UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Alcoy

Aplicación web y servicio RESTful para el seguimiento de pacientes de un Hospital

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Golfe Cazalla, Rubén

Tutor/a: Esparza Peidro, Javier

CURSO ACADÉMICO: 2022/2023

Resumen

Hoy en día, cuando un paciente es dado de alta en un hospital, los sanitarios pierden el control sobre su estado de salud. Muchos servicios de un hospital desearían realizar un seguimiento diario sobre estos sin tenerlo que hacer acudir presencialmente, a no ser que sea realmente necesario.

En este proyecto se viene a dar una posible solución a este problema por medio del desarrollo de una aplicación web y un servicio RESTful.

En este documento se detalla el análisis y el desarrollo que se ha llevado a cabo para conseguir realizar una aplicación capaz de permitir a los sanitarios dar de alta pacientes, crear servicios y asignar servicios a pacientes para poder realizar un control sobre estos. Por otra parte los pacientes podrán registrar distintos tipo de mediciones en función del servicio asignado por cada sanitario. Y se generarán alarmas en caso de que las mediciones registradas por los pacientes sean anormales.

Palabras clave: Web App, RESTful, Hospital, Pacientes, Vue, Node

---

Abstract

Nowadays, when a patient is discharged from a hospital, healthcare professionals lose control over their health status. Many hospital services would like to monitor these patients on a daily basis without requiring them to come in person unless it is absolutely necessary.

This project aims to provide a possible solution to this problem through the development of a web application and a RESTful service.

This document details the analysis and development carried out to create an application that allows healthcare professionals to discharge patients, create services, and assign services to patients for monitoring purposes. Additionally, patients will be able to record different types of measurements based on the service assigned by each healthcare professional. Alarms will be generated in case the measurements recorded by the patients are abnormal.

***Keywords**: Web App, RESTful, Hospital, Patients, Vue, Node*

Índice general

Índice de figuras

Índice de tablas

1 Introducción

El seguimiento que se realiza en los distintos servicios de un hospital, resulta algo clave para la prevención y el tratamiento de enfermedades y condiciones médicas. No obstante, este seguimiento en muchos casos puede llegar a ser difícil o engorroso de llevar por parte tanto de los sanitarios como de los pacientes. Los primeros mencionados pierden demasiado tiempo en realizar pruebas las cuales los pacientes pueden realizar desde casa, esto supone una pérdida de tiempo la cual el sanitario podría dedicar a otras cosas más relevantes. Por otra parte para los segundos mencionados supone también una gran perdida de tiempo en la que se tienen que desplazar y esperar a que los/las atiendan.

Por este motivo, se presenta una posible solución software para hospitales, que necesiten realizar seguimientos a pacientes para el control sobre su salud. Consistirá en una aplicación web y un servicio RESTful que permitirá a los sanitarios gestionar pacientes y realizar seguimientos sobre estos en distintos servicios que puede ofrecer un hospital. El paciente informará sobre los resultados de las pruebas que puede realizar desde casa a través de la aplicación, las cuales supervisará el sanitario responsable. La aplicación también dispondrá de un apartado de alarmas, las cuales se generarán en caso de que algún resultado introducido por el/la paciente sea anormal.

Esta solución pretende ser lo más genérica posible, para que pueda aplicarse de una forma natural a cualquier hospital.

1.1 Motivación

Muchos de los servicios que ofrece un hospital requieren de un seguimiento diario hacia los pacientes, de los cuales no es necesaria la presencia de estos en el hospital, por ejemplo el control de la frecuencia cardíaca para pacientes que sufren de insuficiencia cardíaca, arritmias, y otras muchas enfermedades cardíacas, el control de la glucemia en sangre para pacientes diabéticos o por ejemplo el control del hierro en la sangre para personas con anemia, hemocromatosis, etc. Cada uno de los ejemplos planteados y muchos mas, son pacientes que pueden realizar un control desde casa haciendo uso de dispositivos como monitores de frecuencia cardíaca, glucómetros, kits de medición de hierro en la sangre y muchos otros los cuales se pueden adquirir en cualquier farmacia.

