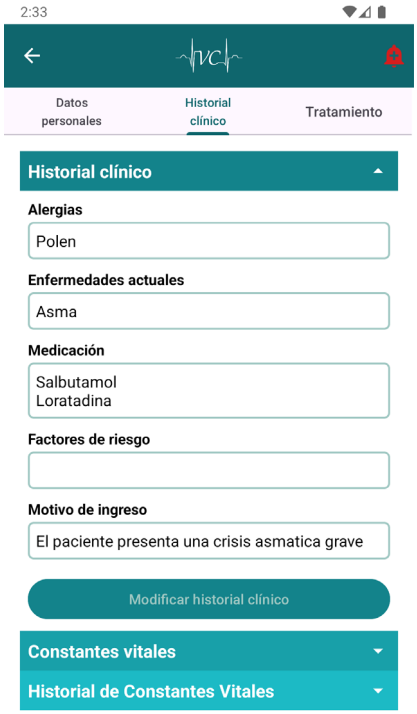


Figura 8-37. Sección para ver el historial clínico

8.3.5 Modificar historial clínico del paciente

En la sección dedicada al historial clínico, el médico dispone de la funcionalidad para modificar cualquier aspecto relacionado con el historial clínico. Al pulsar sobre "Modificar historial clínico", el sistema redirige automáticamente al formulario mostrado en la figura 8-38, donde todos los datos médicos previamente registrados del paciente ya se encuentran cargados.

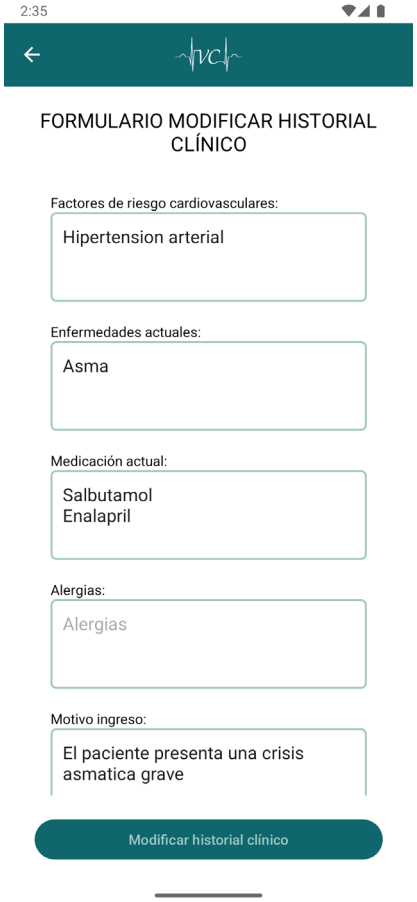


Figura 8-38. Formulario para modificar el historial clínico

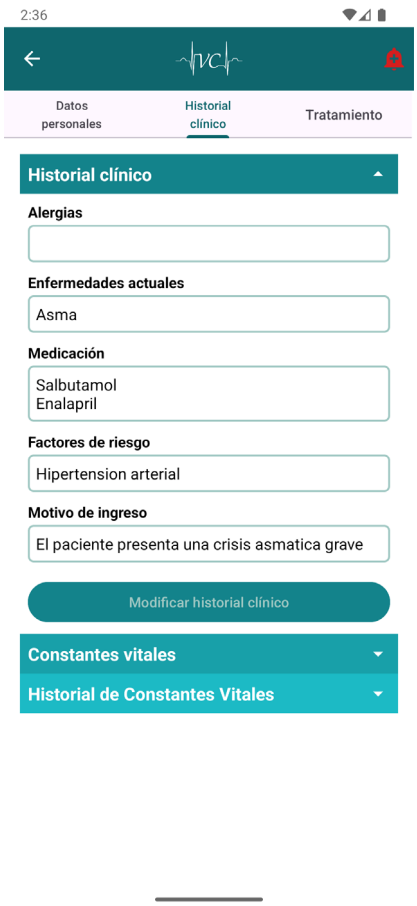


Figura 8-39. Historial clínico modificado

Una vez realizado todas las modificaciones necesarias, el médico debe pulsar el botón inferior de la pantalla para confirmar los cambios. En ese momento, el sistema lanza un cuadro de diálogo de doble confirmación para verificar que el médico desea realmente guardar la nueva versión del historial.

Esta información se guarda dentro de un objeto de tipo Patient, el cual se pasa como parámetro al método mostrado en la figura 8-40, que se encarga de actualizar el historial clínico en la base de datos.

```
public void updateMedicalHistory(Patient patient, Callbacks.UpdateCallback callback) {  1 usage
    Map<String, Object> dataHistory = new HashMap<>();
    dataHistory.put("alergias", patient.getAlergias());
    dataHistory.put("enfermedadesActuales", patient.getEnfermedadesActuales());
    dataHistory.put("factoresRiesgoCardiovascular", patient.getFactoresRiesgoCardiovascular());
    dataHistory.put("medicacionActual", patient.getMedicacionActual());
    dataHistory.put("motivoIngreso", patient.getMotivoIngreso());

    fdb.collection( collectionPath: "historiales").document(patient.getDNI()) DocumentReference
            .update(dataHistory) Task<Void>
            .addOnSuccessListener(aVoid -> {
                callback.onSuccess();
            })
            .addOnFailureListener(e -> {
                callback.onError( errorMessage: "Error al editar el historial clínico: " + e.getMessage());
            });
}
```

Figura 8-40. Implementación para actualizar historial clínico

8.3.6 Añadir/eliminar asignación de paciente a enfermero

Para asignar un enfermero a un paciente, el médico, dentro de la pestaña de datos personales, debe desplegar la sección "Asignar enfermero", figura 8-41. Después, debe pulsa en "Asignar" y se muestra por pantalla un fragmento con todos los enfermeros registrados en el sistema como se observar en la figura 8.42. El médico puede visualizar esta lista y seleccionar al profesional correspondiente. Una vez confirmada la selección, el sistema actualiza automáticamente la base de datos y registra dicha asignación dentro de la colección específica de asignaciones.

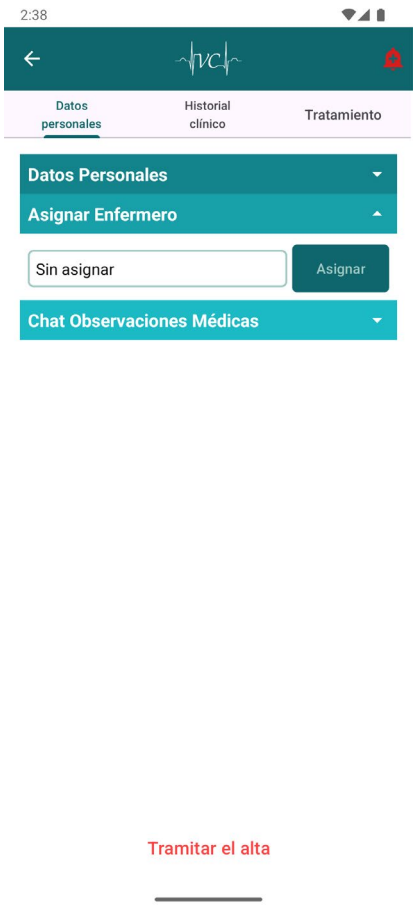


Figura 8-41. Sección para asignar enfermero

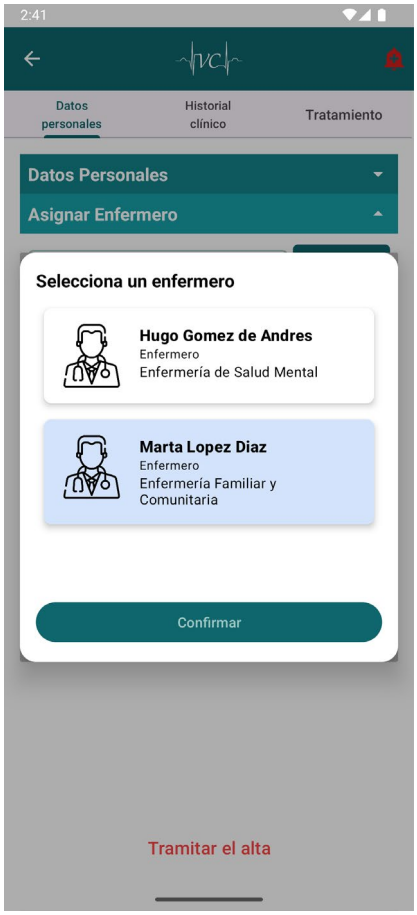


Figura 8-42. Fragmento para elegir enfermero

En cuanto a la lógica de implementación, mostrada en la figura 8-43, el procedimiento es directo: el sistema modifica la colección de asignaciones añadiendo el NIF del enfermero al documento referente al paciente.

```
public void updateAssignNurse(String DNIpaciente, String DNImedico, String DNIenfermero, Callbacks.UpdateCallback callback) {
    Map<String, Object> dataWorker = new HashMap<>();
    dataWorker.put("DNImedico", DNImedico);
    dataWorker.put("DNIenfermero", DNIenfermero);

    fdb.collection( collectionPath: "asignaciones").document(DNIpaciente) DocumentReference
            .update(dataWorker) Task<Void>
            .addOnSuccessListener(aVoid -> {
                callback.onSuccess();
            })
            .addOnFailureListener(e -> {
                callback.onError( errorMessage: "Error al editar enfermero: " + e.getMessage());
            });
}
```

Figura 8-43. Implementación para asignar enfermero

Para eliminar al enfermero asignado, simplemente el médico debe pulsar en el botón de "Eliminar" tal y como se observa en la figura 8-44. A continuación, se muestra un mensaje informativo indicando que la acción ha finalizado con éxito.

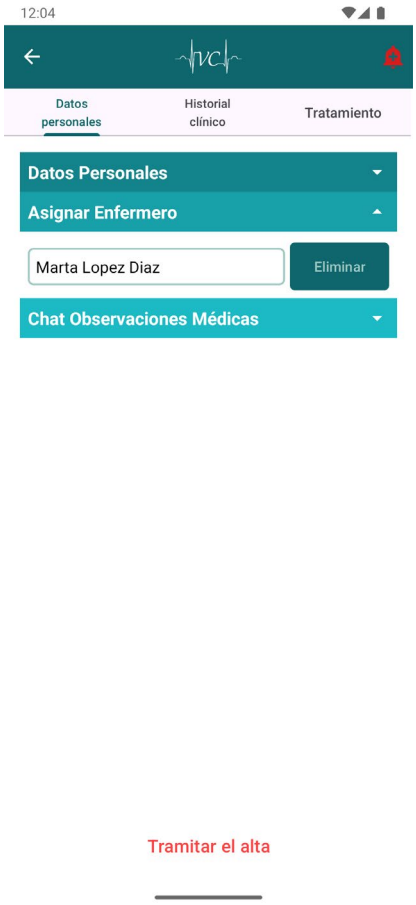


Figura 8-44. Enfermero asignado correctamente

8.3.7 Añadir/consultar observaciones médicas del paciente

Los profesionales sanitarios tienen una sección en la que pueden anotar las observaciones médicas que consideren necesarias para el correcto seguimiento y tratamiento del paciente. De esta forma, pueden compartirlas con el otro sanitario que se encargue del cuidado del paciente durante su ingreso y así ambos estén actualizados respecto al estado del paciente. Para ello, previamente el médico debe haber asignado el paciente a un enfermero para que así las observaciones médicas sean visibles para ambos.

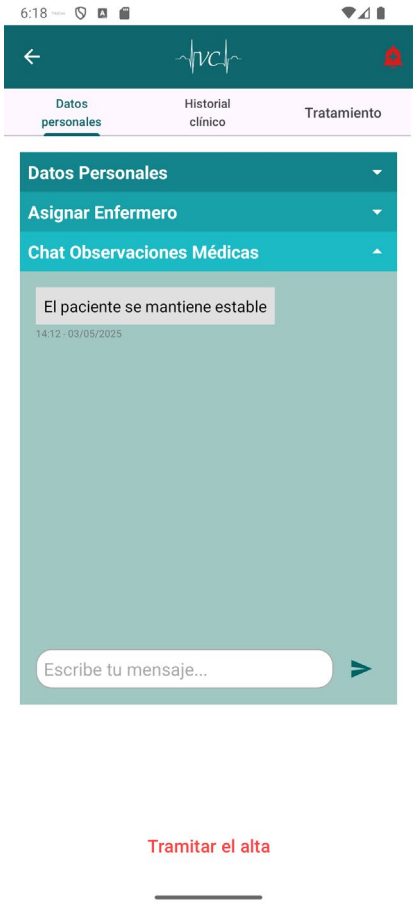


Figura 8-45.Sección para consultar chat de observaciones médicas

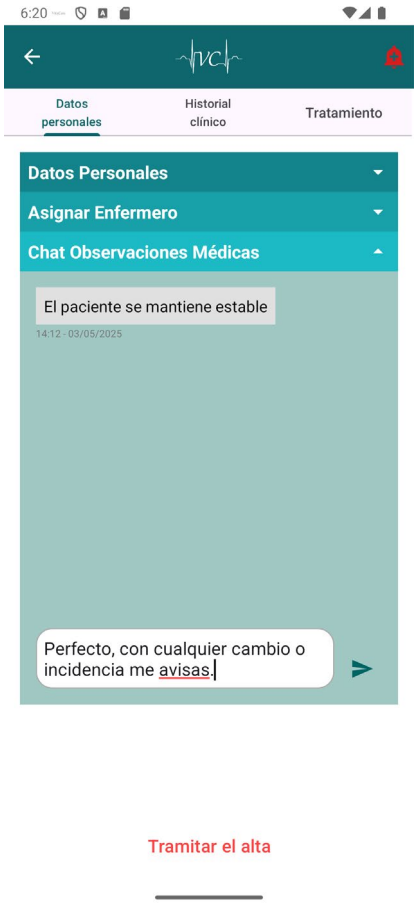


Figura 8-46. Añadir observación médica

Para que el profesional sanitario pueda consultar las observaciones médicas guardadas sobre el paciente debe acceder a la pestaña "Datos personales" del paciente en donde podrá encontrar el chat con las observaciones médicas que tienen entre el médico y el enfermero.

Una vez haya accedido a este apartado, el profesional sanitario puede visualizar todas las observaciones. Como se puede ver en la figura 8-45, aquellos mensajes que aparecen de color grisáceo y en el margen izquierdo del chat, hacen referencia a las observaciones médicas anotadas por el otro sanitario encargado del tratamiento del paciente. En el chat, además de mostrarse la observación médica en el mensaje, se detalla la hora y la fecha en la que fue anotada, de esta manera facilitamos a los sanitarios poder hacer un seguimiento más detallado.

Cuando el profesional sanitario desee añadir una observación, tal y como se muestra en la figura 8-46, es tan sencillo e intuitivo para ellos como pulsar sobre el campo de escritura en el chat y escribir el mensaje. Una vez redactada la observación médica, cuando pulsa sobre el botón de enviar el sistema actualiza la pantalla del chat y se envía el mensaje para que lo reciba el enfermero.

La lógica que se muestra en la figura 8-47, añade la observación médica como un mapa más dentro del documento único creado con el NIF del paciente en la colección observaciones en el que se guardan todas sus observaciones médicas.

```
public void saveNotification(String DNIworker, String titulo, String mensaje, String tabIndex, String dniP) {  6 usages
    DocumentReference docRef = fdb.collection( collectionPath: "notificaciones").document(DNIworker);

    docRef.get().addOnSuccessListener(documentSnapshot -> {
        Map<String, Object> existingNotifications = documentSnapshot.exists()
                ? documentSnapshot.getData()
                : new HashMap<>();

        // Encontrar el mayor índice numérico
        int nextIndex = 0;
        for (String key : existingNotifications.keySet()) {
            try {
                int index = Integer.parseInt(key);
                if (index >= nextIndex) {
                    nextIndex = index + 1;
                }
            } catch (NumberFormatException ignored) {
                Log.e( tag: "Firestore",  msg: "Clave no válida en el mapa de notificaciones");
            }
        }

        // Crear nueva notificación
        Map<String, Object> newNotification = new HashMap<>();
        newNotification.put( k: "fecha", Timestamp.now());
        newNotification.put( k: "mensaje", mensaje);
        newNotification.put( k: "titulo", titulo);
        newNotification.put( k: "tabIndex", tabIndex);
        newNotification.put( k: "dniP", dniP);

        // Guardar en la base de datos
        docRef.update(String.valueOf(nextIndex), newNotification)
                .addOnSuccessListener(aVoid -> Log.d( tag: "Firestore",  msg: "Notificación guardada correctamente"))
                .addOnFailureListener(e -> {
                    Log.e( tag: "Firestore",  msg: "Error al guardar notificación", e);

                    // Si falla, probablemente el documento no existe, asi que lo creamos
                    Map<String, Object> firstNotification = new HashMap<>();
                    firstNotification.put( k: "0", newNotification);
                    docRef.set(firstNotification)
                            .addOnSuccessListener(aVoid2 -> Log.d( tag: "Firestore",  msg: "Documento creado con la primera notificación"))
                            .addOnFailureListener(e2 -> Log.e( tag: "Firestore",  msg: "Error al crear documento", e2));
                });

    }).addOnFailureListener(e -> Log.e( tag: "Firestore",  msg: "Error al obtener documento", e));
}
```

Figura 8-47. Implementación para enviar observación médica

Como queda reflejado en la figura 8-48, una vez el sanitario envía el mensaje, este se muestra en el chat en el margen derecho de color azulado. Y al igual que con las observaciones médicas recibidas, la enviada tiene referenciada la hora y la fecha en la que el usuario envió el mensaje.

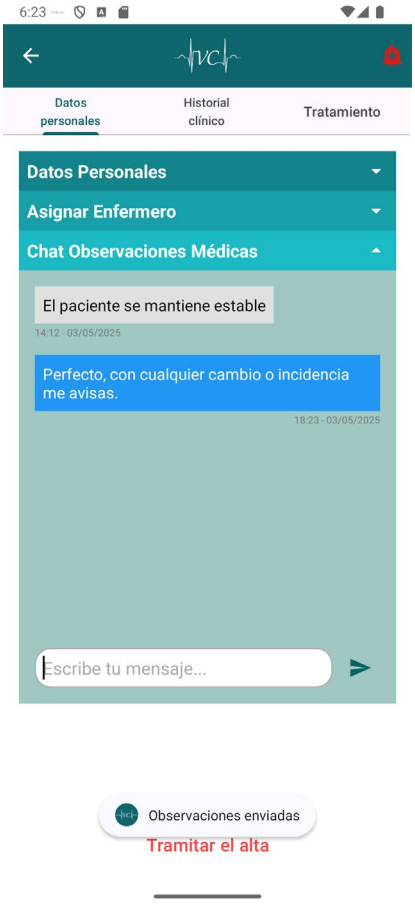


Figura 8-48. Observación médica enviada

8.3.8 Añadir/modificar constantes vitales del paciente

Los sanitarios pueden tomar las constantes vitales al paciente y registrarlas al momento en la sección "Constantes vitales" de la pestaña "Historial clínico". De esta forma, pueden añadir por primera vez las constantes vitales del paciente si acaba de ingresar como se puede ver en la figura 8-49, o modificar las últimas tomadas como se

observa en la figura 8-50 ya que le aparecen los valores del último registro de las constantes que le fueron tomadas al paciente.