Por otra parte, hoy en día la tecnología forma parte de nuestras vidas, cada vez resulta más difícil encontrar personas que no dispongan de un ordenador o de un dispositivo móvil. Por estos motivos se ha decidido iniciar este trabajo.

1.2 Estudio del mercado

A día de hoy, como se ha comentado anteriormente, la tecnología forma parte de nuestra vida diaria. Cada vez resulta más extraño observar a una persona sin un ordenador personal o sin un dispositivo móvil, por esta razón se ha generado un mercado muy grande tanto de aplicaciones *web*, tanto aplicaciones móviles.

En este apartado se van a detallar tres de las diferentes aplicaciones web, aplicaciones del mercado de Google (Google Play Store) y aplicaciones del mercado de Apple (Apple Store) que se han analizado, destinadas al seguimiento de pacientes y analizaremos algunas de sus ventajas e sus inconvenientes.

1.2.1 SocialDiabetes

Esta aplicación gratuita (SocialDiabetes) lanzada tanto en el mecado de Google (SocialDiabetes Google) tanto en el mercado de Apple (SocialDiabetes Apple), es una herramienta que permite la entrada manual de datos para realizar un seguimiento de la diabetes y poder compartirla con el personal sanitario.

Esta aplicación, al igual que la desarrollada, permite introducir mediciones, compartirlas con tu personal sanitario responsable y generar alarmas en casos de mediciones anómalas, todo ello enfocado para realizar un seguimiento exhaustivo de la diabetes.

Aunque esta herramienta presenta diversos gráficos para poder visualizar de una forma más clara la evolución del paciente, queda limitada al único servicio que ofrece. En el caso de *HospitalServices* el propio sanitario/a evaluará los diferentes servicios que precisa el/la paciente para realizar un seguimiento, y el propio sanitario/a los creará y asignará.

1.2.2 BloodPressure

En esta aplicación gratuita (BloodPressure) lanzada en el mercado de Google (BloodPressure Google) permite realizar mediciones de la presión arterial y exportar los datos para poder compartirlos con los/las sanitarios/as.

Con *BloodPressure* podemos ingresar mediciones sobre la presión arterial, ver distintos gráficos detallados sobre la evolución y permite exportar estos datos en diferentes formatos para poder compartirlos con el personal sanitario.

Al igual que la aplicación anterior, esta presenta la misma limitación de servicios, a pesar de ser una aplicación muy completa a la hora de realizar un seguimiento de la presión arterial también presenta un problema a la hora de compratir datos, ya que es necesario

que el paciente los exporte y los envíe de forma separada. En camio con la aplicación desarrollada, al introducir una medición, se le envía directamente al responsable sanitario.

1.2.3 Care Connect de Medtronic

Care Connect es una solución que ofrece Medtronic que combina una plataforma digital para el seguimiento remoto de los pacientes, con servicios exclusivos.

Esta aplicación web realiza seguimientos de servicios de diabetes tipo 1, y de diferentes enfermedades que producen dolor crónico a través de dispositivos de su propia marca de manera automática, es decir que el paciente solo tiene que realizar la medición, no tiene que ingresar los datos en ninguna plataforma.

Aunque no es una mala solución, y presenta más servicios que las aplicaciones gratuitas, sigue siendo una limitación el número de servicios. Por otra parte, a parte de ser de pago, es necesario material adicional de la propia empresa para realizar mediciones automáticas, por se hace difícil ver este tipo de soluciones en hospitales públicos.

1.2.4 Crítica al estado del arte

Después de haber analizado diversas aplicaciones web, móviles y soluciones ya existentes, a parte de las tres mencionadas anteriormente, se aprecian varias similitudes en los problemas que estas presentan.