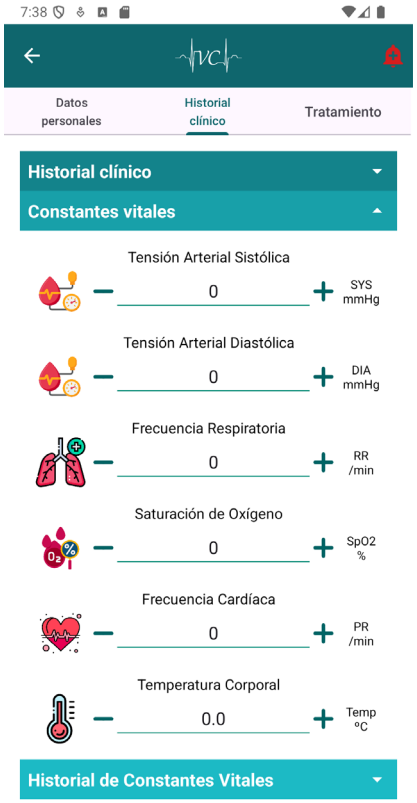


Figura 8-49. Sección para añadir desde cero constantes vitales

Figura 8-50. Sección para modificar constantes vitales ya existentes

Para poder introducir los valores de las constantes vitales el profesional sanitario puede hacerlo de forma manual introduciendo el número por la entrada del teclado o hacerlo con la ayuda del "-/+". Estos van cambiando en unidades el valor de las constantes excepto en el caso de la temperatura corporal que modifican en decimas el valor. Esta lógica queda reflejada en la figura 8-51.

```java
// Método para incrementar el valor
private void incrementValue(EditText editText, double changeValue, double minValue,  double maxValue) {
    double value = getNumericValue(editText, minValue);
    value += changeValue;

    if (value > maxValue) {
        value = maxValue;
    }
    else if (value < minValue){
        value = minValue;
    }
    editText.setText(formatValue(value, changeValue));
}

// Método para decrementar el valor
private void decrementValue(EditText editText, double changeValue, double minValue, double maxValue) {
    double value = getNumericValue(editText, maxValue);
    value -= changeValue;

    if (value < minValue) {
        value = maxValue;
    }
    else if (value > maxValue) {
        value = maxValue;
    }
    editText.setText(formatValue(value, changeValue));
}
```

Figura 8-51. Implementación para aumentar/decrementar valor constante vital

Para cada constante vital se definieron unos máximos y mínimos, mostrados en la figura 8-52, que han sido consultados y preestablecidos. Estos márgenes en los que se puede encontrar el valor cada constante vital han sido especificados teniendo en cuenta la posibilidad de que el paciente pueda tener esos valores y esté vivo. De este modo, podemos asegurar que los valores introducidos por los sanitarios tengan sentido y evitar que estos sean incoherentes.

```
private static final int MAX_SISTOLICA = 300;  7 usages
private static final int MIN_SISTOLICA = 40;  7 usages
private static final int MAX_DIASTOLICA = 200;  7 usages
private static final int MIN_DIASTOLICA = 10;  7 usages
private static final int MAX_RESPIRATORIA = 60;  7 usages
private static final int MIN_RESPIRATORIA = 4;  7 usages
private static final int MAX_SATURACION = 100;  7 usages
private static final int MIN_SATURACION = 30;  7 usages
private static final int MAX_CARDIACA = 350;  7 usages
private static final int MIN_CARDIACA = 15;  7 usages
private static final double MAX_TEMPERATURA = 44.0;  7 usages
private static final double MIN_TEMPERATURA = 30.0;  7 usages
```

Figura 8-52. Límites min/máx de constantes vitales

Para que estos límites se cumplan, se ha implementado una lógica, figura 8-53, que cuando un sanitario modifica cualquier valor de alguna de las constantes vitales, este es revisado al momento para asegurar la consistencia de este. En caso de que no esté dentro del intervalo definido, el valor es autocorregido de manera automática. Así, si el valor introducido sobrepasa el máximo, el valor que se queda registrado es el máximo absoluto definido, y en caso de que sea menor que el mínimo, la constante toma el valor definido como mínimo para esa constante vital.

```
private void validateValue(EditText editText, double minValue, double maxValue, double changeValue) {
    String valueStr = editText.getText().toString();
    if (valueStr.isEmpty()) { // No hacemos nada si el campo está vacío
        return;
    }

    double value = getNumericValue(editText, minValue);
    if (value < minValue) {
        editText.setText(formatValue(minValue, changeValue));
    } else if (value > maxValue) {
        editText.setText(formatValue(maxValue, changeValue));
    }
}
```

Figura 8-53. Implementación para validar valores constantes vitales

A su vez, si el valor de la constante vital es igual que cualquier de los límites inferiores o superiores, los botones de acción "-/+" son deshabilitados en función del valor. Si el valor es igual al mínimo, se deshabilita el "-" y si es igual al máximo el que queda deshabilitado es el "+". Lo hemos implementado con el método que se muestra en la figura 8-54.

```
private void updateButtonStates(EditText editText, ImageButton btnDecrease, ImageButton btnIncrease, double minValue, double maxValue) {
    String valueStr = editText.getText().toString();
    double value;

    if (!valueStr.isEmpty()) {
        value = getNumericValue(editText, minValue);
        boolean canDecrease = value > minValue;
        boolean canIncrease = value < maxValue;

        // Aplicamos colorFilter (deshabilitarlo y que sea visible)
        btnDecrease.setEnabled(canDecrease);
        if (!canDecrease) {
            btnDecrease.setColorFilter(Color.GRAY, PorterDuff.Mode.SRC_IN);
        } else {
            btnDecrease.setColorFilter(null);
        }

        btnIncrease.setEnabled(canIncrease);
        if (!canIncrease) {
            btnIncrease.setColorFilter(Color.GRAY, PorterDuff.Mode.SRC_IN);
        } else {
            btnIncrease.setColorFilter(null);
        }
    } else {
        // Si el campo está vacío, deshabilitamos el botón de disminuir
        btnDecrease.setEnabled(false);
        btnDecrease.setColorFilter(Color.GRAY, PorterDuff.Mode.SRC_IN);
        // No lo deshabilitamos para que pueda aumentar
        btnIncrease.setEnabled(true);
        btnIncrease.setColorFilter(null);
    }
}
```

Figura 8-54. Implementación para actualizar estado de botones "-/+"

Una vez el sanitario ha introducido los nuevos valores de las constantes vitales que desea registrar, de forma automática en cuanto el sistema detecta un cambio en algunos de los valores de cualquiera constante vital respecto al original que había anteriormente, le aparecerá en pantalla el botón para que pueda guardar los cambios como se puede observar en la figura 8-56.

```
private void checkForChanges() {  1 usage
    boolean hasChanges = false;

    // Obtenemos los valores actuales de los EditText
    String sistolicaStr = editTextSistolica.getText().toString();
    String diastolicaStr = editTextDiastolica.getText().toString();
    String respiratoriaStr = editTextRespiratoria.getText().toString();
    String saturacionStr = editTextSaturacion.getText().toString();
    String cardiacaStr = editTextCardiaca.getText().toString();
    String temperaturaStr = editTextTemperatura.getText().toString();

    // Convertimos los valores a números
    int sistolica = sistolicaStr.isEmpty() ? 0 : Integer.parseInt(sistolicaStr);
    int diastolica = diastolicaStr.isEmpty() ? 0 : Integer.parseInt(diastolicaStr);
    int respiratoria = respiratoriaStr.isEmpty() ? 0 : Integer.parseInt(respiratoriaStr);
    int saturacion = saturacionStr.isEmpty() ? 0 : Integer.parseInt(saturacionStr);
    int cardiaca = cardiacaStr.isEmpty() ? 0 : Integer.parseInt(cardiacaStr);
    double temperatura = temperaturaStr.isEmpty() ? 0 : Double.parseDouble(temperaturaStr);


    // Comparamos con los valores originales
    if (sistolica != patient.getConstantesVitales().getSistolica()) {
        hasChanges = true;
    } else if (diastolica != patient.getConstantesVitales().getDiastolica()) {
        hasChanges = true;
    } else if (respiratoria != patient.getConstantesVitales().getRespiratoria()) {
        hasChanges = true;
    } else if (saturacion != patient.getConstantesVitales().getSaturacion()) {
        hasChanges = true;
    } else if (cardiaca != patient.getConstantesVitales().getCardiaca()) {
        hasChanges = true;
    } else if (temperatura != patient.getConstantesVitales().getTemperatura()) {
        hasChanges = true;
    }

    // Mostramos u ocultamos botón si hay cambios
    if (hasChanges) {
        buttonGuardarConstantes.setVisibility(View.VISIBLE);
    } else {
        buttonGuardarConstantes.setVisibility(View.GONE);
    }
}
```

Figura 8-55. Implementación para comprobar cambio de valores en constantes vitales

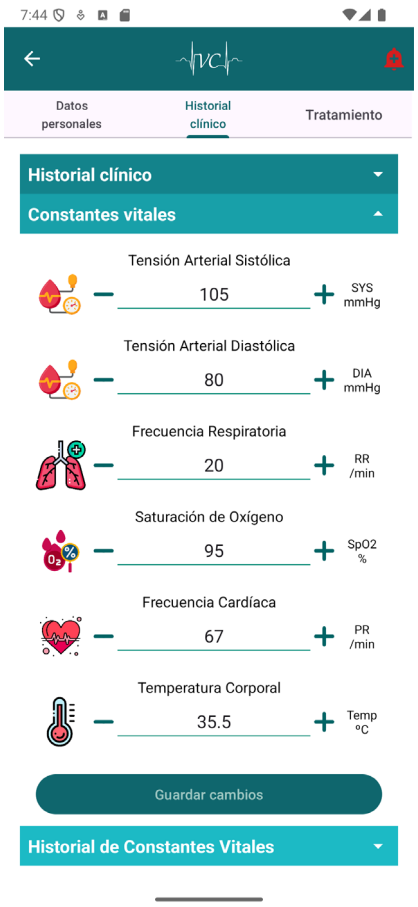


Figura 8-56. Guardar cambios en constantes vitales

Cuando el sanitario procede a guardar los cambios y confirmarlo, el sistema vuelve a validar los valores comprobando que estén dentro de los intervalos de máximo y mínimo y que ninguno de ellos está vacío, figura 8-55. Las constantes vitales son actualizadas y se añade un registro de estas en el historial de constantes vitales con la hora y la fecha de cuando han sido tomadas como se observa en la figura 9-57. Así pueden tener un registro completo de todos los valores cada vez que le tomaron las constantes vitales.

```
public void updateVitalSigns(Patient patient, Callbacks.UpdateCallback callback) {  1 usage
    Map<String, Object> dataVitalSigns = new HashMap<>();

    Map<String, Object> actualConstantes = patient.getConstantesVitales().toMap();
    dataVitalSigns.put( k: "actualConstantes", actualConstantes);

    List<VitalSigns> historialConstantes = patient.getHistorialConstantesVitales();
    List<Map<String, Object>> historialConstantesConverted = new ArrayList<>();
    if (historialConstantes != null) {
        for (VitalSigns vs : historialConstantes) {
            historialConstantesConverted.add(vs.toMap());
        }
    }
    dataVitalSigns.put( k: "historialConstantes", historialConstantesConverted);

    fdb.collection( collectionPath: "constantes").document(patient.getDNI()) DocumentReference
            .update(dataVitalSigns) Task<Void>
            .addOnSuccessListener(aVoid -> {
                callback.onSuccess();
            })
            .addOnFailureListener(e -> {
                callback.onError( errorMessage: "Error al editar las constantes vitales: " + e.getMessage());
            });
}
```

Figura 8-57. Implementación para modificar constantes vitales

8.3.9 Consultar constantes vitales del paciente

Para consultar las constantes vitales que han sido tomadas al paciente, deben ir a la misma sección desde la que pueden modificar los valores. Esta sección sirve tanto para visualizar como para modificar las constantes vitales, figura 8-58.

Además, la sección situada por debajo de "Constantes vitales" se ubica el historial de constantes vitales. Como se observa en la figura 8-59, en esta sección los sanitarios pueden visualizar una tabla en la que se quedan registrados todos los cambios que se han introducido en las constantes vitales, acompañados de una marca temporal en la que se especifica la hora y la fecha en la que se tomaron las constantes y fueron guardadas. Así, pueden hacer un seguimiento y tener una visión general de la evolución del paciente guiándose por sus constantes vitales.

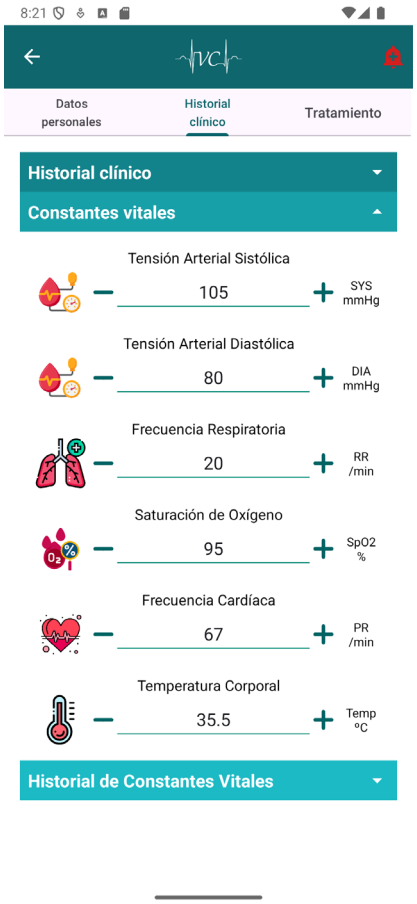


Figura 8-58. Sección para consultar últimas constantes vitales tomadas

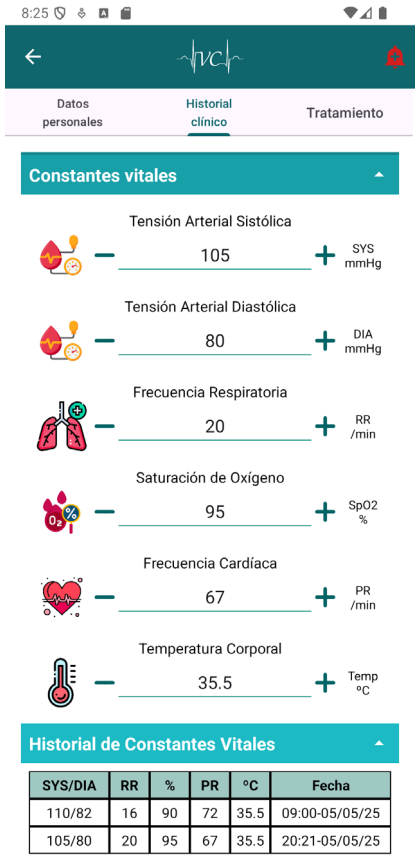


Figura 8-59. Sección para consultar historial de constantes vitales

8.3.10 Crear tratamiento del paciente

Para que el médico pueda crear un tratamiento para el diagnóstico del paciente debe acceder a la sección tratamiento del paciente y pulsar sobre el botón "Crear nuevo tratamiento", que le lleva al formulario correspondiente para establecer un tratamiento a seguir para el paciente como se muestra en la figura 8-60.

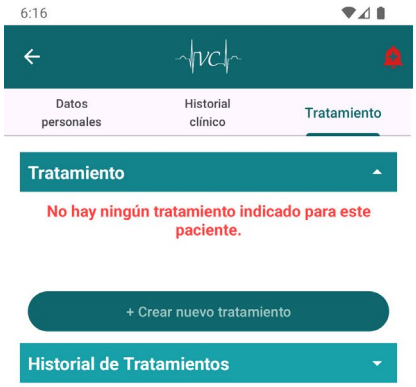


Figura 8-60. Pantalla sección tratamiento

Al pulsar sobre el botón aparece en pantalla un formulario, figura 8-61, para crear el tratamiento que desea pautar al paciente. En el formulario hay que rellenar los campos obligatorios que son los sueros, fármacos que deben ser administrados, la dieta a seguir y la actividad que recomienda que haga el paciente. Además, el médico debe dar una prioridad al tratamiento que va a crear. Esta después será utilizada en la funcionalidad que se describirá posteriormente que es ordenar los pacientes por prioridad de tratamiento. Este campo puede tomar uno de los siguientes valores haciendo referencia a la prioridad del tratamiento de mayor a menor correspondientemente: "Alta", "Media" o "Baja".

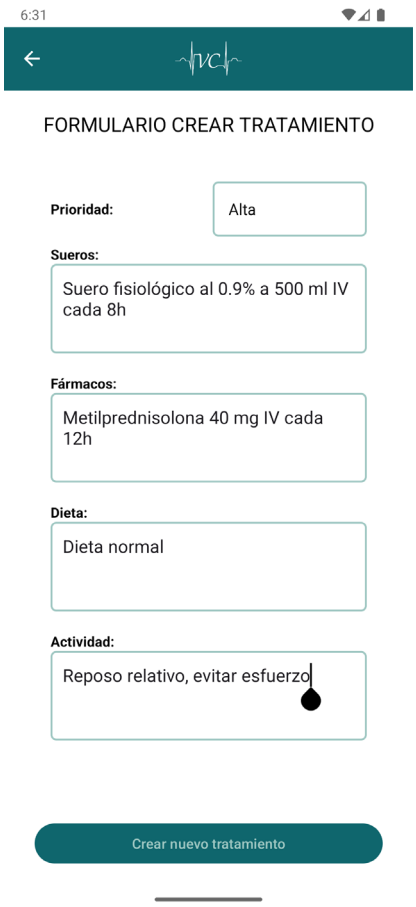


Figura 8-61. Formulario para crear tratamiento

Una vez rellenado el formulario, al pulsar el botón "Crear nuevo tratamiento", el sistema se encarga de validar todos los datos introducidos y que ninguno esté vacío. Una vez superado el proceso de validación, el tratamiento es almacenado en la base de datos, quedando registrado correctamente en el sistema como tratamiento actual y también en el historial de tratamiento del paciente como se muestra en el método de la figura 8-62.

```
public void createNewTreatment(Patient patient, Callbacks.CreateCallback callback) {  1 usage
    Map<String, Object> dataTreatment = new HashMap<>();

    Map<String, Object> actualTratamiento = patient.getTratamiento().toMap();
    dataTreatment.put( k: "actualTratamiento", actualTratamiento);

    List<Treatment> historialTratamiento = patient.getHistorialTratamientos();
    List<Map<String, Object>> historialTratamientoConverted = new ArrayList<>();
    if(historialTratamiento != null) {
        for (Treatment tr : historialTratamiento) {
            historialTratamientoConverted.add(tr.toMap());
        }
    }
    dataTreatment.put( k: "historialTratamiento", historialTratamientoConverted);

    fdb.collection( collectionPath: "tratamientos").document(patient.getDNI())  DocumentReference
            .set(dataTreatment)  Task<Void>
            .addOnSuccessListener(aVoid -> {
                callback.onSuccess();
            })
            .addOnFailureListener(e -> {
                callback.onError( errorMessage: "Error al editar las constantes vitales: " + e.getMessage());
            });
}
```

Figura 8-62. Implementación para crear nuevo tratamiento

8.3.11 Consultar/eliminar tratamiento del paciente

Para consultar el tratamiento de un paciente, el personal sanitario se debe desplazar hasta la pestaña de tratamiento y en la sección "Tratamiento" se muestra el tratamiento que se ha pautado como se observa en la figura 8-63. Además, también hay una sección en la que los sanitarios pueden ver el historial completo de todos los tratamientos que han sido administrados al paciente en la sección "Historial de tratamientos".

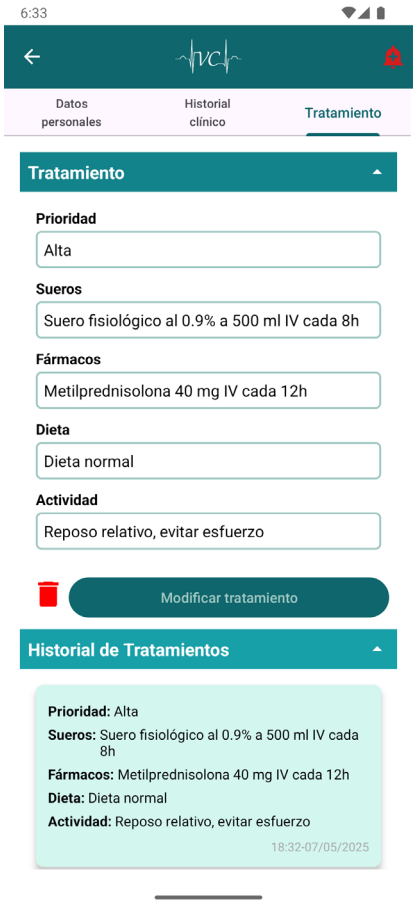


Figura 8-63. Sección para ver el tratamiento

En caso de que el médico desee eliminar el tratamiento que tiene, debe pulsar sobre el botón de la papelera que se encuentra en la esquina inferior izquierda del tratamiento. El sistema le pedirá una doble confirmación antes de realizar el borrado del tratamiento almacenado en la base de datos como se muestra en la figura 8-64.

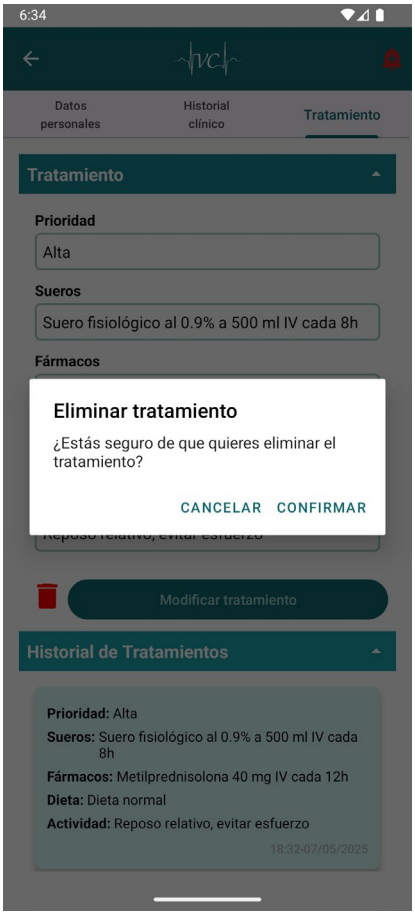


Figura 8-64. Confirmación para eliminar tratamiento

En cuanto a la lógica de la implementación, figura 8-65, una vez el médico ha confirmado la eliminación del tratamiento, el sistema se encarga de hacer un borrado de este de la base de datos eliminando el campo "actualTratamiento" del documento del paciente en la colección tratamientos. Aunque se elimina ese campo, el tratamiento no es borrado del array que contiene el historial completo de los tratamientos para que se siga mostrando.

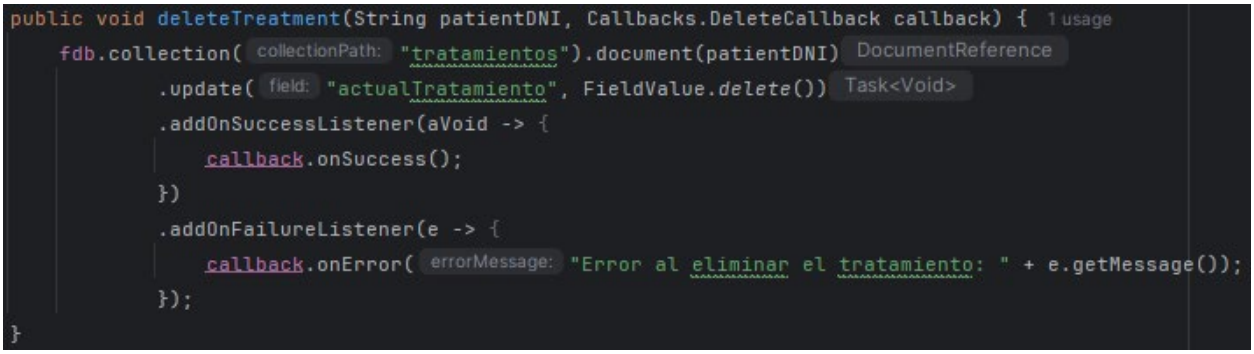


Figura 8-65. Implementación para eliminar tratamiento

8.3.12 Modificar tratamiento del paciente

En la sección dedicada al tratamiento, el médico dispone de una funcionalidad para poder modificar el tratamiento. Al pulsar sobre "Modificar tratamiento", el sistema redirige automáticamente al formulario mostrado en la figura 8-66, donde todas las pautas previamente registradas del tratamiento ya se encuentran cargadas.

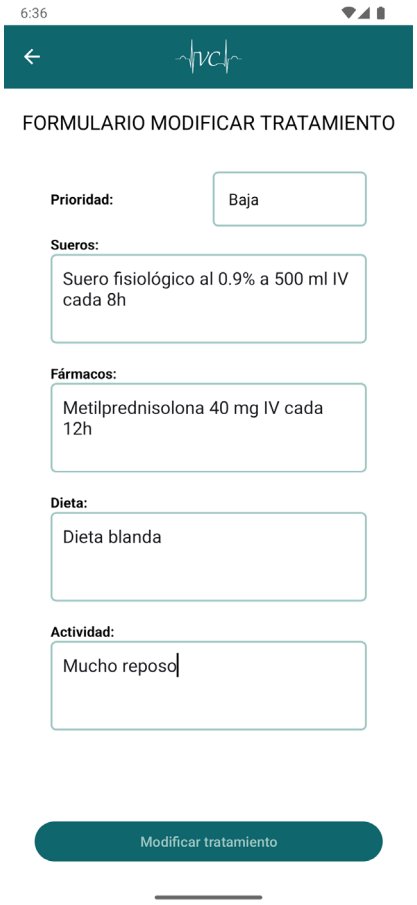


Figura 8-66. Formulario para modificar el tratamiento

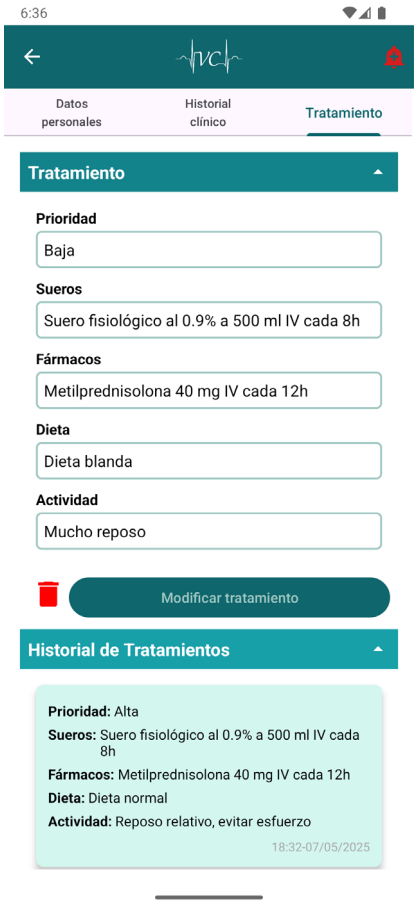


Figura 8-67. Tratamiento modificado

Cuando el médico realice todas las modificaciones que considere oportunas, deber pulsar el botón inferior de la pantalla para confirmar los cambios. En ese instante, el sistema lanza un cuadro de diálogo de doble confirmación para verificar que el médico desea realmente guardar los cambios hechos sobre el tratamiento.

Esta información se guarda dentro de un objeto de tipo Patient, el cual se pasa como parámetro a la función encargada de actualizar el tratamiento en la base de datos, figura 8-68.

```
public void updateTreatment(Patient patient, Callbacks.UpdateCallback callback) {  1 usage
    Map<String, Object> dataTreatment = new HashMap<>();

    Map<String, Object> actualTratamiento = patient.getTratamiento().toMap();
    dataTreatment.put( k: "actualTratamiento", actualTratamiento);

    List<Treatment> historialTratamiento = patient.getHistorialTratamientos();
    List<Map<String, Object>> historialTratamientoConverted = new ArrayList<>();
    if(historialTratamiento != null) {
        for (Treatment tr : historialTratamiento) {
            historialTratamientoConverted.add(tr.toMap());
        }
    }
    dataTreatment.put( k: "historialTratamiento", historialTratamientoConverted);

    fdb.collection( collectionPath: "tratamientos").document(patient.getDNI())  DocumentReference
            .update(dataTreatment)  Task<Void>
            .addOnSuccessListener(aVoid -> {
                callback.onSuccess();
            })
            .addOnFailureListener(e -> {
                callback.onError( errorMessage: "Error al editar las constantes vitales: " + e.getMessage());
            });
}
```

Figura 8-68. Implementación para actualizar tratamiento

Además, una vez se haya actualizado la base de datos el sistema redirige de nuevo a la pestaña del tratamiento. En la sección "Tratamiento" se verá el nuevo tratamiento pautado como se muestra en la figura 8-67, y en la sección "Historial de tratamientos" podremos ver este nuevo tratamiento añadido en un nuevo cardView como se puede observar en la figura 8-69.

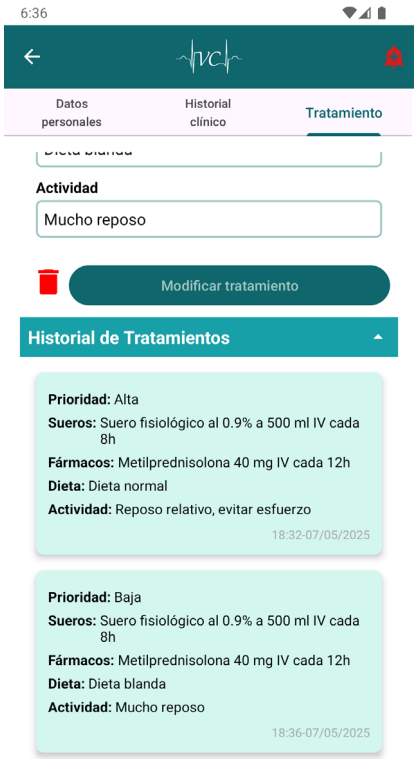


Figura 8-69. Historial de tratamientos actualizado

8.3.13 Ordenar pacientes por prioridad de tratamiento

Al igual que el listar los pacientes asignados, la ordenación de pacientes por prioridad de tratamiento es una funcionalidad que va de forma implícita en la aplicación. Los pacientes se ordenan de mayor a menor prioridad del tratamiento que tengan pautado. Como se puede observar, la diferencia que se observa entre las figuras 8-70 y 8-71, es el orden de los pacientes. En la figura 8-71 están ordenados por la prioridad del tratamiento de cada paciente, pero en la figura 8-70 al no haber ningún paciente con un tratamiento pautado, no están ordenados.

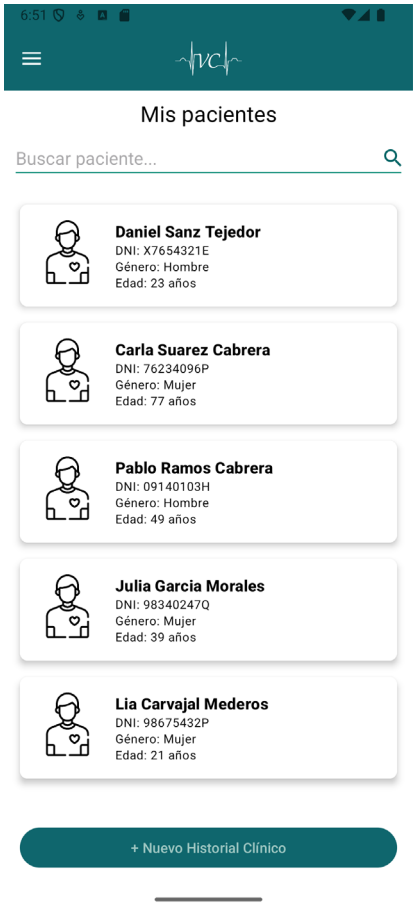


Figura 8-70. Listado de paciente sin ordenar

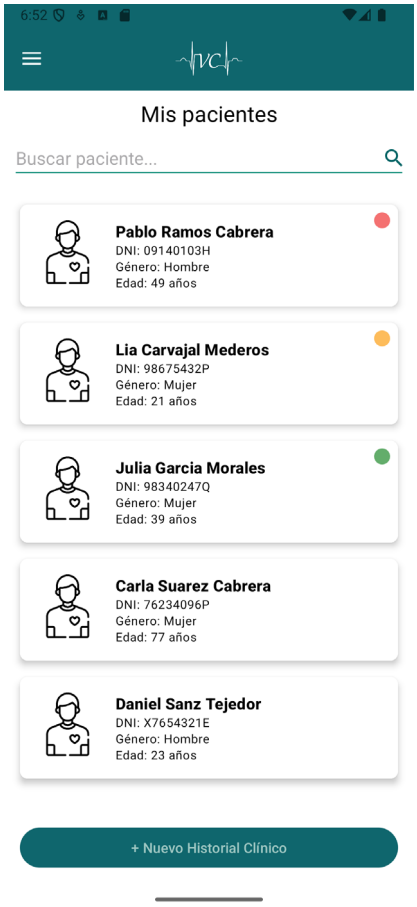
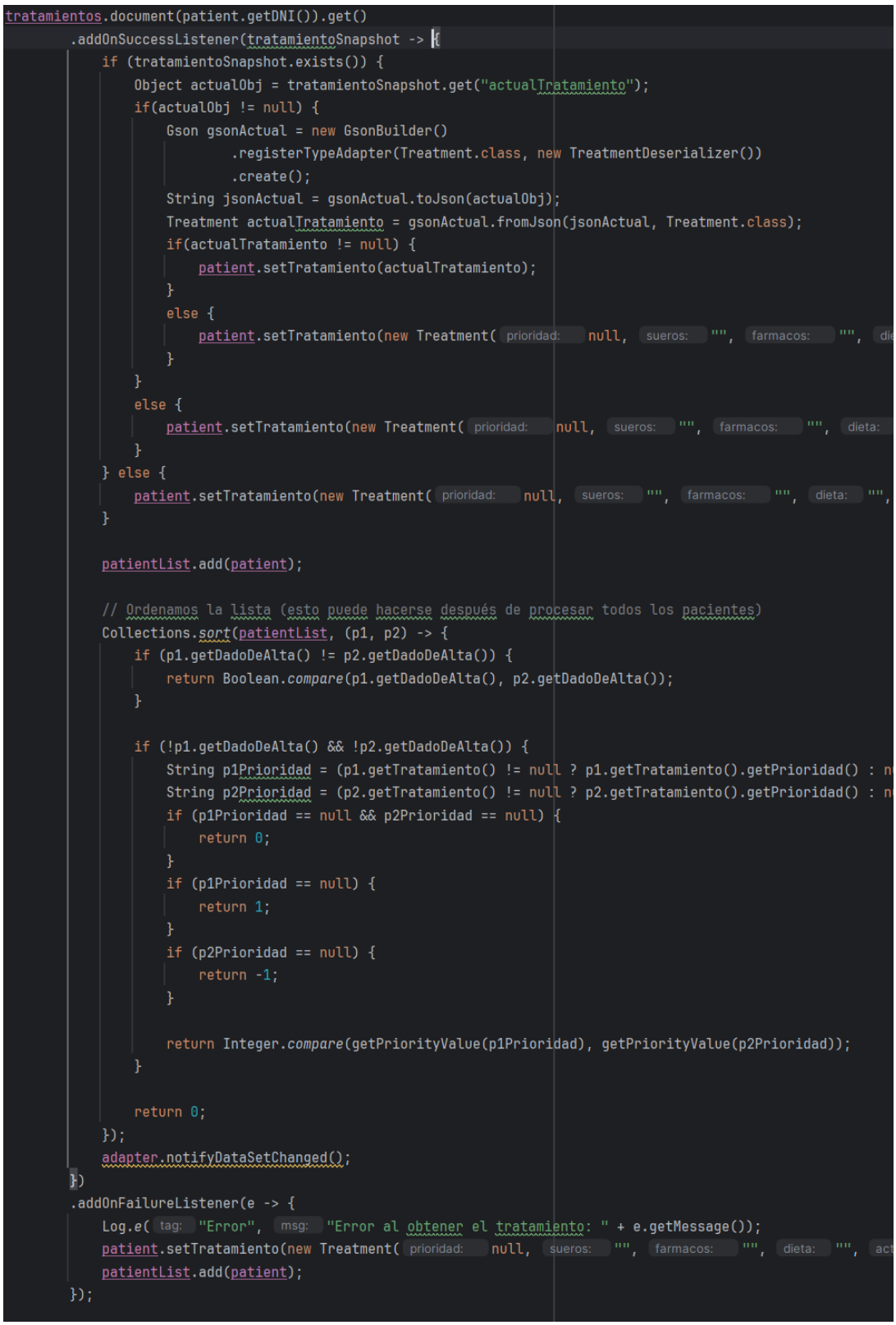


Figura 8-71. Listado de pacientes ordenados por tratamiento

La implementación de mostrar los pacientes ordenados se desarrolla en el mismo método que se muestra en la figura 8-26, en ese se hace la carga de los pacientes que tiene asignados el sanitario y a la par se van ordenando por la prioridad del tratamiento que ha sido seleccionada durante la creación del tratamiento. La implementación de esta ordenación se puede ver en la figura 8-72.



```
tratamientos.document(patient.getDNI()).get()
        .addOnSuccessListener(tratamientoSnapshot -> {
            if (tratamientoSnapshot.exists()) {
                Object actualObj = tratamientoSnapshot.get("actualTratamiento");
                if(actualObj != null) {
                    Gson gsonActual = new GsonBuilder()
                            .registerTypeAdapter(Treatment.class, new TreatmentDeserializer())
                            .create();
                    String jsonActual = gsonActual.toJson(actualObj);
                    Treatment actualTratamiento = gsonActual.fromJson(jsonActual, Treatment.class);
                    if(actualTratamiento != null) {
                        patient.setTratamiento(actualTratamiento);
                    }
                    else {
                        patient.setTratamiento(new Treatment( prioridad: null, sueros: "", farmacos: "", di
                    }
                }
                else {
                    patient.setTratamiento(new Treatment( prioridad: null, sueros: "", farmacos: "", dieta:
                }
            } else {
                patient.setTratamiento(new Treatment( prioridad: null, sueros: "", farmacos: "", dieta: "",
            }

            patientList.add(patient);

            // Ordenamos la lista (esto puede hacerse después de procesar todos los pacientes)
            Collections.sort(patientList, (p1, p2) -> {
                if (p1.getDadoDeAlta() != p2.getDadoDeAlta()) {
                    return Boolean.compare(p1.getDadoDeAlta(), p2.getDadoDeAlta());
                }

                if (!p1.getDadoDeAlta() && !p2.getDadoDeAlta()) {
                    String p1Prioridad = (p1.getTratamiento() != null ? p1.getTratamiento().getPrioridad() : n
                    String p2Prioridad = (p2.getTratamiento() != null ? p2.getTratamiento().getPrioridad() : n
                    if (p1Prioridad == null && p2Prioridad == null) {
                        return 0;
                    }
                    if (p1Prioridad == null) {
                        return 1;
                    }
                    if (p2Prioridad == null) {
                        return -1;
                    }

                    return Integer.compare(getPriorityValue(p1Prioridad), getPriorityValue(p2Prioridad));
                }

                return 0;
            });
            adapter.notifyDataSetChanged();
        })
        .addOnFailureListener(e -> {
            Log.e( tag: "Error", msg: "Error al obtener el tratamiento: " + e.getMessage());
            patient.setTratamiento(new Treatment( prioridad: null, sueros: "", farmacos: "", dieta: "", act
            patientList.add(patient);
        });
```

Figura 8-72. Implementación para ordenar por prioridad de tratamiento

Además, cuando se carga el cardView de cada paciente se muestra un círculo en el que se representa la prioridad que tiene el tratamiento del paciente, figura 8-73. Siendo de color rojo para prioridad Alta, ámbar para Media y verde para Baja. En caso de que el paciente no tenga un tratamiento asignado, no se mostrará ningún círculo.

Los pacientes con un tratamiento con prioridad alta estarán posicionados al principio de listado, después irán los de prioridad media y a continuación los de prioridad baja. Al final de la lista de pacientes estarán todos aquellos que no tengan ningún tratamiento en el momento.

```
if(patient.getTratamiento().getPrioridad() != null) {
    switch (patient.getTratamiento().getPrioridad()){
        case "Alta":
            holder.statusCircle.setBackgroundTintList(ColorStateList.valueOf(ContextCompat.getColor(doctorActivity, R.color.shadow_red)));
            break;
        case "Media":
            holder.statusCircle.setBackgroundTintList(ColorStateList.valueOf(ContextCompat.getColor(doctorActivity, R.color.shadow_orange)));
            break;
        case "Baja":
            holder.statusCircle.setBackgroundTintList(ColorStateList.valueOf(ContextCompat.getColor(doctorActivity, R.color.shadow_green)));
            break;
    }
}
```

Figura 8-73. Implementación para asignar color en función del tratamiento

8.3.14 Generar alerta por el paciente para los profesionales sanitarios

En casos de emergencia, los sanitarios pueden enviar una notificación de alerta al otro sanitario encargado de su cuidado. Para ello, deben acceder al paciente del que ser quiere avisar sobre la urgencia y pulsar sobre el botón de la campana para lanzar una notificación de alerta.

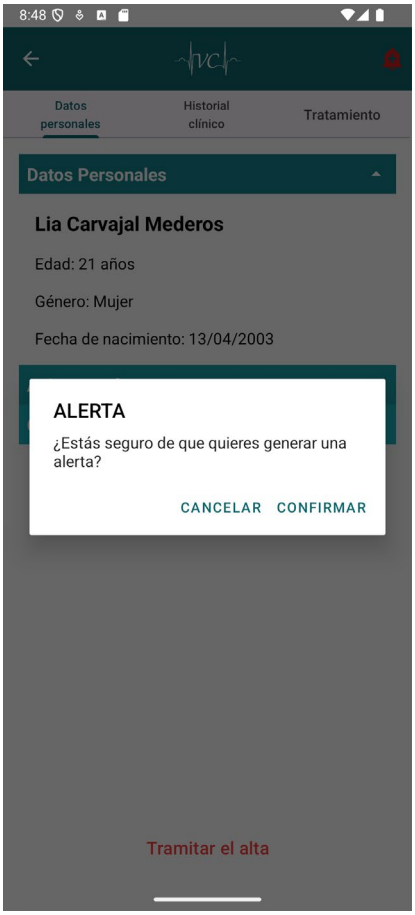


Figura 8-74. Generar alerta por paciente

Cuando el sanitario confirma que quiere general una alerta por el paciente como se ve en la figura 8-74, se registra en la base de datos el envío de una alerta en la colección notificaciones en el documento del sanitario destinatario de la alerta como se observa en el método de la figura 8-75. Además, mediante el método "enviarNotificacionTema" mostrado en la figura 8-76, se genera una notificación en tiempo real mediante los servicios de Firebase Cloud Messaging y Cloud Functions que llega al dispositivo del sanitario destinatario.

```java
public void saveNotification(String DNIworker, String titulo, String mensaje, String tabIndex, String dniP) {  6 usages
    DocumentReference docRef = fdb.collection( collectionPath: "notificaciones").document(DNIworker);

    docRef.get().addOnSuccessListener(documentSnapshot -> {
        Map<String, Object> existingNotifications = documentSnapshot.exists()
                ? documentSnapshot.getData()
                : new HashMap<>();

        // Encontrar el mayor indice numérico
        int nextIndex = 0;
        for (String key : existingNotifications.keySet()) {
            try {
                int index = Integer.parseInt(key);
                if (index >= nextIndex) {
                    nextIndex = index + 1;
                }
            } catch (NumberFormatException ignored) {
                Log.e( tag: "Firestore",  msg: "Clave no válida en el mapa de notificaciones");
            }
        }

        // Crear nueva notificación
        Map<String, Object> newNotification = new HashMap<>();
        newNotification.put( k: "fecha", Timestamp.now());
        newNotification.put( k: "mensaje", mensaje);
        newNotification.put( k: "titulo", titulo);
        newNotification.put( k: "tabIndex", tabIndex);
        newNotification.put( k: "dniP", dniP);

        // Guardar en la base de datos
        docRef.update(String.valueOf(nextIndex), newNotification)
                .addOnSuccessListener(aVoid -> Log.d( tag: "Firestore",  msg: "Notificación guardada correctamente"))
                .addOnFailureListener(e -> {
                    Log.e( tag: "Firestore",  msg: "Error al guardar notificación", e);

                    // Si falla, probablemente el documento no existe, así que lo creamos
                    Map<String, Object> firstNotification = new HashMap<>();
                    firstNotification.put( k: "0", newNotification);
                    docRef.set(firstNotification)
                            .addOnSuccessListener(aVoid2 -> Log.d( tag: "Firestore",  msg: "Documento creado con la primera notificación"))
                            .addOnFailureListener(e2 -> Log.e( tag: "Firestore",  msg: "Error al crear documento", e2));
                });

    }).addOnFailureListener(e -> Log.e( tag: "Firestore",  msg: "Error al obtener documento", e));
}
```

Figura 8-75. Implementación para guardar registro de notificación

```
public static void enviarNotificacionTema(String tema, String titulo, String mensaje, String tabIndex, String dniP) {
    Map<String, Object> datos = new HashMap<>();
    datos.put( k: "tema", tema);
    datos.put( k: "titulo", titulo);
    datos.put( k: "mensaje", mensaje);
    datos.put( k: "tabIndex", tabIndex);
    datos.put( k: "dniP", dniP);

    mFunctions
            .getHttpsCallable( name: "enviarNotificacionTema") HttpsCallableReference
            .call(datos) Task<HttpsCallableResult>
            .addOnSuccessListener(httpsCallableResult -> {
                Log.d( tag: "Notificación",  msg: "Notificación enviada exitosamente al tema.");
            })
            .addOnFailureListener(e -> {
                Log.e( tag: "Notificación",  msg: "Error al enviar la notificación al tema", e);
            });
}
```

Figura 8-76. Implementación para enviar notificación

8.3.15 Tramitar el alta del paciente

Cuando un paciente se ha recuperado y ya no necesita recibir atención hospitalaria, el médico puede tramitar su alta para que así deje de ocupar recursos del hospital y personal sanitario. Además, es una forma de organizar los pacientes de los profesionales sanitarios y que sepan que ese paciente ya no requiere de sus cuidados.

Para tramitar el alta del paciente, el médico debe pulsar sobre el botón "Tramitar el alta" que hay en la pestaña "Datos personales" tal y como se muestra en la figura 8-77.

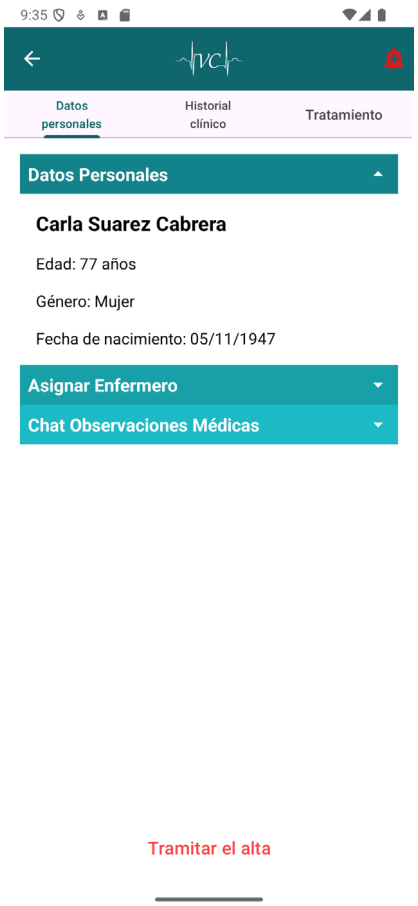


Figura 8-77. Sección para tramitar alta del paciente

Esto se verá reflejado en el listado de pacientes de los sanitarios teniendo un color grisáceo el cardView del paciente dado de alta como se ve en la figura 8-78. De esta manera, el cardView queda deshabilitado y se deniega el acceso a los sanitarios de su información para evitar errores en consultas o registros de datos con otros pacientes. Además, el sistema posiciona al paciente al final del listado de los pacientes del sanitario, lo que reduce su relevancia en relación con los demás.