En primer lugar, se puede observar que la mayoría de las aplicaciones analizadas suelen presentar un problema a la hora de compartir las mediciones realizadas, ya que es necesaria la exportación y el envío de estas por parte del paciente al responsable sanitario.

Por otra parte, tanto las soluciones gratuitas, como las de pago (en menor medida), presentan una limitación en el número de servicios que puede necesitar el paciente. Es decir, las aplicaciones suelen centrarse en crear un seguimiento al paciente de un único servicio, en vez de centralizar todos estos en una única aplicación.

Se puede observar en muchas de ellas la falta de generación de alarmas automáticas a la hora de registrar mediciones, las cuales son necesarias para que en cuyo caso, el sanitario pueda realizar las acciones pertinentes para evitar poner en riesgo la salud del paciente.

Y por último, las pocas aplicaciones que presentan una variedad amplia de servicios y con generación de alarmas, a parte de que sigue dejando al sanitario fuera de la creación de servicios que el precise, estas son de pago y se realizan únicamente en hospitales privados.

1.2.5 Propuesta

En el apartado anterior se han comentado ciertas funcionalidades que ciertas aplicaciones en la actualidad tienen, pero que precisan de otras. Por lo tanto, en esta sección vamos a recopilar las diferentes funcionalidades que formaran parte de esta primera versión de la aplicación desarrollada, ya que en un futuro se piensan ampliar:

- **Gestión de pacientes.** Algo fundamental que ciertas aplicaciones de seguimiento de pacientes tienen es permitir a los sanitarios tener el control de los pacientes a su cargo, por lo que resulta algo imprescindible en la aplicación.
- **Gestión de servicios.** Esta funcionalidad es la que más se echa en falta en las aplicaciones actuales, la cual permitirá al personal sanitario crear los servicios necesarios para el seguimiento de cada paciente.
- **Registro de mediciones.** Como es evidente, presente en todas las aplicaciones de seguimiento. Estas llegarán de forma automática al personal sanitario al ser registradas, a diferencia de ciertas aplicaciones, que requieren que el/la paciente previamente las exporte y las envíe vía mail, etc.
- **Generación de alarmas.** Otra funcionalidad que no está muy presente en las aplicaciones actuales y necesaria para que el personal sanitario pueda actuar de la forma correspondiente a tiempo.
- **Histórico de pacientes.** También una funcionalidad presente en ciertas aplicaciones, pero a diferencia de las demás este histórico presentará los diferentes servicios activos e inactivos del paciente junto a sus mediciones y las alarmas que generó, en cuyo caso, por cada servicio.

1.3 Objetivos

El objetivo de este proyecto es desarrollar una aplicación web y un servicio RESTful que sea capaz de realizar las siguientes acciones:

- Permitir a los/las sanitarios/as crear los servicios que vea necesarios para realizar un seguimiento al paciente.
- Permitir a los/las sanitarios/as dar de alta y baja a los pacientes tanto en la aplicación como en los diferentes seguimientos de servicios que requiera el paciente.
- Permitir a los/las pacientes registrar las diferentes mediciones realizadas necesarias, en los diferentes servicios asignados, para el correcto seguimiento del mismo.
- Permitir a los/las sanitarios/as poder acceder tanto a los seguimientos activos y no activos del paciente como al histórico de alarmas generadas por estos.
- Generar alarmas en casos de mediciones anómalas registradas por pacientes.

Además de estas funcionalidades, se persigue también como objetivo desarrollar tanto la aplicación web como el servicio RESTful de la manera más genérica posible, para que esta solución se pueda aplicar a cualquier hospital.

1.4 Estructura de la memoria

La estructura de la memoria se compone por el análisis del problema (Capítulo 2) donde haciendo uso de diferentes diagramas y casos de uso, se detallarán las distintas funcionalidades de la aplicación.

En el Capítulo 3 se describe la arquitectura de la aplicación (sección 3.1), las tecnologías que se han utilizado (sección 3.2) y como se ha construido tanto la aplicación web (sección 3.3) como el servicio RESTful (sección 3.4).

En el Capítulo 4 se realiza un pequeño tour por la aplicación construida. Y por último en el Capítulo 5 se describe lo que he conseguido realizando este proyecto y las funcionalidades y mejoras que se han quedado por implementar y que se realizarán en un futuro.

2 Análisis del problema

En la siguiente sección, se realiza un análisis del problema que se viene a resolver haciendo uso de diagramas UML y casos de uso.

Podremos observar que en la aplicación habrán tres tipos de usuario: administrador, sanitario y paciente, y dependiendo de el usuario con el que se está conectado podremos realizar según que acciones. Es decir, que salvo ciertas acciones como la consulta de los datos del usuario y la modificación de la contraseña, un paciente nunca podrá realizar acciones de un sanitario, y un usuario que ha iniciado sesión como sanitario nunca podrá realizar acciones como paciente, véase la figura 2.1 y la figura 2.2. Por otra parte el administrador es un usuario de tipo sanitario, el cual es el único que tiene permisos para poder dar de alta nuevos sanitarios, véase la figura 2.3.

Se le deja tanto al sanitario la capacidad de registrar pacientes como al administrador la de registrar sanitarios, para evitar registros de usuarios malintencionados, los cuales podrían producir una sobrecarga innecesaria de la base de datos. Por lo tanto en el apartado del login no se verá ningún formulario para registrar ni pacientes ni sanitarios.