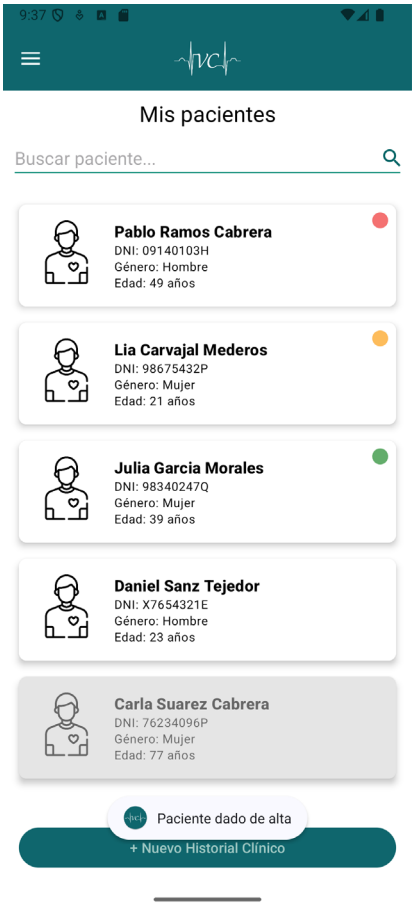


Figura 8-78. Listado de paciente con paciente dado de alta

Cuando se formaliza la tramitación del alta se actualiza la información guardada en la base de datos sobre el paciente, figura 8-79. En el historial clínico se mantiene todos los datos que se le tomaron cuando ingresó, pero se modifica el motivo de ingreso y se deja en blanco ese campo. A su vez, los tratamientos que se habían creado para el paciente, las observaciones anotadas y las constantes vitales que se le tomaron son eliminadas para que en caso de que el paciente tenga que volver a ser ingresado, su información esté actualizada y limpia para poder ser utilizada en el momento.

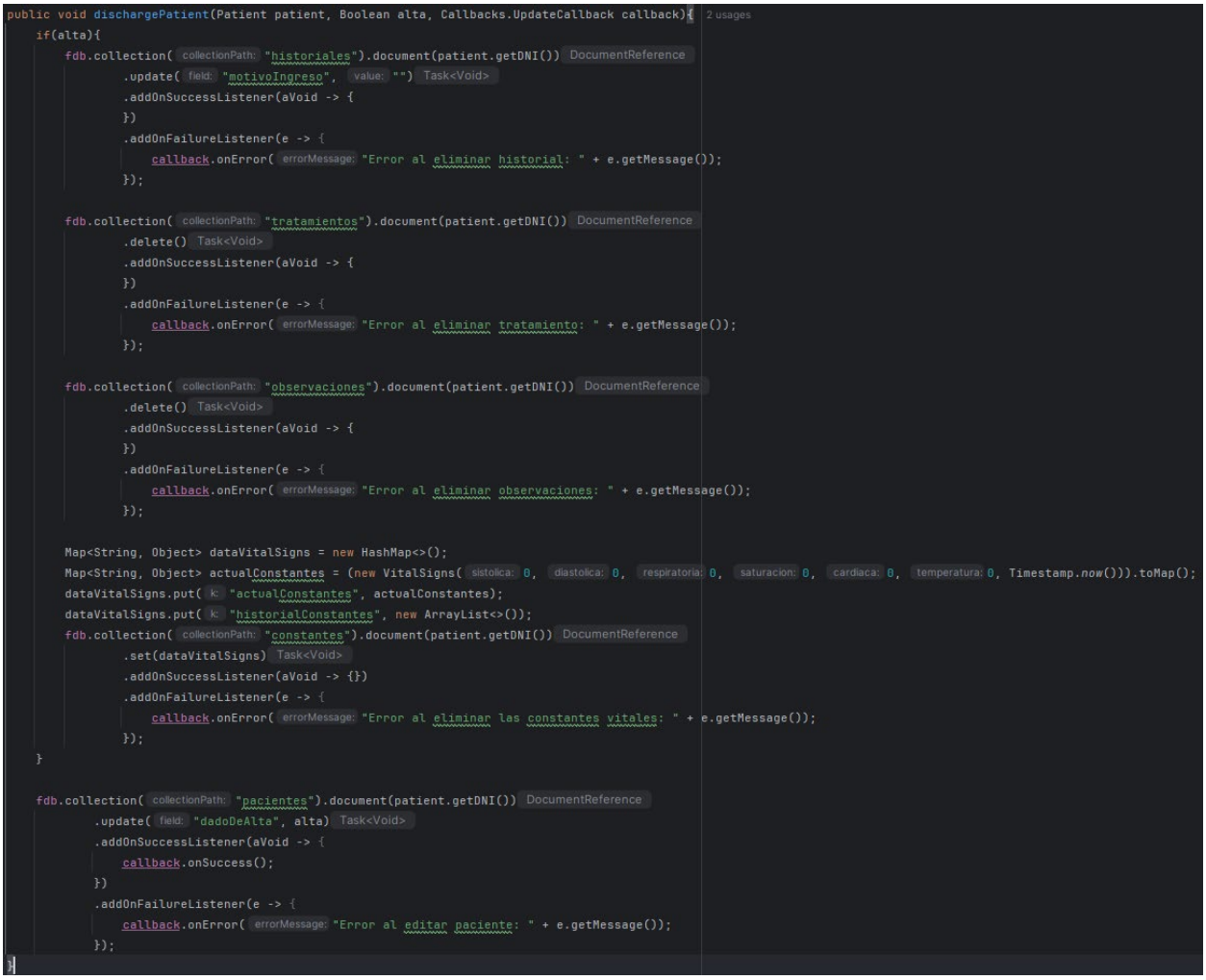


Figura 8-79. Implementación para actualizar información con el alta del paciente

8.3.16 Reingresar a paciente dado de alta

Si el paciente requiere de nuevo de los servicios sanitarios y tiene que ser internado en planta habiendo estado ingresado anteriormente, al estar registrado en el sistema, el médico puede procesar el reingreso del paciente pulsando sobre el cardView del paciente y confirmando su reingreso como se observa en la figura 8-80. El sistema le muestra al médico la sección de datos personales para que así pueda empezar a modificar los datos que quiera sobre su nuevo ingreso rápidamente.

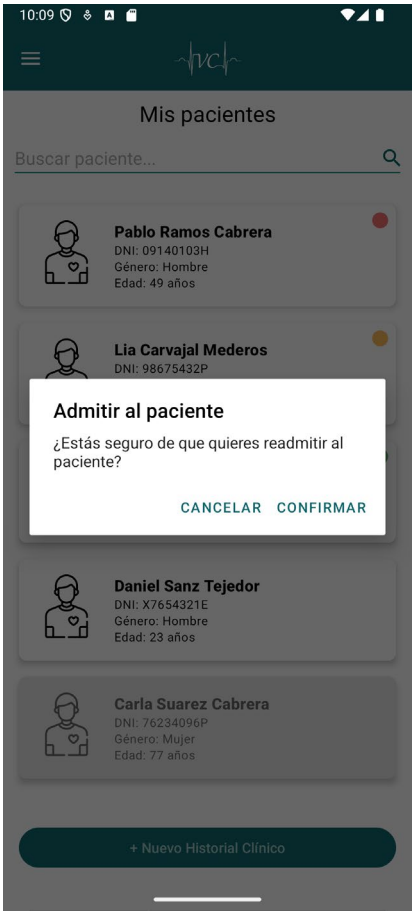


Figura 8-80. Solicitud de confirmación para reingresar a paciente

Para conseguir esta lógica el sistema comprueba que el que está intentando readmitir al paciente es el médico ya que es el único que puede tomar la decisión. Una vez el sistema ha hecho las comprobaciones que se muestran en la figura 8-81 y el médico confirma el reingreso, se actualiza la base de datos con el mismo método de la figura 8-79 para que se refleje que está interno en el hospital de nuevo. Con esto, conseguimos que el enfermero que le había atendido en su anterior ingreso se le muestre habilitado el paciente en su listado y así el enfermero sea consciente de su reingreso y pueda acceder a su información de nuevo.

```
holder.itemView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (patient.getDadoDeAlta()) {
            if(fdb.getLoggedInWorker().getRol().equals("Médico")){
                new AlertDialog.Builder(doctorActivity)
                        .setTitle("Admitir al paciente")
                        .setMessage("¿Estás seguro de que quieres readmitir al paciente?")
                        .setPositiveButton( text: "Confirmar", (dialog, which) -> { //El usuario confirma
                            fdb.dischargePatient(patient,  alta: false, new Callbacks.UpdateCallback() {
                                @Override
                                public void onSuccess() {
                                    patient.setDadoDeAlta(false);
                                    Toast.makeText(doctorActivity,  text: "Paciente readmitido", Toast.LENGTH_SHORT).show();

                                    Intent intent = new Intent(v.getContext(), InfoPatientActivity.class);
                                    intent.putExtra( name: "DNI", patient.getDNI());
                                    v.getContext().startActivity(intent);
                                }

                                @Override
                                public void onError(String errorMessage) {
                                    Toast.makeText(doctorActivity, errorMessage, Toast.LENGTH_SHORT).show();
                                }
                            });
                        })
                        .setNegativeButton( text: "Cancelar", (dialog, which) -> { //El usuario cancela
                            dialog.dismiss();
                        })
                        .show();
            }
            else if(fdb.getLoggedInWorker().getRol().equals("Enfermero")){
                holder.itemView.setOnClickListener(null);
                Toast.makeText(doctorActivity,  text: "El paciente está dado de alta", Toast.LENGTH_SHORT).show();
            }
        }
        else {
            Intent intent = new Intent(v.getContext(), InfoPatientActivity.class);
            intent.putExtra( name: "DNI", patient.getDNI());
            v.getContext().startActivity(intent);
        }
    }
});
```

Figura 8-81. Implementación de comprobaciones para reingresar a paciente

8.3.17 Leer notificaciones

Las notificaciones se pueden generar de varias maneras con el fin de facilitar la comunicación entre enfermero y médico. Los motivos por los cuales se crea una notificación son: cuando un sanitario ha generado una alerta por un paciente, el

médico ha creado un nuevo tratamiento, el tratamiento ha sido modificado por el médico, uno de los sanitarios ha anotado una nueva observación en el chat, se han registrado nuevas constantes vitales o el médico ha asignado al enfermero un nuevo paciente. Todos estos tipos de notificaciones se pueden observar en la figura 8-82.

Figura 8-82. Listado de notificaciones recibidas

Como ya explicamos en este capítulo en el apartado de la implementación del Login y en el capítulo 3 en donde detallamos las tecnologías empleadas, para la gestión de notificaciones hemos utilizado Cloud Messaging y Cloud Functions.

En el apartado en el que mostramos el funcionamiento de la generación de la alerta ya detallamos el funcionamiento del envío de las notificaciones y almacenamiento de estas en la base de datos, por lo que ahora nos vamos a centrar en detallar la recepción y visionado de estas.

Cuando el dispositivo del sanitario recibe una notificación en su dispositivo móvil se muestra como una notificación flotante como se muestra en la figura 8-83, y también está visible en el panel de notificaciones, figura 8-84.

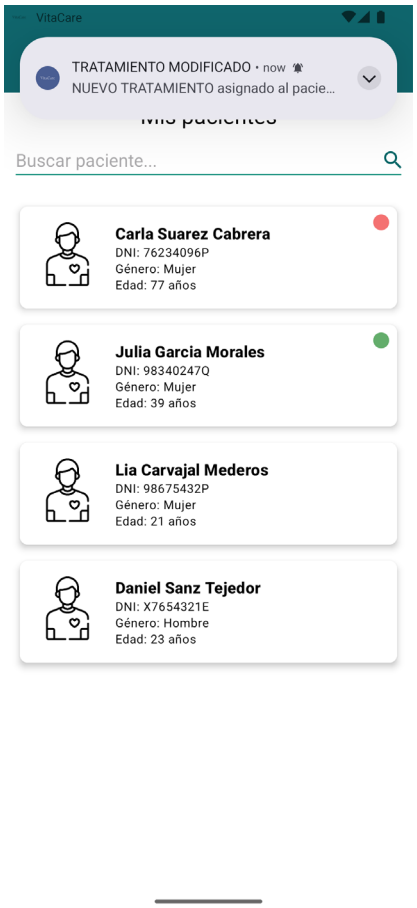


Figura 8-83. Notificación flotante entrante de la aplicación

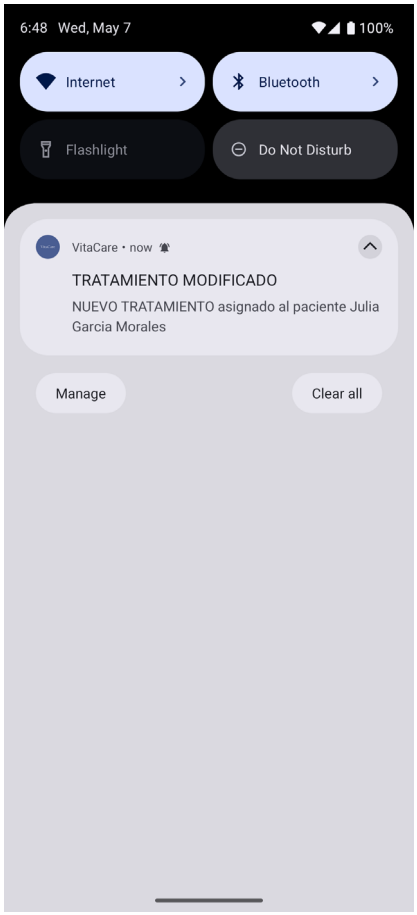


Figura 8-84. Panel de notificaciones del móvil con notificación de la aplicación

Cuando el sanitario recibe la notificación puede pulsar sobre ella para que el sistema le redireccione automáticamente a la pantalla con el listado completo de todas las notificaciones que ha recibido como se muestra en la figura 8-82. Para implementar esta lógica, figura 8-85, lo que se ha hecho es crear una clase para gestionar la visualización y acciones de las notificaciones. Cuando reciben la notificación, si pulsan sobre ella el sistema abre la aplicación y les redirecciona a esta nueva pantalla.

```java
@Override  3 usages
public void onMessageReceived(RemoteMessage remoteMessage) {
    super.onMessageReceived(remoteMessage);
    showNotification(remoteMessage);
}

@Override  2 usages
public void onNewToken(String token) {
    super.onNewToken(token);
    Log.d(TAG,  msg: "Nuevo Token: " + token);
}

private void showNotification(RemoteMessage remoteMessage) {  1 usage
    //Abrir la NotificationActivity
    Intent intent = new Intent( packageContext: this, NotificationActivity.class);

    TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
    stackBuilder.addParentStack(NotificationActivity.class);
    stackBuilder.addNextIntent(intent);
    pendingIntent = stackBuilder.getPendingIntent( requestCode: 0,
            flags: PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE);

    if (pendingIntent == null) {
        Log.e( tag: "FCMessagingService",  msg: "Error: PendingIntent no se creó correctamente");
    }

    // Construir la notificación
    Notification notification = new NotificationCompat.Builder( context: this, Login.CHANNEL_ID)
            .setSmallIcon(R.drawable.ic_notification)
            .setContentTitle(remoteMessage.getNotification().getTitle())
            .setContentText(remoteMessage.getNotification().getBody())
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .setAutoCancel(true)
            .setContentIntent(pendingIntent)
            .build();

    // Mostrar la notificación
    NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

    if (notificationManager != null) {
        notificationManager.notify( id: 1, notification);
    }
}
```

Figura 8-85. Implementación para gestionar notificaciones

Una vez el sistema ha redireccionado correctamente, aparece un listado con todas las notificaciones recibidas mostradas en orden de llegada, como se muestra en la figura 8-82.

Para que sea más sencillo y así ayudar a los sanitarios a identificar rápidamente qué tipo de notificación es la que han recibido, hemos asignado a cada notificación un color como se observa en la figura 8-86, para que de esta forma sin necesidad de leer la cabecera de la notificación puedan identificarla rápidamente. Para ello, cuando enviamos y almacenamos una notificación en la base de datos además de guardar el destinatario, la fecha, la descripción o mensaje que va a ir incluido, tenemos un título que es lo que nos sirve para identificar qué tipo de notificación ha recibido el sanitario.

```
switch (notification.getTitulo()) {
    case "ALERTA":
        colorTituloFondo = ContextCompat.getColor(context, R.color.alerta1);
        break;
    case "NUEVO TRATAMIENTO":
        colorTituloFondo = ContextCompat.getColor(context, R.color.tratamiento1);
        break;
    case "TRATAMIENTO MODIFICADO":
        colorTituloFondo = ContextCompat.getColor(context, R.color.editado1);
        break;
    case "NUEVAS OBSERVACIONES":
        colorTituloFondo = ContextCompat.getColor(context, R.color.observaciones1);
        break;
    case "NUEVAS CONSTANTES VITALES":
        colorTituloFondo = ContextCompat.getColor(context, R.color.constantes1);
        break;
    case "NUEVO PACIENTE":
        colorTituloFondo = ContextCompat.getColor(context, R.color.nuevopaciente1);
        break;
    default:
        colorTituloFondo = ContextCompat.getColor(context, R.color.gris);
        break;
}

holder.layoutTitulo.setBackgroundColor(colorTituloFondo);
```

Figura 8-86. Implementación para asignar color por tipo notificación

Además, se ha implementado una lógica con la que el sanitario si lo desea pueda ver la información relacionada con la notificación. Por ejemplo, se le ha notificado de nuevas constantes vitales, si el sanitario pulsa sobre el cardView de esta, el sistema le redirige automáticamente a la pestaña y sección en la que está la información relacionada con la notificación, en este caso sería las nuevas constantes vitales del paciente especificado en la notificación.

Esto se consigue almacenando en la base de datos junto con el resto de campos de la notificación el dni del paciente sobre el que se ha generado la notificación y un índice que marca la pestaña en la que se encuentra la sección que contiene los datos a visionar. Esta lógica de redirección queda reflejada en la siguiente figura, la 8-87.

```
holder.itemView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //Verificar si el paciente está dado de alta o si está asignado correctamente
        fdb.verifyPatient(notification.getDniP(), new Callbacks.VerificarPacienteCallback() {
            @Override  5 usages
            public void onResult(Boolean ok) {
                if (ok) {
                    Intent intent = new Intent(v.getContext(), InfoPatientActivity.class);
                    intent.putExtra( name: "DNI", notification.getDniP());
                    intent.putExtra( name: "tabIndex", Integer.valueOf(notification.getTabIndex()));
                    v.getContext().startActivity(intent);
                } else if(!ok){
                    Toast.makeText(context,  text: "El paciente está dado de alta o no lo tiene asignado.", Toast.LENGTH_SHORT).show();
                }
                else{
                    Toast.makeText(context,  text: "No se puede acceder a la informacion del paciente.", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
});
```

Figura 8-87. Implementación para redireccionar desde la notificación

Por último, el sanitario tiene la opción de hacer un borrado de todas las notificaciones entrantes, de esta forma conseguimos evitar que se les acumulen y que sea más sencillo para ellos poder tener una visión más limpia y focalizar su atención únicamente en las más recientes.

Esta opción se encuentra en la esquina superior derecha de la pantalla desde la que se pueden visualizar las notificaciones como se aprecia en la figura 8-88. Cuando el sanitario pulsa el botón "Limpiar bandeja de entrada", el sistema le pide una doble confirmación para asegurar que no ha sido una equivocación. Una vez el sistema obtiene la doble verificación, hace un borrado de todas las notificaciones almacenadas en la base de datos de dicho sanitario.

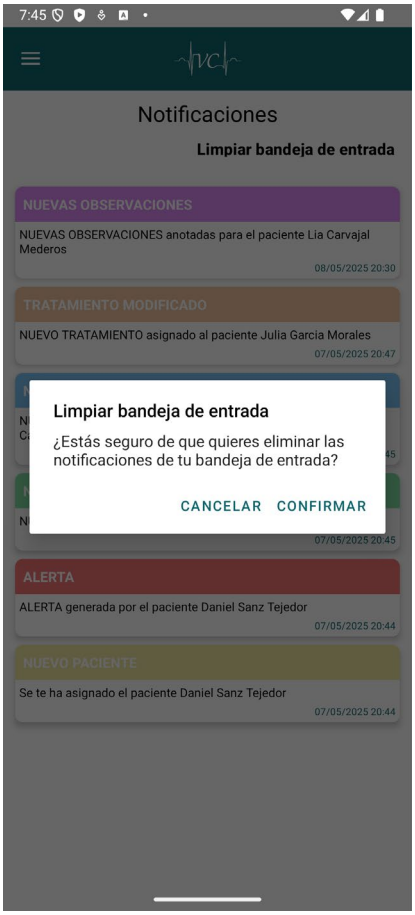


Figura 8-88. Confirmación para eliminar notificaciones

Capítulo 9 - Conclusiones y trabajo futuro

9.1 Conclusiones

En este proyecto se ha diseñado y desarrollado un sistema de gestión de información de pacientes ingresados en un hospital, con el objetivo de transformar y modernizar el acceso a la información de los tratamientos y cuidados en los centros sanitarios. En el transcurso de este ha quedado reflejado cómo la digitalización de procesos manuales a través de una aplicación móvil puede mejorar la atención proporcionada al paciente y la coordinación entre distintos profesionales sanitarios.

Entre las distintas funcionalidades del sistema destaca la centralización de la información clínica, posibilitando el acceso en tiempo real a los datos para la toma de decisiones médicas, lo que facilita poder ofrecer una asistencia sanitaria más rápida, segura y personalizada. Esto se consigue mediante la corrección de erratas, mejora en la coordinación de los procesos que previamente eran analógicos y creación de un entorno de interacción y comunicación más fluido entre médicos y enfermeros más fluido.

Se consideró llevar a cabo una evaluación empírica del sistema por medio de pruebas con usuarios y análisis de rendimiento en situaciones reales, las limitaciones de tiempo y recursos dificultaron la implementación de esta validación completa. No obstante, somos conocedores de la importancia de analizar el desempeño y la efectividad del sistema en situaciones reales. Por ello, este aspecto se propone como una línea esencial para investigaciones y desarrollos futuros.