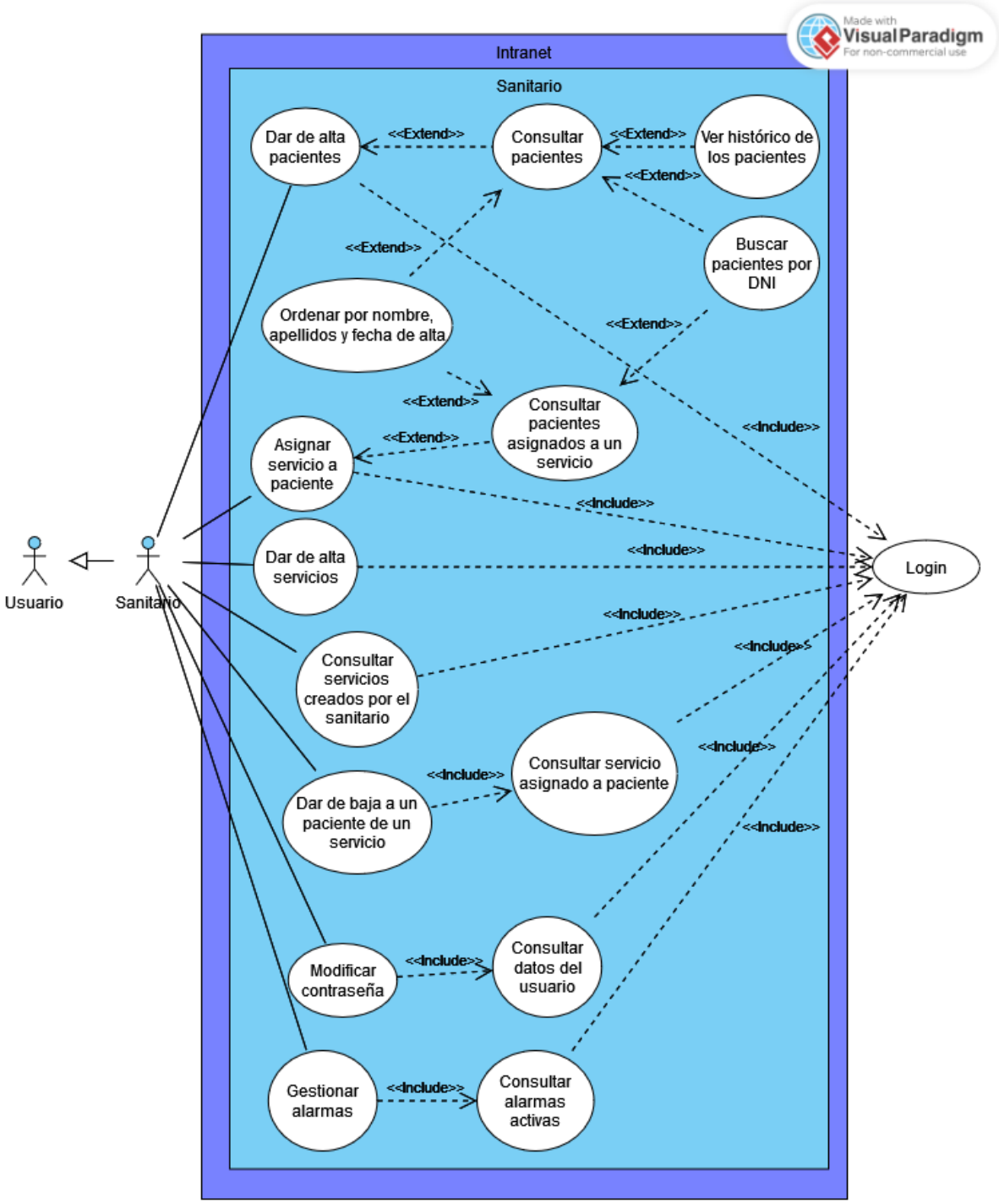

Figura 2.1: Diagrama casos de usos: Sanitario, realizada en VisualParadigm

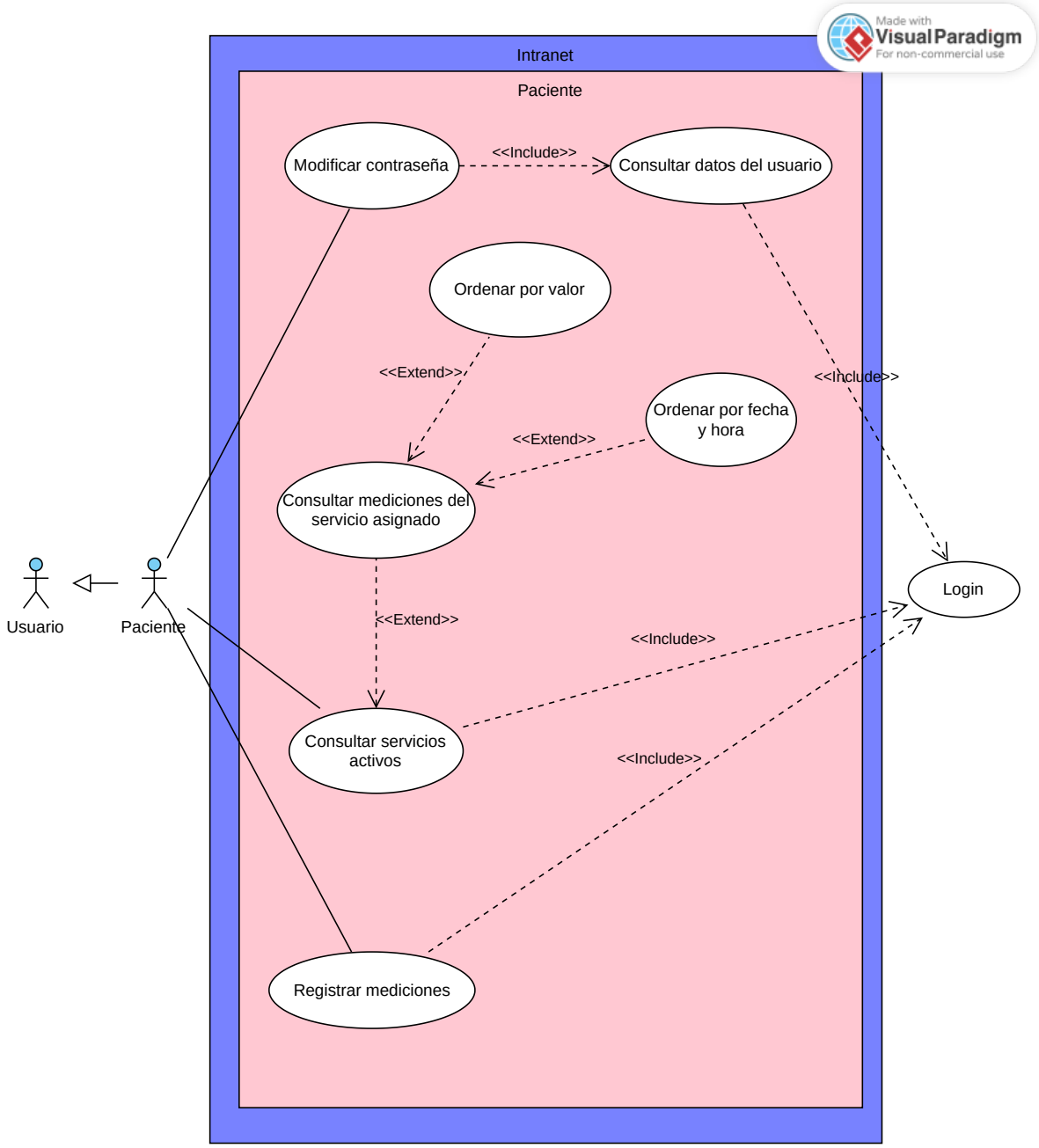

Figura 2.2: Diagrama de casos de usos: Paciente, realizada en VisualParadigm

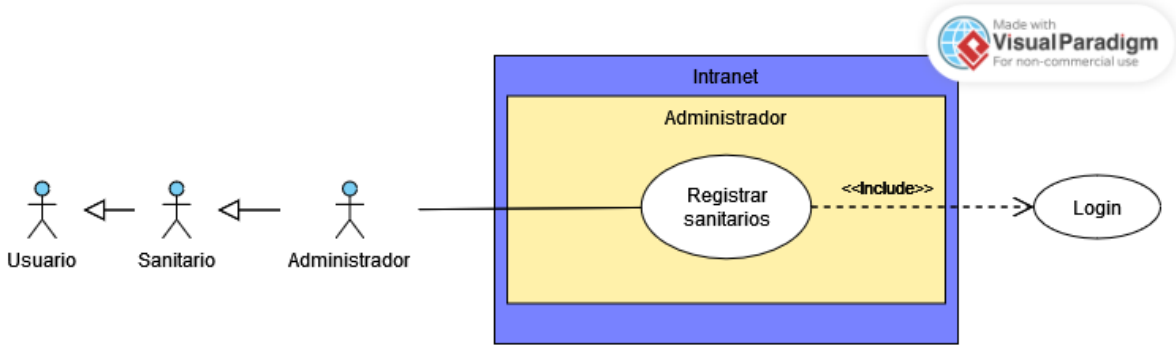

Figura 2.3: Diagrama de casos de usos: Administrador, realizada en VisualParadigm

2.1 Casos de uso

2.1.1 Login

El primer caso de uso consiste en que el sanitario, el paciente o el administrador inicien sesión en la aplicación, esta será la primera ventana que verán al acceder a la web. Estos deberán de introducir su DNI y su contraseña para acceder, si los datos son correctos accederán a la intranet, véase la figura 3.4.

Descripción	Inicia sesión en la aplicación web introduciendo el DNI y la contraseña del usuario.
Actores	Sanitario, administrador y paciente.
Precondición	**Paciente**: Que un sanitario le haya dado de alta en la aplicación. **Sanitario**: Que un administrador le haya dado de alta en la aplicación.
Postcondición	**Paciente**: Entrar a la intranet del paciente. **Sanitario y Administrador**: Entrar a la intranet del sanitario.

Tabla 2.1: Caso de uso: Login

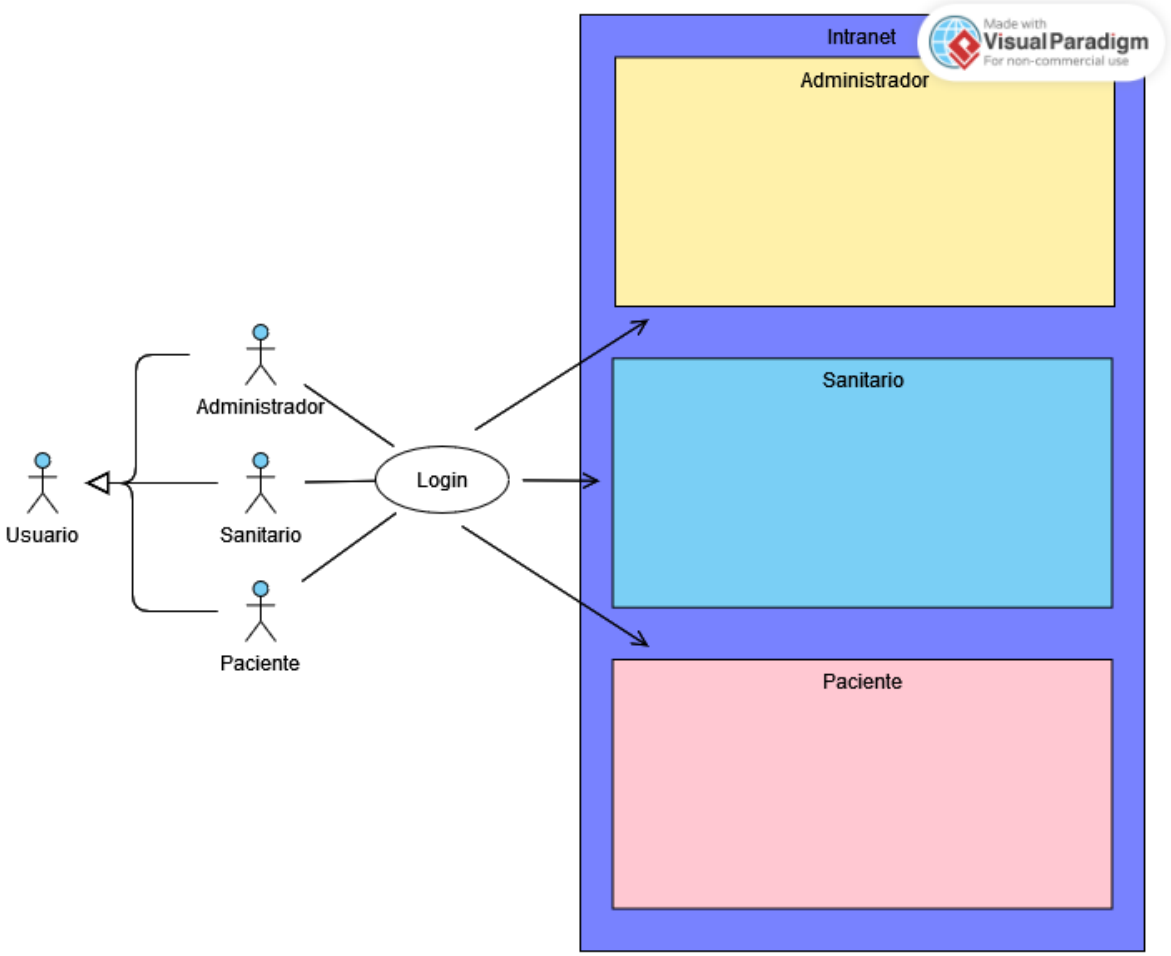

Figura 2.4: Representación de acceso, realizada en VisualParadigm

2.1.2 Consultar datos del usuario

Se mostrará los datos personales del usuario como el dni, nombre y apellidos, desde esta pantalla el usuario también podrá cambiar su contraseña.