Además, dada la sensibilidad de los datos médicos gestionados, desde el inicio del proyecto se tuvo en cuenta la necesidad de incorporar medidas de seguridad y privacidad, como la aplicación de protocolos de protección de datos y el cumplimiento del Reglamento General de Protección de Datos (RGPD). Por consiguiente, se implementaron mecanismos básicos de protección, como el almacenamiento de contraseñas hasheadas en la base de datos para evitar la suplantación de identidad. No obstante, dado que el desarrollo de un sistema de seguridad integral no se estableció

como uno de los objetivos principales del proyecto, y considerando el limitado tiempo disponible, este tratamiento se centró en aspectos esenciales. Por tanto, para futuras versiones del sistema es crucial profundizar en los protocolos de seguridad, en conformidad con normativas como el RGPD, para garantizar una protección más robusta de la información y los datos médicos tratados.

Finalmente, con este proyecto sentamos las bases para posibles futuras ampliaciones y mejoras que se detallan más adelante, ya que con la diversidad de tecnologías existentes hoy en día y la evolución que hay constantemente se abren las puertas a integraciones con otros sistemas externos y al desarrollo de nuevas funcionalidades según las necesidades que vayan apareciendo en el ámbito sanitario.

El código fuente de la aplicación está disponible en el siguiente enlace a través de Github:

https://github.com/AlbertoGomezdeAndres/TFG_VitaCare.git

9.2 Trabajo futuro

El desarrollo de este proyecto ha abierto múltiples líneas de mejora y ampliación, tanto en términos de funcionalidades como de integración con sistemas existentes y estándares internacionales. A continuación, se detallan las propuestas específicas:

9.2.1 Pruebas complementarias

Sería interesante poder añadir el servicio de pruebas complementarias con su correspondiente usuario y su interfaz. Con esto podemos ofrecer una aplicación más completa dándoles la posibilidad a los sanitarios de solicitar una prueba complementaria como pueden ser pruebas de laboratorio (analítica de sangre, de orina…), pruebas de imagen (radiografía, TAC, ecografía…) o de otras especialidades. Además de poder solicitar las pruebas, se crearía un actor entre los usuarios que se encargue de la gestión de las pruebas complementarias, reciba las solicitudes de pruebas y finalmente registre los resultados de las pruebas para que el médico las pueda consultar.

9.2.2 Importar documentos y archivos especiales

En relación con el punto anterior, para perfeccionar el servicio de pruebas complementarias, sería muy positivo poder importar a la aplicación documentos PDF, en donde se pueda resumir los resultados de las pruebas, ver los valores de una analítica de sangre… Al mismo tiempo, se podría intentar implementar un sistema de visionado de las pruebas de imágenes para que el médico, además de consultar el informe del especialista que analizó las imágenes de las pruebas, pueda revisarlas en el momento sin necesidad de desplazarse hasta un ordenador de escritorio.

9.2.3 Responsive

Sería recomendable hacer que la aplicación fuera completamente responsive, para que así se adapte de manera óptima a diferentes tipos de dispositivos en los que el tamaño de la pantalla sea distinto. Aunque gran parte de la aplicación ya es responsive, también sería favorable que el tamaño de la fuente fuera variable en proporción con el tamaño de la pantalla para que fuese más agradable en el uso y proporcione una experiencia consistente, de calidad y mejorada al usuario sin importar el dispositivo que utilice.

9.2.4 Accesibilidad para personas con discapacidad visual

Ante la posibilidad de encontrar sanitarios que padezcan problemas visuales tales como el albinismo o el daltonismo, sería de gran utilidad que la aplicación fuese lo más accesible posible. Algunas de las medidas que se pueden implementar son: asistente de voz, transcripción de pantalla, lectores digitales, posibilidad de hacer aumento máximo de pantalla y crear un buen contraste de color entre el fondo y el texto.

9.2.5 Inteligencia artificial y redes neuronales

Podría ser de gran utilidad añadirle una capa de inteligencia artificial (IA) y redes neuronales para que pueda ayudar al médico con el tratamiento del paciente. Se podría desarrollar una IA que fuera capaz de proponer al médico un posible diagnóstico mediante los síntomas que presente el paciente, así como dar posibles tratamientos

para dicho diagnóstico. Esto se podría implementar contando con un amplio dataset para poder entrenar a la IA y añadirle redes neuronales para mejorar la precisión de las predicciones que haga esta.

9.2.6 Aplicación multiplataforma

Para hacer posible la integración de la aplicación en los centros sanitarios a nivel masivo, sería muy recomendable desarrollarla en multiplataforma. Así, podríamos evitar las barreras de los sistemas operativos que tenga cada centro sanitario, sin impedir el acceso a ella. Además, con el desarrollo de nuevas tecnologías no sería necesario desarrollarla por separado en código Kotlin o Java para Android y Swift para IOS, sino que con el uso de KMM (Kotlin Multiplatform) [17] sería más sencillo y rápido poder llevarla a cabo.

9.2.7 Versión programa para escritorio

Se podría considerar desarrollar un programa de escritorio complementario a la aplicación móvil ya existente. Aunque la aplicación móvil se ha creado para ofrecer movilidad y poder liberar al sanitario de la dependencia de un ordenador, contar con una versión de escritorio permitiría consultar y modificar la información a través de una pantalla más amplia, utilizando una base de datos común. Esto disminuiría el esfuerzo visual que supondría usar el dispositivo móvil durante un tiempo prolongado o tras una guardia de 24 horas y enriquecería la experiencia del usuario.

9.2.8 Integración con sistemas hospitalarios y adopción de estándares internacionales

Para facilitar la integración de la aplicación en entornos hospitalarios modernos, se propondría desarrollar capacidades de interoperabilidad con sistemas existentes, como SIH (Sistema de Información Hospitalaria), RME (Registro Médico Electrónico) o HCE. La adopción de estándares internacionales en el ámbito sanitario, en particular HL7 FHIR (Recursos de Interoperabilidad Rápida en Salud desarrollado por Health Level Seven), permitiría un intercambio seguro y eficiente de datos clínicos. Esta integración

bidireccional facilitaría el acceso a información actualizada y promovería una gestión coordinada entre distintas plataformas, lo que resultaría en una mayor eficiencia asistencial.

Conclusions and future work

This project has been designed to develop an information management system for patients admitted to a hospital, with the aim of transforming and modernising the administration of treatments in healthcare centres. Throughout the project, it has been shown how the digitalisation of manual processes through a mobile application can improve patient care and coordination between different healthcare professionals.

Among the different functionalities of the system, the centralisation of clinical information stands out, enabling real-time access to data for medical decision-making, which facilitates faster, safer and more personalised healthcare for patients. This is achieved by correcting errors, improving the coordination of previously analogue processes, and creating a more fluid environment for interaction and communication between doctors and nurses.

An empirical evaluation of the system through user testing and real-life performance analysis was considered; however, time and resources constraints made it difficult to implement this full validation. Nevertheless, we acknowledge the importance of analysing the system's performance and effectiveness in real scenarios. For this reason, this aspect is proposed as an essential line for future research and development.

Moreover, due to the sensitivity of the medical data being handled, the need to incorporate security and privacy measures, such as data protection protocols and General Data Protection Regulation (GDPR) compliance, was considered from the beginning of the project. Therefore, basic protection mechanisms were implemented, such as the storage of hashed passwords in the database to prevent identity theft. However, since the development of a comprehensive security system was not established as one of the project's main objectives, and considering the limited time available, the approach was focused on essential aspects. For future versions of the system, it is crucial to enhance security protocols, in accordance with regulations such as the GDPR, to ensure a more robust protection of the information and medical data processed.

Finally, with this project we lay the groundwork for possible future expansions and improvements, as detailed below. With The diversity of technologies available today and their constant evolution, there is great potential for integration with other external systems and for development of new functionalities based on emerging needs in the healthcare field.

The application's source code is available at the following link through Github:

https://github.com/AlbertoGomezdeAndres/TFG_VitaCare.git

Future Work

The development of this project has opened up multiple ways for improvement and expansion, both in terms of functionality and integration with existing systems and international standards. The following section details the specific proposals:

Complementary tests

It would be interesting to be able to add the service of complementary tests with its corresponding user and interface. This would allow us to offer a more complete application, giving healthcare professionals the possibility of requesting complementary tests such as laboratory tests (blood tests, urine tests, etc.), imaging tests (X-rays, CT scans, ultrasound scans, etc.) or tests in other specialities. In addition to being able to request tests, an actor would be created among the users who would be in charge of managing complementary tests, receiving test requests and then, recording the results of the tests so that the doctor can consult them.

Importing special documents and files

In relation to the previous point, in order to improve the complementary tests service, it would be very positive to be able to import PDF documents into the application, where the results of the tests can be summarised, the values of a blood test can be viewed… At the same time, an attempt could be made to implement a system for viewing the image tests so that the doctor can consult the report of the specialist who

analysed the images of the tests, as well as reviewing them at the same time without having to go to a desktop computer.

Responsive

It would be advisable to make the application fully responsive, so that it adapts optimally to different types of devices with different screen sizes. Although much of the application is already responsive, it would also be beneficial for the font size to be variable in proportion to the screen size to make it more pleasant to use and provide a consistent, high-quality and enhanced user experience regardless of the device being used.

Accessible for the visually impaired

Given the possibility of encountering healthcare professionals with visual impairments such as albinism or colour blindness, it would be beneficial for the application to be as accessible as possible. Some measures that could be implemented include: a voice assistant, screen transcription, digital readers, the possibility of maximum screen magnification, and creating a strong colour contrast between the background and text.

Artificial intelligence and neural networks

It could be very useful to add a layer of artificial intelligence (AI) and neural networks so that it can help the doctor with the treatment of the patient. An AI could be developed in order to help proposing a possible diagnosis to the doctor based on the patient's symptoms, as well as giving possible treatments for such diagnosis. This could be implemented with a large dataset to train the AI and add neural networks to improve the accuracy of its predictions.

Cross-platform application

In order to integrate the application in healthcare centres on a mass scale, it would be highly recommended to develop it on a multiplatform basis. In this way, we

could avoid the barriers of the operating system that each health centre has, without impeding access to it. Furthermore, with the development of new technologies it would not be necessary to develop it separately in Kotlin or Java code for Android and Swift for IOS. Instead, using KMM (Kotlin Multiplatform) [17] would make it simpler and faster to carry it out.

Desktop software version

Considering developing a desktop programme to complement the existing mobile application could be a good idea. Although the mobile application has been developed to enable mobility and to free healthcare professionals from being dependent on a computer, a desktop version would allow information to be viewed and modified on a larger screen, using a common database. This would not only reduce the visual string of using a mobile device during a long period of time or after a 24-hour shift, but also enhance the user experience.

Integration with hospital systems and adoption of international standards

To facilitate the application's integration into modern hospital environments, it is proposed to develop interoperability capabilities with existing systems such as HIS (Hospital Information System), EMR (Electronic Medical Record), or EHR. The adoption of international standards in the healthcare field, particularly HL7 FHIR (Fast Healthcare Interoperability Resources developed by Health Level Seven), would enable secure and efficient exchange of clinical data. This bidirectional integration would facilitate access to updated information and promote coordinated management across different platforms, resulting in greater healthcare efficiency.

CONTRIBUCIONES PERSONALES

En este capítulo se describirán las aportaciones individuales que ha realizado cada miembro en el proyecto.

Lia Carvajal Mederos

- **Especificación de requisitos**

  La definición de requisitos es una fase esencial en el desarrollo del proyecto, fue la primera tarea realizada y la base sobre la cual se organizó todo el trabajo. En esta etapa se llevó a cabo un análisis detallado de las necesidades, las funcionalidades y las características que debía cumplir el proyecto, elaboramos un diagrama de casos de uso y la descripción de los distintos casos de uso. Este proceso fue realizado por ambos.

- **Diseño y creación de la base de datos**

  Se diseño la base de datos del proyecto estructurándola para poder organizar de manera correcta todos los datos con los que se iban a trabajar en la aplicación. Este proceso fue realizado por ambos.

- **Decisión de tecnologías a utilizar**

  Una vez conocidas las necesidades de la aplicación, el siguiente paso fue seleccionar las tecnologías a utilizar. Para ello, se optó por aquellas herramientas que son más usadas en la actualidad, tener en consideración cuales resultaban más cómodas de utilizar y conllevaban un rápido aprendizaje. Además, aprovechamos los conocimientos con los que contaba Lia previamente ya que había cursado la asignatura Programación de Aplicaciones para Dispositivos Móviles. Este proceso fue realizado por ambos.

- **Creación de repositorio Github**

Se creó un repositorio Github para almacenar y gestionar nuestro código fuente de manera colaborativa. Esto nos ha permitido tener un control y registro de todos los cambios realizados en el código en el desarrollo del proyecto.

- **Diseño de bocetos de la interfaz para la aplicación**

  El diseño de bocetos preliminares de la interfaz de la aplicación fue fundamental para plasmar la organización visual y funcional del proyecto desde sus inicios, además de elegir la gama de colores que queríamos usar en nuestra interfaz que fuera familiar con el ámbito sanitario. Estos bocetos nos permitieron definir de forma clara y estructurada la disposición de elementos, la navegación entre pantallas y la interacción prevista para el usuario. Buscamos que fuera una aplicación intuitiva, atractiva visualmente y de fácil aprendizaje en su uso para los sanitarios. Este proceso fue realizado por ambos.

- **Creación del SideBar**

  Se diseñó e implementó el SideBar (menú desplegable de la izquierda de la aplicación) para navegar de forma más fácil e intuitiva.

- **Participación en la gestión de cuentas**

  Se desarrollo un sistema de autenticación para permitir a los usuarios registrados iniciar sesión en la aplicación, proporcionando sus NIF y contraseña. Además, también se implementó la opción de cerrar sesión.

- **Participación en la gestión de administrador**

  En el módulo de gestión de administrador se realizó la implementación de algunas funcionalidades. En primer lugar, se diseñó la interfaz del usuario administrador, y se creó el listado de pacientes y trabajadores con sus correspondientes acciones sobre los cardView para poder editar y eliminar los usuarios registrados.

- **Participación en la gestión de pacientes**

En el módulo de gestión de paciente se realizó la implementación de algunas funcionalidades. Entre las funcionalidades desarrolladas están todas aquellas relacionadas con la asignación del enfermero, el historial clínico y el tratamiento del paciente. Además, se diseñaron las diferentes pestañas para facilitar la navegación por ellas al sanitario.

- **Participación en la gestión de notificaciones**

  Se diseñó un servicio de notificaciones para mantener actualizado a los sanitarios de cualquier cambio sobre sus pacientes. Se desarrolló la lógica necesaria para crear el canal mediante Cloud Messaging y Cloud Functions, implementando la lógica para que el usuario al iniciar sesión se suscribiera a un tema para la recepción de las notificaciones y posterior desuscripción con el cierre de sesión. Además, se desarrolló la visualización y redirección automática de las notificaciones hacia la aplicación.

- **Redacción de la memoria del proyecto**

  La redacción de la memoria ha reflejado el trabajo realizado durante todo el año de manera clara, detallada y concisa, abordando los aspectos más significativos del proyecto. Este proceso fue realizado por ambos.

Alberto Gómez de Andrés

- **Especificación de requisitos**

  La definición de requisitos es una fase esencial en el desarrollo del proyecto, fue la primera tarea realizada y la base sobre la cual se organizó todo el trabajo. En esta etapa se llevó a cabo un análisis detallado de las necesidades, las funcionalidades y las características que debía cumplir el proyecto, elaboramos un diagrama de casos de uso y la descripción de los distintos casos de uso. Este proceso fue realizado por ambos.

- **Diseño y creación de la base de datos**

Se diseño la base de datos del proyecto estructurándola para poder organizar de manera correcta todos los datos con los que se iban a trabajar en la aplicación. Este proceso fue realizado por ambos.

- **Decisión de tecnologías a utilizar**

  Una vez conocidas las necesidades de la aplicación, el siguiente paso fue seleccionar las tecnologías a utilizar. Para ello, se optó por aquellas herramientas que son más usadas en la actualidad, tener en consideración cuales resultaban más cómodas de utilizar y conllevaban un rápido aprendizaje. Además, aprovechamos los conocimientos con los que contaba Lia previamente ya que había cursado la asignatura Programación de Aplicaciones para Dispositivos Móviles.

- **Creación de repositorio Github**

  Se creó un repositorio Github para almacenar y gestionar nuestro código fuente de manera colaborativa. Esto nos ha permitido tener un control y registro de todos los cambios realizados en el código en el desarrollo del proyecto.

- **Diseño de bocetos de la interfaz para la aplicación**

  El diseño de bocetos preliminares de la interfaz de la aplicación fue fundamental para plasmar la organización visual y funcional del proyecto desde sus inicios, además de elegir la gama de colores que queríamos usar en nuestra interfaz que fuera familiar con el ámbito sanitario. Estos bocetos nos permitieron definir de forma clara y estructurada la disposición de elementos, la navegación entre pantallas y la interacción prevista para el usuario. Buscamos que fuera una aplicación intuitiva, atractiva visualmente y de fácil aprendizaje en su uso para los sanitarios. Este proceso fue realizado por ambos.

- **Participación en la gestión de cuentas**

Para darle una mayor seguridad al proceso de inicio de sesión se implementó la opción de ocultar la contraseña, pero que también pudiese ser visible a elección del usuario. Además, se optó por almacenar la contraseña del usuario hasheada impidiendo la posible suplantación de identidad.

- **Participación en la gestión de administrador**

  En el módulo de gestión de administrador se realizó la implementación de algunas funcionalidades. Se implementó el formulario para registrar nuevos trabajadores y poder modificar los datos de estos.

- **Participación en la gestión de pacientes**

  En el módulo de gestión de paciente se realizó la implementación de algunas funcionalidades. Entre las funcionalidades desarrolladas están todas aquellas relacionadas con el registro de constantes vitales, el tratamiento del paciente y el chat de observaciones médicas entre el médico y enfermero. Además, se implementó una mejora sobre el diseño de las pestañas añadiendo secciones expandibles en cada pestaña para separar la información de estas.

- **Ayuda en la gestión de notificaciones**

  Se diseñó un servicio de notificaciones para mantener actualizado a los sanitarios de cualquier cambio sobre sus pacientes. Se colaboró en el despliegue e integración de las tecnologías de notificaciones en la aplicación, asegurando que el sistema funcionara de manera óptima y se adaptara correctamente al entorno implementado.

- **Implementación del diseño del logo y pantalla de carga**

  Además de contribuir en la conceptualización, se encargó de realizar y plasmar la idea mediante el programa GIMP, consiguiendo crear un logo y pantalla de carga único.

- **Redacción de la memoria del proyecto**

La redacción de la memoria ha reflejado el trabajo realizado durante todo el año de manera clara, detallada y concisa, abordando los aspectos más significativos del proyecto. Este proceso fue realizado por ambos.

BIBLIOGRAFÍA

[1] Grupo Quirónsalud, «Casiopea Mobility,» 27 Marzo 2023. [En línea]. Available: https://www.hgc.es/es/sala-prensa/innova-news/casiopea-mobility.

[2] GCM Clinical, «CGM Selene,» [En línea]. Available: https://www.cgm.com/esp_es/products/redes-de-salud/cgm-selene-1.html.

[3] GCM Clinical, «GCM Selene Mobility,» [En línea]. Available: https://www.cgm.com/esp_es/products/redes-de-salud/cgm-mobility.html.

[4] Dedalus, «HCIS,» [En línea]. Available: https://www.dedalus.com/spain/es/our-offer/products/hcis-can-be-implemented-using-a-best-of-breed-approach/.

[5] Android Developers, «Android Studio,» [En línea]. Available: https://developer.android.com/studio?hl=es-419.

[6] Github, Inc., «Github,» [En línea]. Available: https://github.com/apps/desktop.

[7] Google, «Firebase,» [En línea]. Available: https://firebase.google.com/?hl=es-419.

[8] Google, «Cloud Firestore,» [En línea]. Available: https://firebase.google.com/docs/firestore?hl=es-419.

[9] Google, «Firebase Cloud Messaging (FCM),» [En línea]. Available: https://firebase.google.com/docs/cloud-messaging?hl=es-419.

[10] Google, «Cloud Fuctions,» [En línea]. Available: https://firebase.google.com/docs/functions?hl=es-419.

[11] Oracle Corporation, «Java 8,» [En línea]. Available: https://www.java.com/es/download/.

[12] Netsacape Communications Corporation, Fundación Mozilla, «JavaScript,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/JavaScript.

[13] World Wide Web Consortium (W3C), «Extensible Markup Language (XML),» [En línea]. Available: https://www.w3.org/XML/.

[14] The GIMP Development Team, «GIMP - GNU Image Manipulation Program,» [En línea]. Available: https://www.gimp.org/.

[15] Oracle Corporation, «MySQL,» [En línea]. Available: https://dev.mysql.com/doc/.

[16] Ministerior de Sanidad, «Especialidades reconocidas en España,» [En línea]. Available: https://www.sanidad.gob.es/areas/profesionesSanitarias/profesiones/especialistasExtracomunitarios/docs/2020EspecialidadesRecoEspanaV2.pdf.

[17] JetBrains, «Kotlin Multiplatform,» [En línea]. Available: https://www.jetbrains.com/kotlin-multiplatform/.

APÉNDICES

Apéndice A - Manual de uso

El siguiente manual de usuario estará a la disposición de cualquier persona con acceso a la aplicación que quiera navegar por ella y realizar todas las funcionalidades existentes.

1. Gestión de cuentas

1.1 Login

La pantalla de Login se muestra al abrir la aplicación, siempre y cuando, el usuario tenga la sesión cerrada previamente. Se deben introducir el NIF y la contraseña en los campos correspondientes y pulsar en el botón de "Login" como se aprecia en la figura Apéndice A-1.

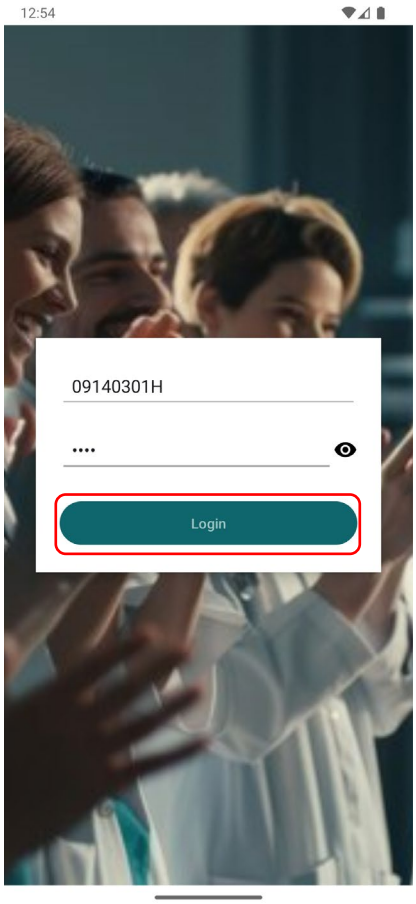


Figura Apéndice A-1. Pantalla de login

1.2 Logout

Al entrar en la aplicación, siempre será visible el toolbar superior, figura Apéndice A-2, desde el cual se puede acceder al menú lateral.

Figura Apéndice A-2. Toolbar

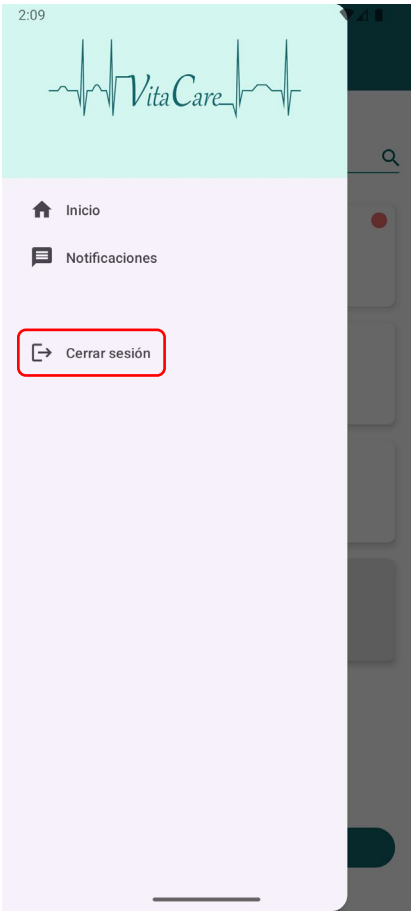


Figura Apéndice A-3. Menú lateral del profesional sanitario

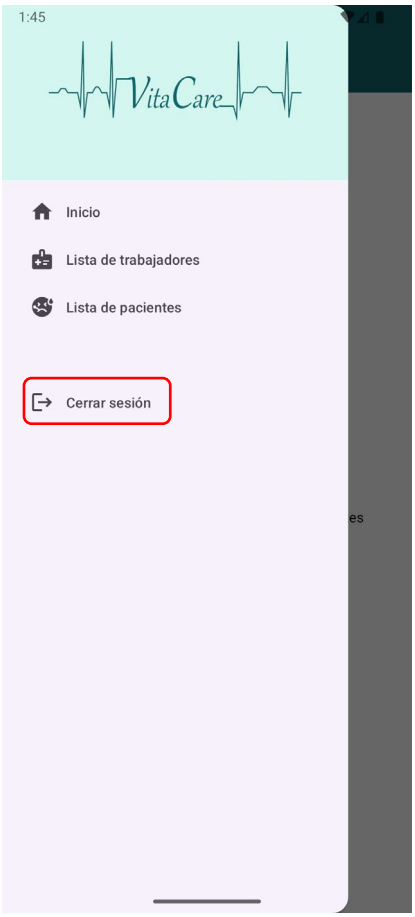


Figura Apéndice A-4. Menú lateral del admin

Al pulsar en la opción de "Cerrar sesión" presente en las figuras Apéndice A-3 y A-4, aparece en pantalla el diálogo de doble confirmación, el cual se debe aceptar para concluir con la acción tal y como se muestra en la figura Apéndice A-5.

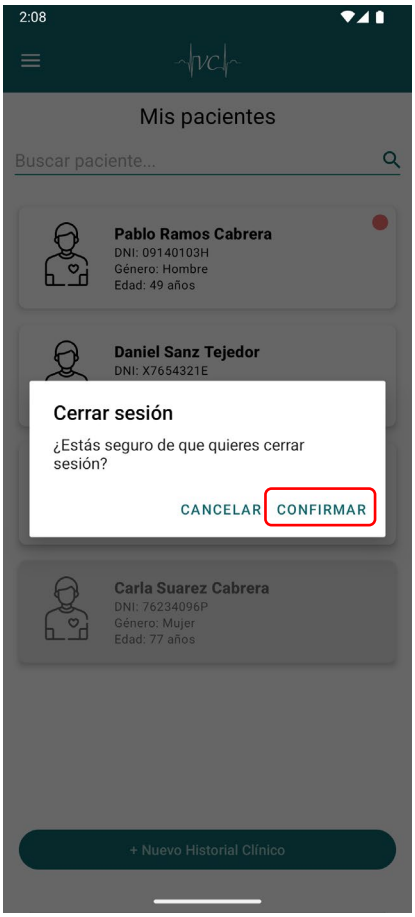


Figura Apéndice A-5. Confirmación para cerrar sesión

2. Gestión del administrador

2.1 Registrar trabajador

El administrador cuando tenga que añadir un nuevo trabajador deberá entrar en la sección de "Lista de trabajadores", como se observa en la figura Apéndice A-6, y pulsar en "Añadir nuevo trabajador", figura Apéndice A-7.

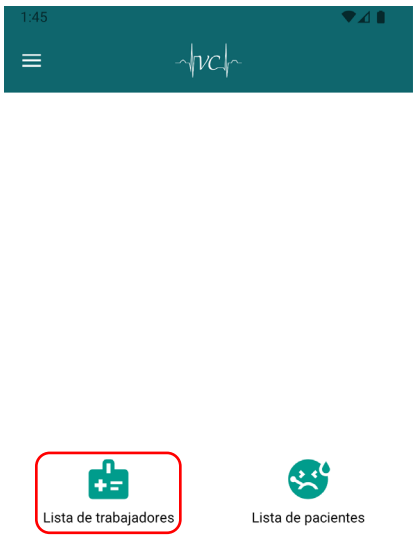


Figura Apéndice A-6. Pantalla principal del admin

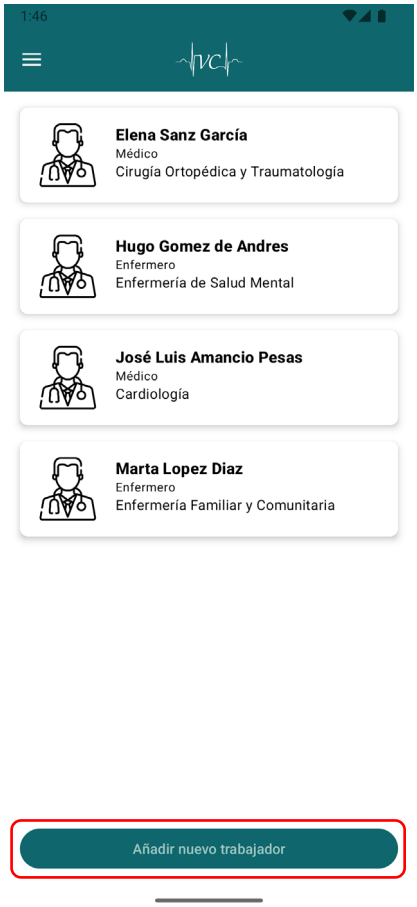


Figura Apéndice A-7. Lista de trabajadores

Al rellenar el formulario de registro, se debe pulsar en el botón de "Añadir trabajador" que aparece en la figura Apéndice A-8. Luego se debe hacer click en "Confirmar", para que, automáticamente, si todos los datos son correctos, se añada al nuevo usuario, tal como se muestra en la figura Apéndice A-9.

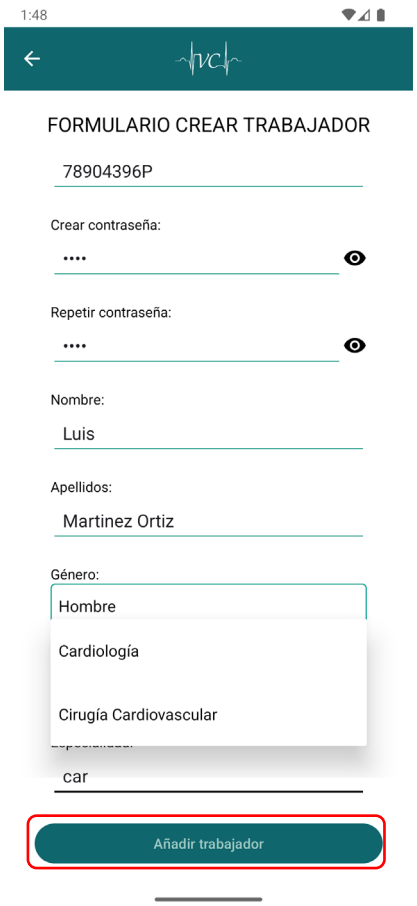


Figura Apéndice A-8. Formulario para añadir trabajador

Figura Apéndice A-9. Confirmación para añadir trabajador

2.2 Modificar trabajador

Al deslizar hacia la derecha sobre un profesional sanitario, dentro de "Lista de trabajadores" como en la figura Apéndice A-10, se muestra la opción de editar sus datos. Al abrir el formulario, cambiar lo necesario y darle a "Modificar trabajador" y "Confirmar", figura Apéndice A-11 y A-12, los datos son actualizados.

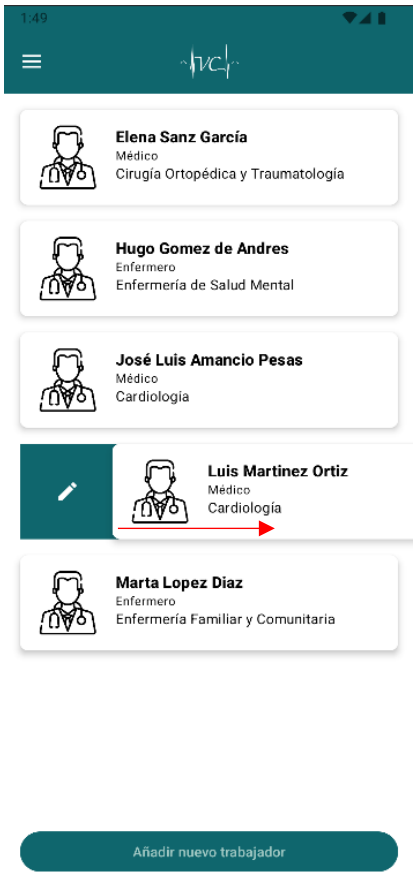


Figura Apéndice A-10. Movimiento de deslizar para editar

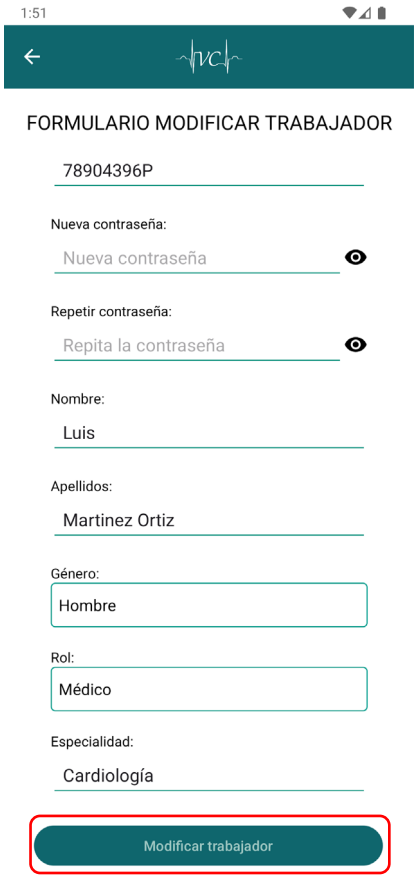


Figura Apéndice A-11. Formulario para editar trabajador

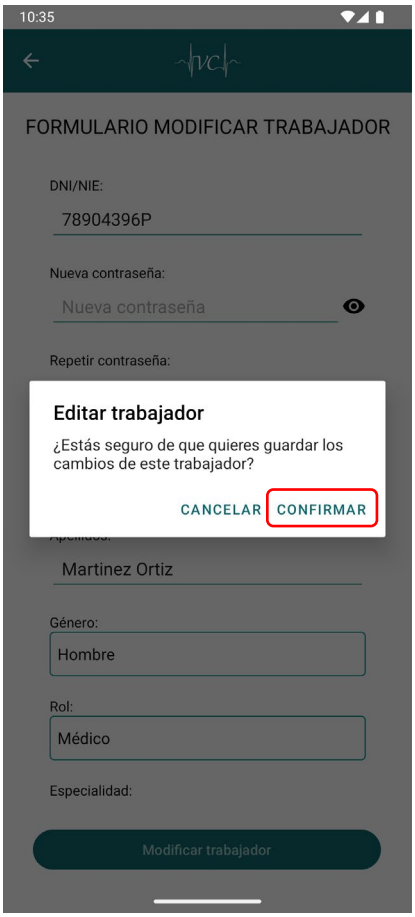


Figura Apéndice A-12. Confirmación para modificar trabajador

2.3 Eliminar usuario

Para eliminar un usuario, ya sea profesional sanitario o paciente, figura Apéndice A-13, se debe acceder a la lista correspondiente y deslizar en este caso hacia la izquierda, como se muestra en la figura Apéndice A-14 y A-15.

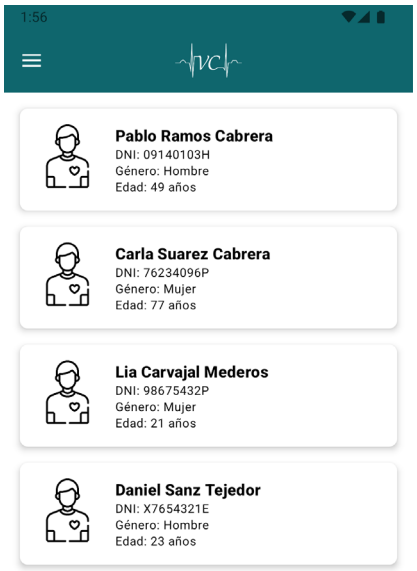


Figura Apéndice A-13. Lista de pacientes

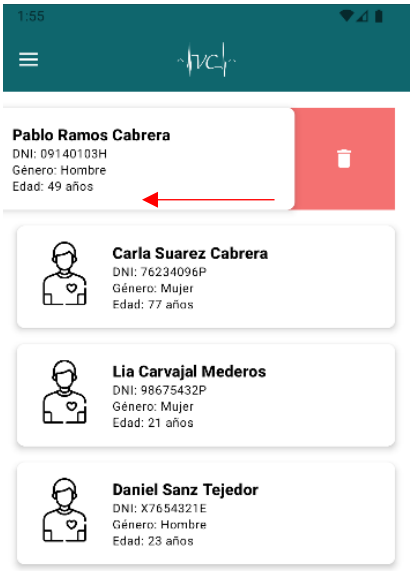


Figura Apéndice A-14. Movimiento para eliminar usuario

Para que el usuario sea eliminado definitivamente, se debe pulsar en "Confirmar", como se muestra en la figura Apéndice A-16.

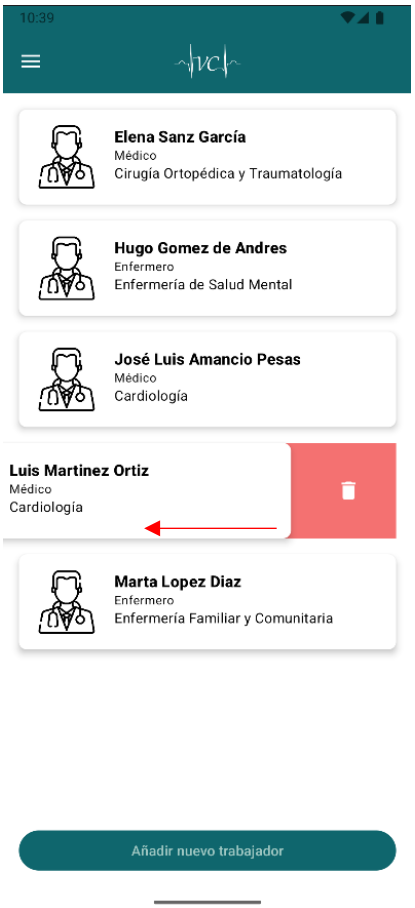


Figura Apéndice A-15. Movimiento para eliminar trabajador

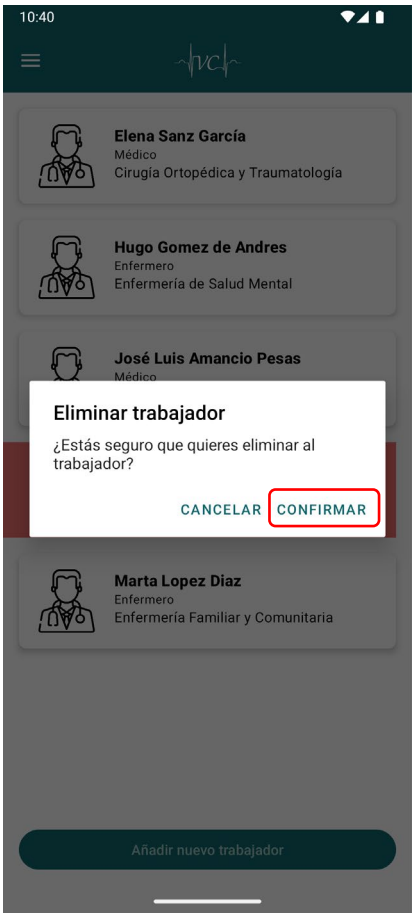


Figura Apéndice A-16. Confirmación para eliminar trabajador

3. Gestión de pacientes

3.1 Buscar paciente

Cuando el médico o el enfermero deseen buscar a un paciente dentro de su lista de asignados, podrá introducir su nombre o alguno de sus apellidos en el buscador, figura Apéndice A-17. A medida que escriban, se irán mostrando automáticamente en pantalla todos los pacientes que coincidan con la información ingresada, figura Apéndice A-18.

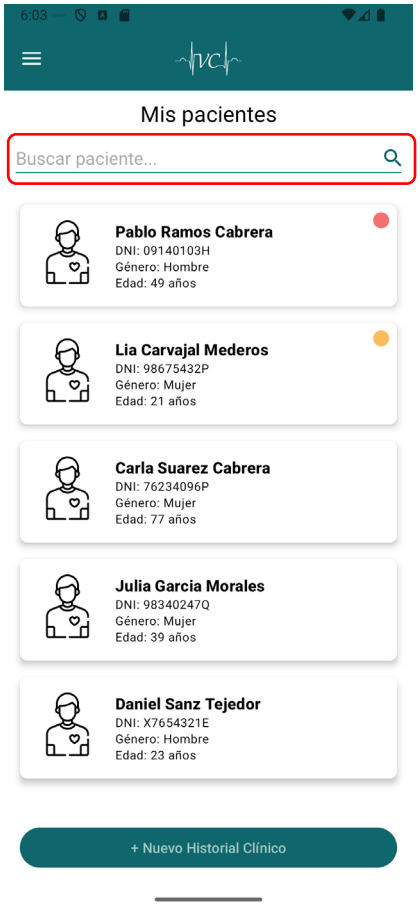


Figura Apéndice A-17. Listado de pacientes

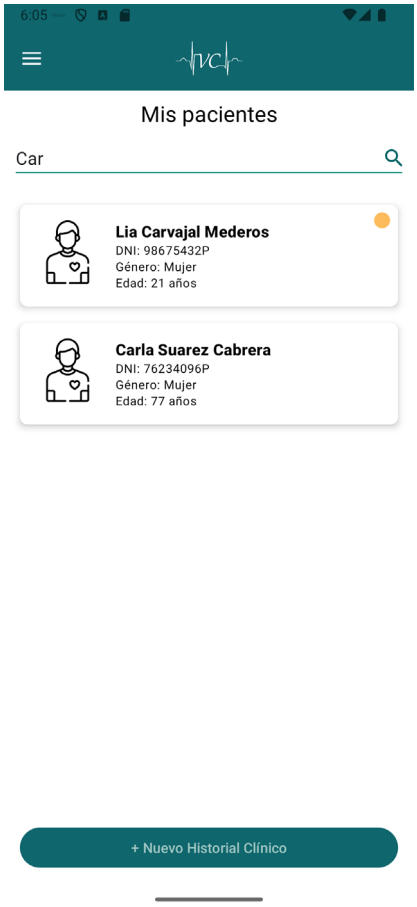


Figura Apéndice A-18. Listado de pacientes actualizado tras búsqueda

3.2 Crear historial clínico del paciente

Al ingresar un nuevo paciente, el médico debe crear un nuevo historial clínico pulsando en el botón "+ Nuevo Historial Clínico", como se observa en la figura Apéndice A-19.

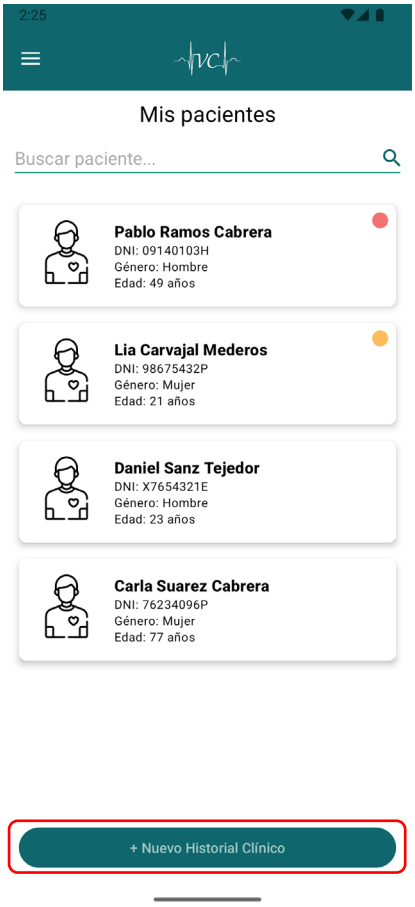


Figura Apéndice A-19. Pantalla principal del médico

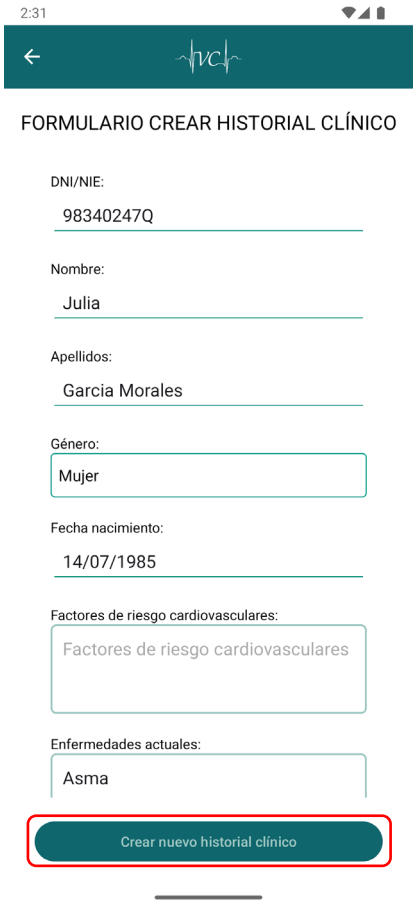


Figura Apéndice A-19. Formulario para crear historial clínico

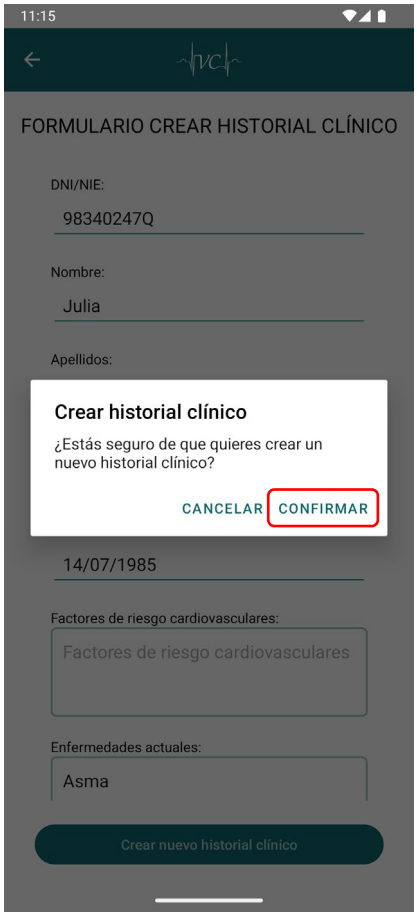


Figura Apéndice A-20. Confirmación para crear un historial clínico

Al completar el formulario con todos los datos necesarios, el médico debe pulsar en "Crear nuevo historial clínico" y luego "Confirmar", como en la figura Apéndice A-19 y A-20, respectivamente.

3.3 Consultar historial clínico del paciente

Cuando el médico o el enfermero necesitan consultar el historial clínico de algún paciente necesitan pulsar en alguno de los pacientes asignados de su lista, por ejemplo, como se señala en la figura Apéndice A-21.

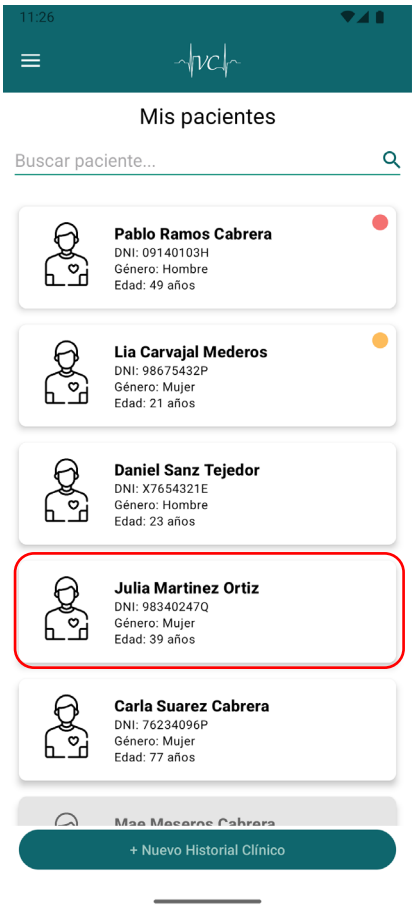


Figura Apéndice A-21. Nuevo paciente en la lista de asignados

Una vez dentro, en la pestaña "Datos personales" se muestra toda la información del nuevo paciente, mientras que en la pestaña "Historial clínico" se presentan sus datos médicos, figura Apéndice A-22 y A-23.

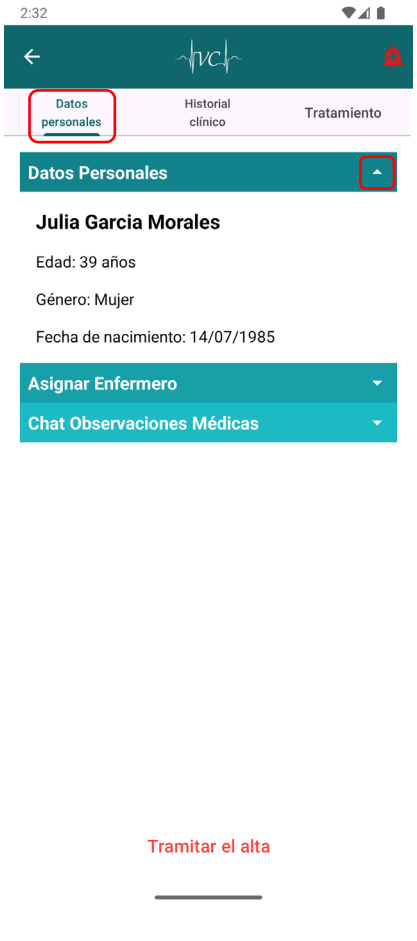


Figura Apéndice A-22. Sección para ver los datos personales

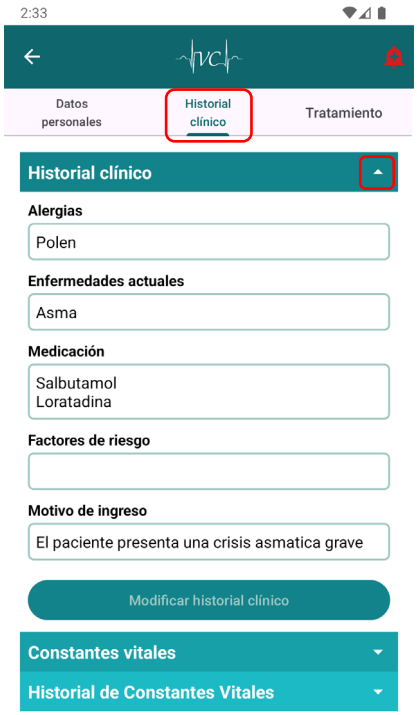


Figura Apéndice A-23. Sección para ver el historial clínico

3.4 Modificar historial clínico del paciente

Cuando el médico necesite modificar el historial clínico debe pulsar en el botón de "Modificar historial clínico", como indica la figura Apéndice A-24.

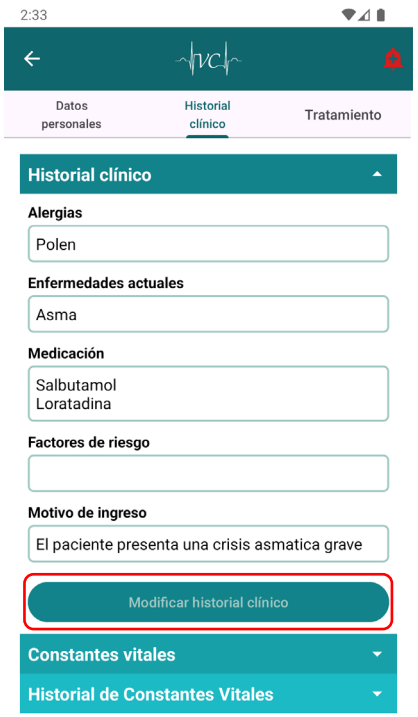


Figura Apéndice A-24. Botón para modificar el historial clínico

Al modificar todos los datos necesarios se pulsa en "Modificar historial clínico" y en "Confirmar", según lo representado en la figura Apéndice A-25 y A-26.

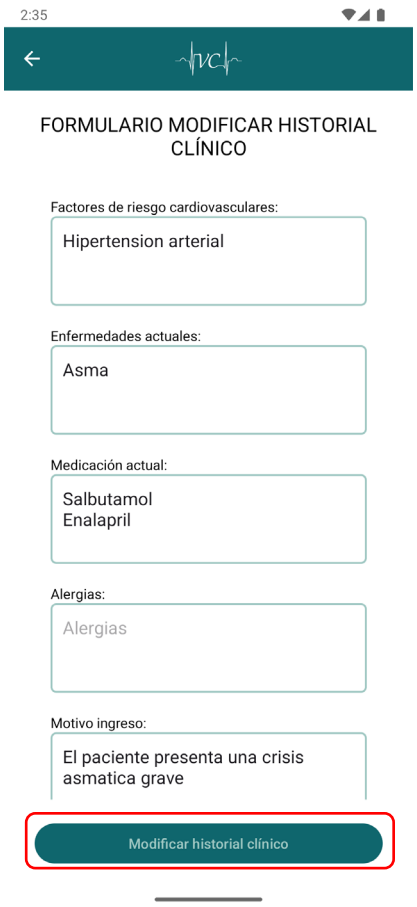


Figura Apéndice A-25. Formulario para modificar el historial clínico

Figura Apéndice A-26. Confirmación para modificar historial clínico

3.5 Añadir/eliminar asignación de paciente a enfermero

Para asignar un enfermero a un paciente, el médico debe acceder a la pestaña "Datos personales" y dirigirse a la sección "Asignar enfermero", como se ve en la figura Apéndice A-27. Allí, debe pulsar el botón "Asignar", seleccionar un enfermero disponible en la lista y, finalmente, hacer clic en "Confirmar", figura Apéndice A-28.

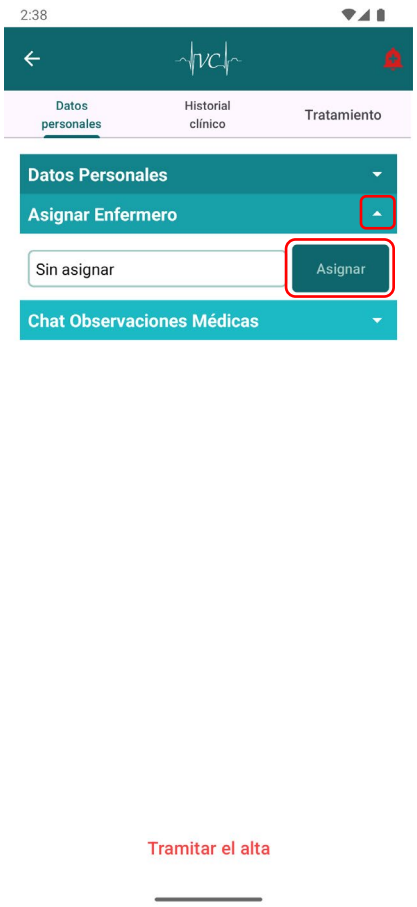


Figura Apéndice A-27. Sección para asignar enfermero

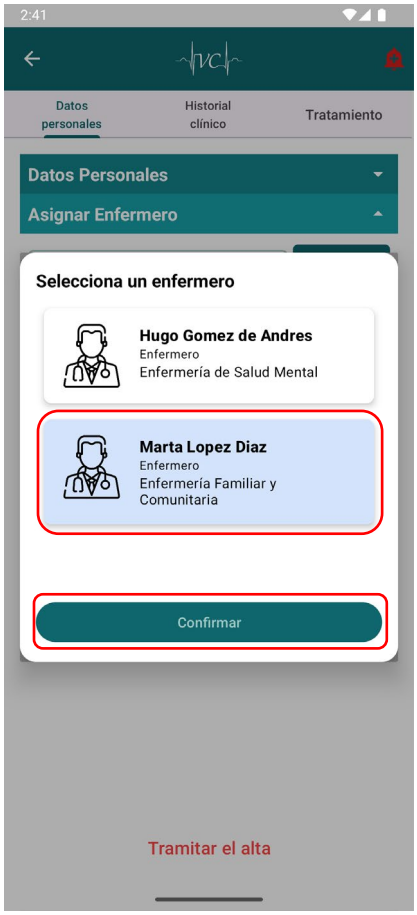


Figura Apéndice A-28. Fragmento para elegir enfermero

Para eliminar, se pulsa el botón "Eliminar", como se indica en la figura Apéndice A-29.

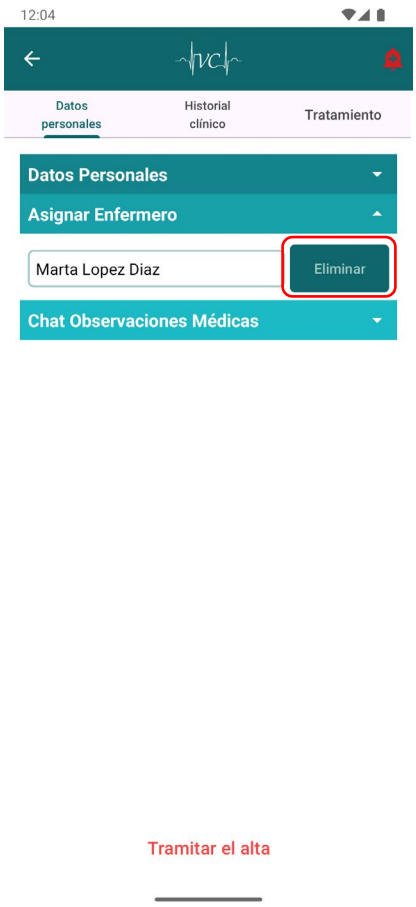


Figura Apéndice A-29. Botón para eliminar enfermero asignado

3.6 Añadir/consultar observaciones médicas del paciente

Para consultar las observaciones médicas, el profesional sanitario debe acceder al paciente y desplazarse hasta la pestaña "Datos personales" y desplegar la sección "Chat observaciones médicas", de la manera que se muestra en la figura Apéndice A-30. Para enviar el mensaje, se debe pulsar en el botón inferior, a la derecha del cuadro de texto, como se muestra en la figura Apéndice A-31. Los mensajes en el margen izquierdo de color grisáceo serán los recibidos y los del margen derecho con color azulado los enviados.

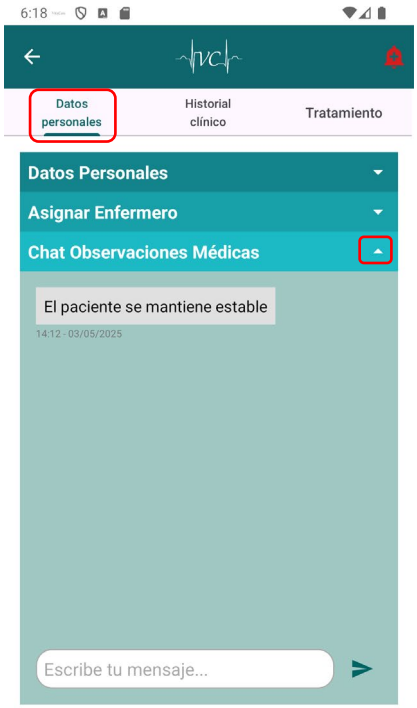


Figura Apéndice A-30. Sección para ver las observaciones médicas

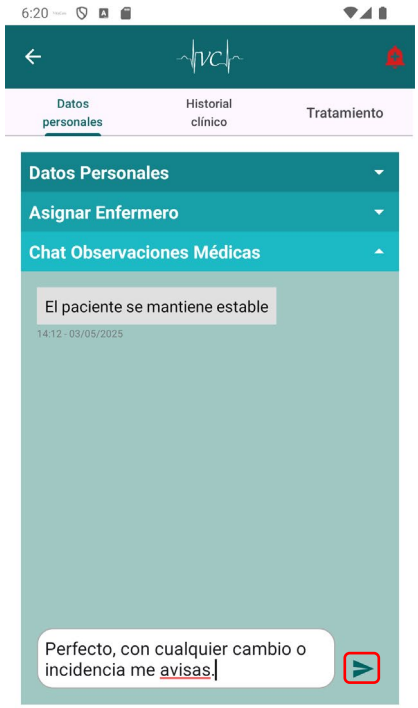


Figura Apéndice A-31. Enviar observación médica

3.7 Añadir/modificar constantes vitales del paciente

Para que el profesional pueda añadir o modificar las constantes vitales ya registradas con anterioridad debe ir hasta la pestaña "Historial clínico" y desplegar la sección "Constantes vitales". Se le muestran los últimos valores registrados o ninguno si es la primera vez. Para registrar los nuevos valores pueden usar los botones de acción de "+" o "-", o modificarlo con la entrada de texto por el teclado, tal como se ilustra en la figura Apéndice A-32.

Figura Apéndice A-32. Sección para añadir constantes vitales

Una vez modificado los valores de las constantes vitales, debe pulsar sobre el botón "Guardar cambios", figura Apéndice A-33, y "Confirmar" para registrar los nuevos valores, figura Apéndice A-34.

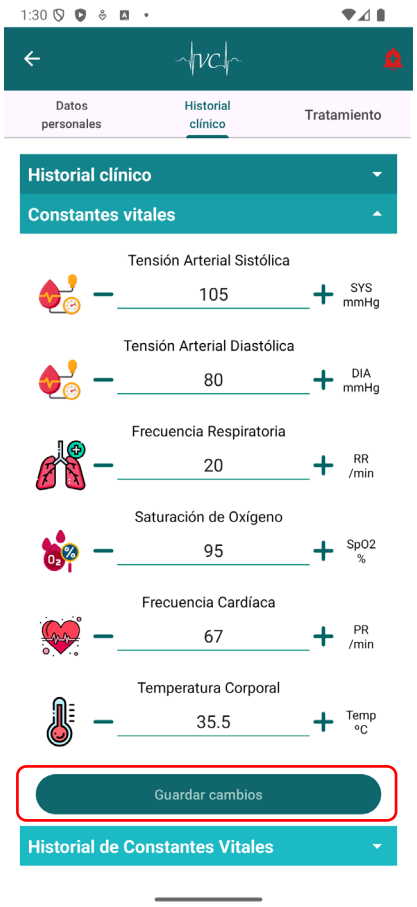


Figura Apéndice A-33. Guardar cambios de constantes vitales

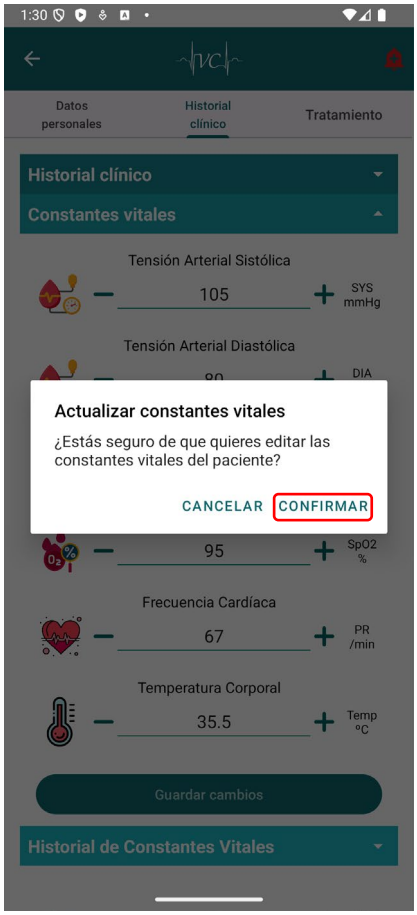


Figura Apéndice A-34. Confirmación para actualizar constantes vitales

3.8 Consultar constantes vitales del paciente

Para que los sanitarios puedan ver las constantes vitales registradas en la base de datos deberá desplazarse hasta la misma sección en la que puede modificarlas. En la sección "Constantes vitales" puede ver los últimos valores que fueron guardados, y desplegando la sección "Historial de constantes vitales" pueden ver en detalle todas las constantes vitales que han sido tomadas al paciente en diferentes momentos durante su ingreso, de acuerdo con lo que se muestra en la figura Apéndice A-35 y A-36.

Figura Apéndice A-35. Sección para consultar últimas constantes vitales tomadas

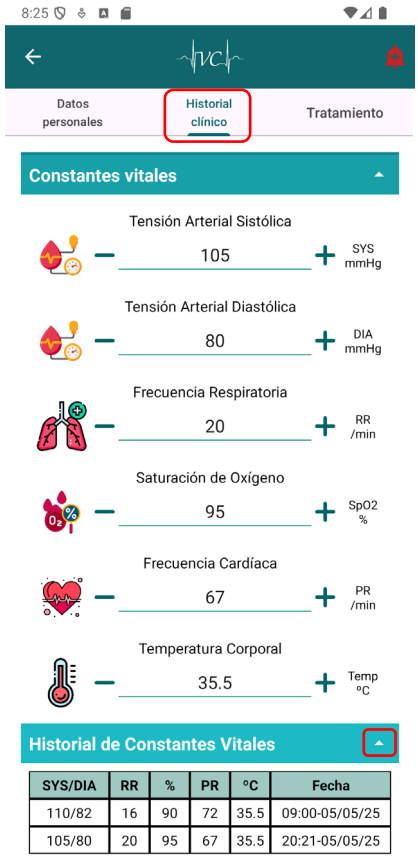


Figura Apéndice A-36. Sección para consultar el historial de todas las constates vitales tomadas

3.9 Crear tratamiento del paciente

El médico puede crear un tratamiento para su paciente. Para ello debe entrar en el paciente interesado, moverse hasta la pestaña "Tratamiento" y pulsar sobre la flecha para que se despliegue la sección "Tratamiento" y pulsar sobre el botón "Crear nuevo tratamiento", como en la figura Apéndice A-37.

Figura Apéndice A-37. Sección para crear un nuevo tratamiento

Una vez haya pulsado, se le redirecciona a una nueva pantalla con un formulario para que rellene los campos. Para finalizar y crear el nuevo tratamiento debe pulsar en "Crear nuevo tratamiento" y luego "Confirmar", figura Apéndice A-38 y A-39.

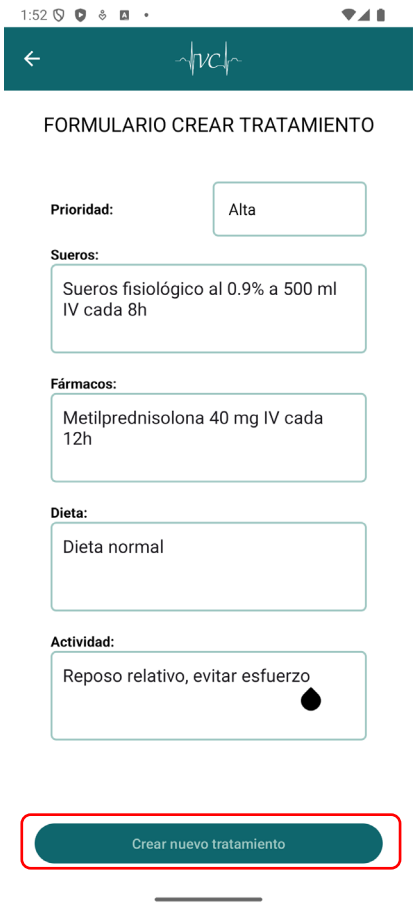


Figura Apéndice A-38. Formulario para crear nuevo tratamiento

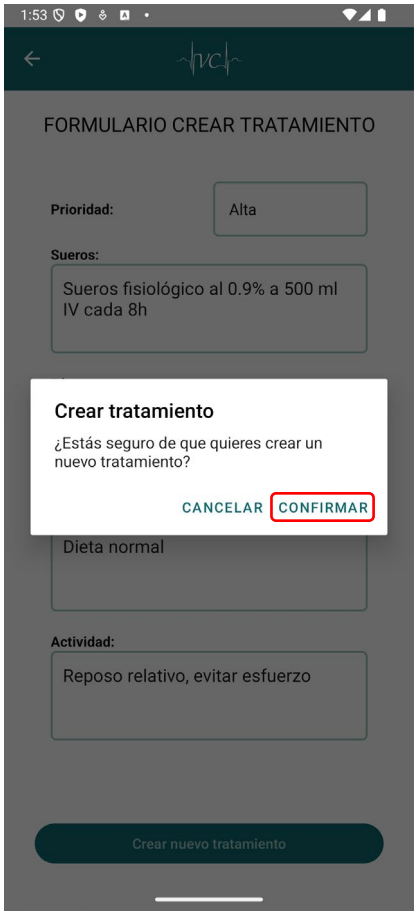


Figura Apéndice A-39. Confirmar para crear nuevo tratamiento.

3.10 Consultar/eliminar tratamiento del paciente

Los sanitarios pueden consultar el tratamiento desplazándose hasta la pestaña del tratamiento y pulsando sobre la sección "Tratamiento", como se muestra en la figura Apéndice A-40. De esta forma pueden visualizar el tratamiento que ha sido pautado para el paciente. Además, desplegando la sección "Historial de tratamiento" pueden ver todos los tratamientos que han sido creados para el paciente, tal como se representa en la figura Apéndice A-41.

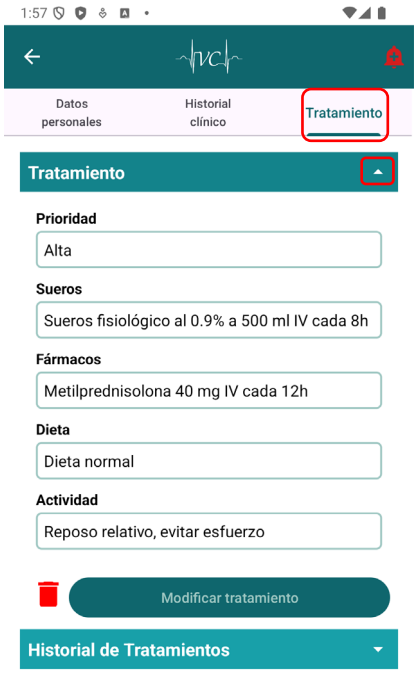


Figura Apéndice A-40. Sección para consultar tratamiento

Figura Apéndice A-41. Sección para consultar historial completo de todos los tratamientos pautados

En caso de que el médico desee eliminar el tratamiento actual, en la misma sección, debe pulsar sobre el botón con forma de papelera y luego confirmar la eliminación pulsando sobre "Confirmar", figura Apéndice A-42 y A-43, respectivamente.

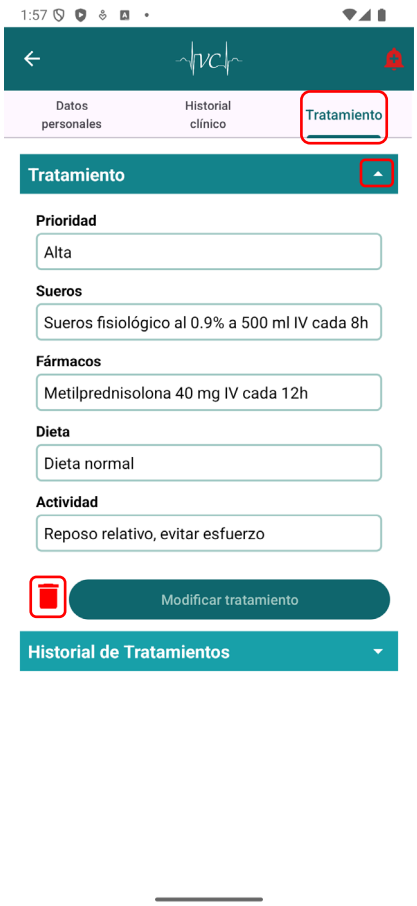


Figura Apéndice A-42. Sección para eliminar el tratamiento

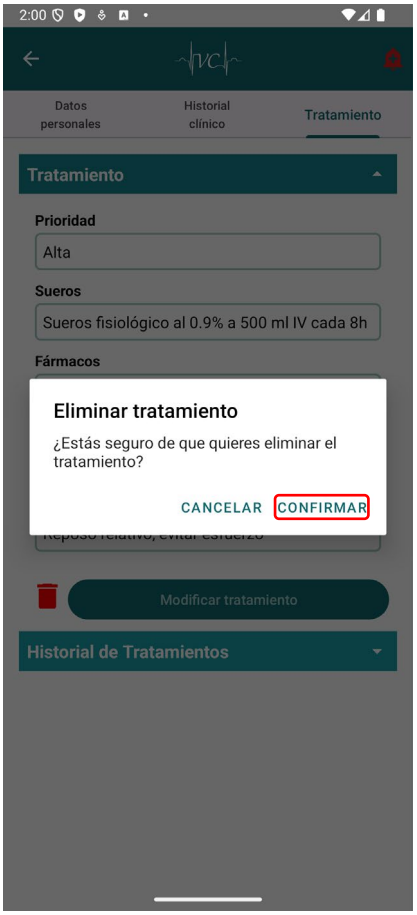


Figura Apéndice A-43. Confirmación para eliminar el tratamiento

3.11 Modificar tratamiento del paciente

El médico además de crear y eliminar el tratamiento, también puede modificarlo. Para hacerlo tiene que pulsar sobre el botón "Modificar tratamiento", ilustrado en la figura Apéndice A-44, y le lleva al formulario para modificarlo.

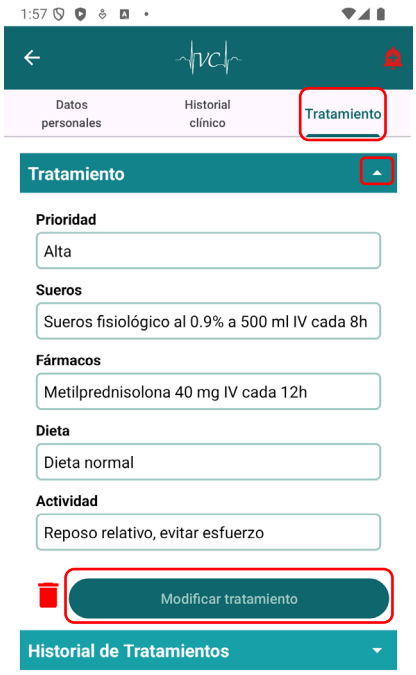


Figura Apéndice A-44. Sección para modificar tratamiento

Una vez dentro del formulario, al médico le aparece precargada la información del tratamiento actual, pudiendo modificar cualquier campo de este. Para guardar el nuevo tratamiento modificado debe pulsar el botón "Modificar tratamiento" y luego "Confirmar", como se representa en la figura Apéndice A-45 y A-46.

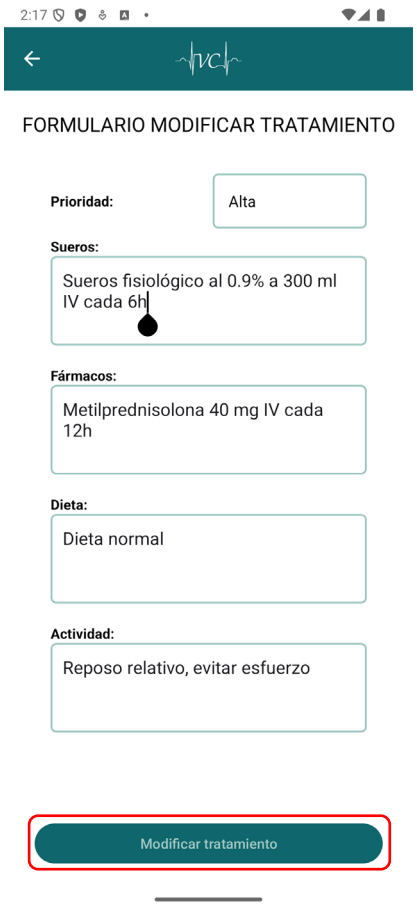


Figura Apéndice A-45. Formulario para modificar tratamiento

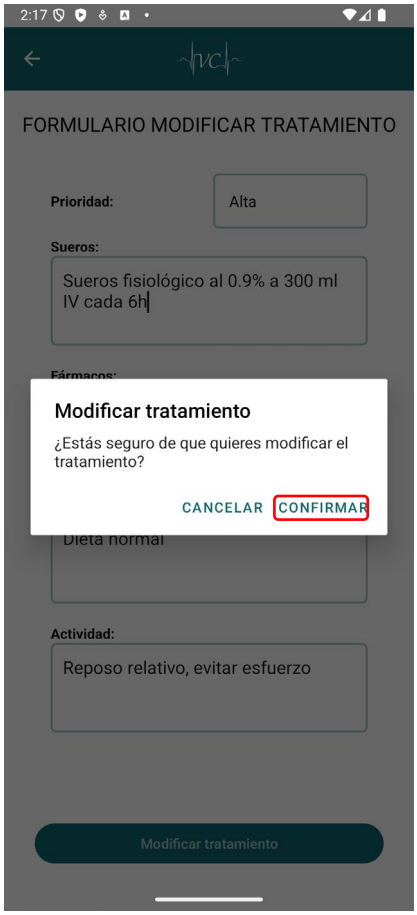


Figura Apéndice A-46. Confirmación para modificar tratamiento

3.12 Generar alerta por el paciente para los profesionales sanitarios

Para generar una alerta sobre un paciente, el médico o el enfermero puede utilizar el botón con forma de campaña, ubicado en la parte superior de la pantalla dentro del toolbar, figura Apéndice A-47. Este botón permanece accesible mientras se navega entre las distintas pestañas. Para confirmar la acción, debe aceptar el diálogo de confirmación que le aparece por pantalla, como en la figura Apéndice A-48.

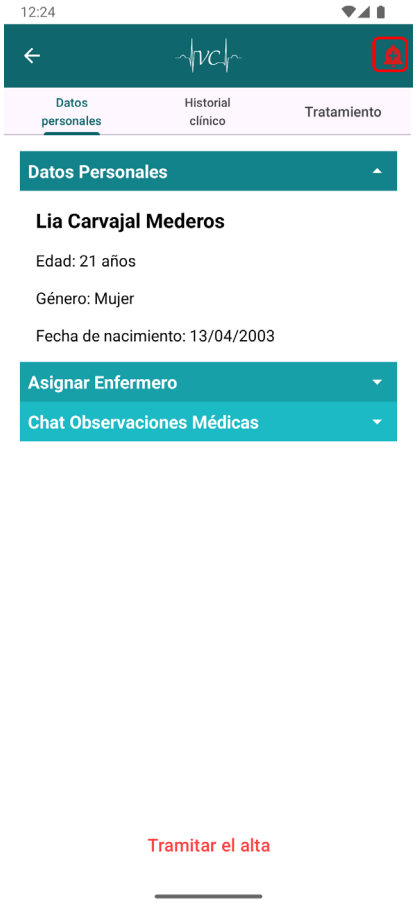


Figura Apéndice A-47. Botón para generar alerta

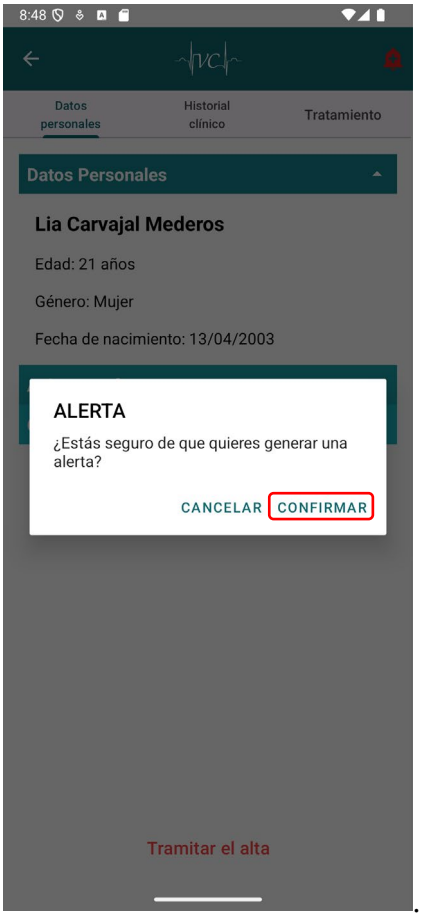


Figura Apéndice A-48. Confirmación para generar alerta

3.13 Tramitar el alta del paciente

Cuando el paciente se haya recuperado y se encuentre en mejor estado sin necesitar los cuidados sanitarios del hospital, el médico puede tramitar el alta de este. Debe entrar en el paciente, desplazarse hasta la pestaña de datos personales y pulsar sobre "Tramitar el alta", figura Apéndice A-49. Después deberá pulsar en "Confirmar" para llevar a cabo la acción, como en la figura Apéndice A-50.

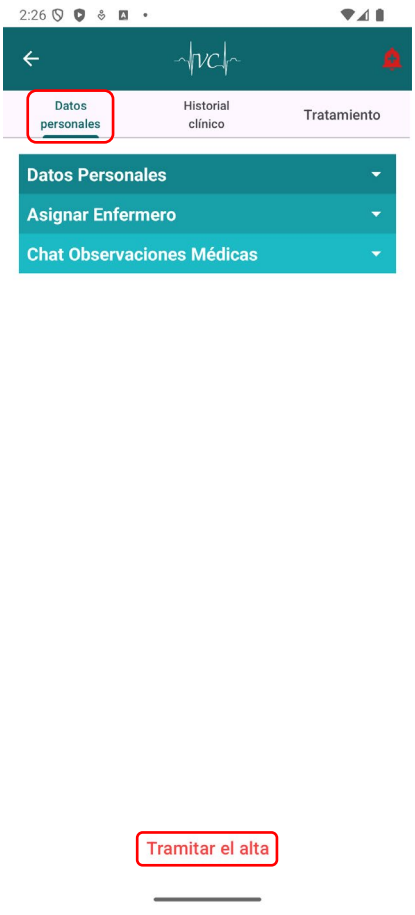


Figura Apéndice A-4949. Sección para tramitar el alta del paciente

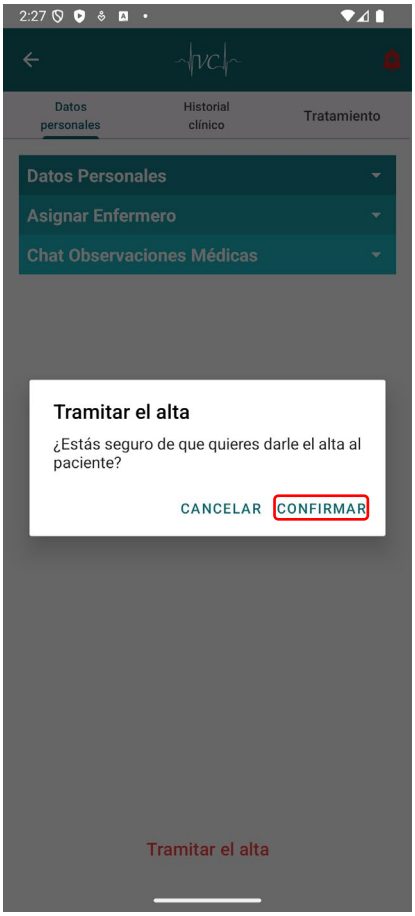


Figura Apéndice A-5050. Confirmación para tramitar el alta del paciente

3.14 Reingresar a paciente dado de alta

El médico puede volver a readmitir a un paciente que ya estuvo ingresado si necesita de nuevo los cuidados sanitarios. Para poder hacerlo debe buscar al paciente entre la lista de sus pacientes y pulsar sobre su cardView, figura Apéndice A-51. Para confirmar el reingreso del paciente deberá pulsar "Confirmar", como en la figura Apéndice A-52.

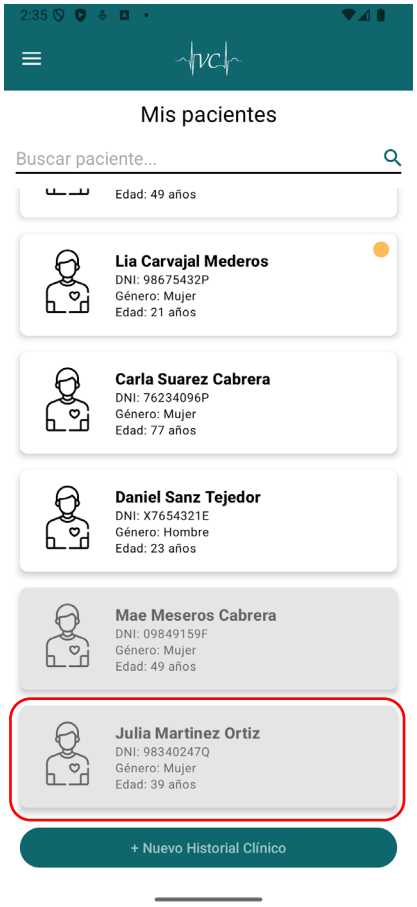


Figura Apéndice A-51. Pantalla para readmitir a paciente

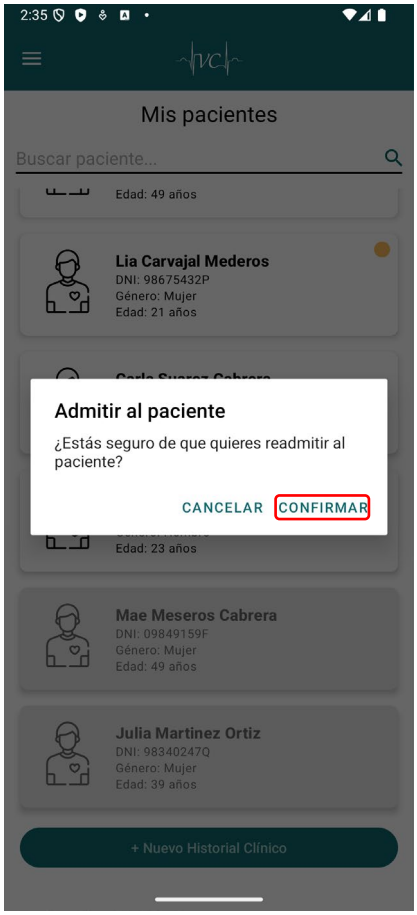


Figura Apéndice A-52. Confirmación para readmitir a paciente

3.15 Leer notificaciones

El personal sanitario puede revisar todas las notificaciones que ha recibido pulsando sobre el botón "Notificaciones" del menú lateral de la aplicación, el cual se representa en la figura Apéndice A-53. Seguidamente, le lleva a la pantalla en donde le aparecen todas las notificaciones entrantes que recibieron.

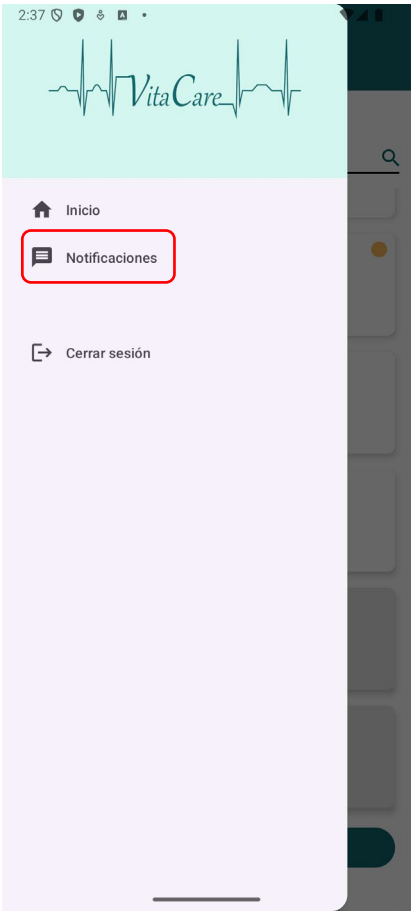


Figura Apéndice A-53. Acceso a notificaciones desde menú lateral

Figura Apéndice A-54. Pantalla de notificaciones recibidas

Además, si desean eliminar las notificaciones que recibieron, pueden pulsar sobre "Limpiar bandeja de entrada" y luego "Confirmar", figura Apéndice A-54 y A-55.

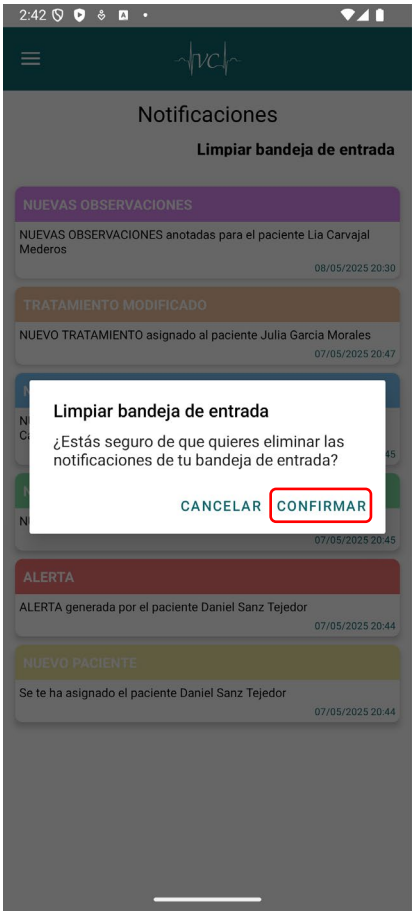


Figura Apéndice A-55. Confirmación para limpiar la bandeja de entrada de notificaciones