Descripción	Muestra los datos personales del usuario y permite modificar la contraseña del mismo.
Actores	Sanitario, administrador y paciente.
Precondición	Que el usuario haya iniciado sesión y accedido al menú de información del usuario.
Postcondición	Cargar los datos personales del usuario que haya iniciado sesión.

Tabla 2.2: Caso de uso: Información del usuario

2.1.3 Modificar contraseña

Accesible desde la consulta de datos del usuario, desde esta pantalla el usuario puede modificar su contraseña.

Descripción	Modifica la contraseña del usuario que ha iniciado sesión.
Actores	Sanitario, administrador y paciente.
Precondición	Que el usuario haya iniciado sesión, accedido al menú de información del usuario, haya introducido una contraseña nueva y pulse sobre el botón cambiar contraseña.
Postcondición	Actualizar la contraseña del usuario.

Tabla 2.3: Caso de uso: Modificar contraseña

2.1.4 Consultar pacientes

Se mostrarán los pacientes de los cuales está a cargo el sanitario. El sanitario podrá realizar una búsqueda de paciente por DNI y podrá ordenar la lista de usuarios por DNI, nombre del paciente, primer apellido del paciente, segundo apellido del paciente y por la fecha de alta del paciente.

Descripción	Pantalla encargada de mostrar un listado de pacientes, en la cual se podrá realizar búsqueda de pacientes por DNI y ordenar el listado por diferentes campos.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador y accedido al menú de pacientes.
Postcondición	Mostrar la lista de pacientes que estén a cargo del sanitario.

Tabla 2.4: Caso de uso: Consultar pacientes

2.1.5 Dar de alta pacientes

Aparecerá un formulario con los campos de DNI, primer apellido, segundo apellido y contraseña para dar de alta un paciente. No se permitirá dar de alta un paciente con un DNI ya registrado en el sistema o si el sanitario se ha dejado algún campo del formulario en blanco.

Descripción	Pantalla encargada de dar de alta un paciente en el sistema.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador, accedido al menú de pacientes, pulsado el botón de registro de pacientes, rellenado los datos y le de al botón de 'dar de alta'.
Postcondición	Registrar un paciente, o mostrar un error en caso de que el sanitario o administrador no haya rellenado el formulario al completo o que el paciente ya esté dado de alta.

Tabla 2.5: Caso de uso: Dar de alta pacientes

2.1.6 Ver histórico de los pacientes

Mostrará los datos personales del paciente y los servicios activos e inactivos del mismo.

Descripción	Pantalla encargada de mostrar el histórico del paciente seleccionado.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador, accedido al menú de pacientes y pulsado sobre un paciente.
Postcondición	Cargar los datos personales y los servicios activos e inactivos del paciente.

Tabla 2.6: Caso de uso: Ver histórico de los pacientes

2.1.7 Consultar servicios creados por el sanitario

Se mostrarán en el menú lateral izquierdo los servicios que haya creado el sanitario o administrador que ha iniciado sesión.

Descripción	Cargar los servicios creados por el sanitario o administrador que haya iniciado sesión.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador.
Postcondición	Se cargan los servicios creados por el sanitario o administrador que haya iniciado sesión en la parte inferior del menú lateral izquierdo.

Tabla 2.7: Caso de uso: Consultar servicios creados por el sanitario

2.1.8 Dar de alta servicios

Dese esta pantalla el sanitario o administrador podrá crear servicios introduciendo en el formulario el nombre del servicio, una breve descripción del mismo y las unidades en las que los pacientes tienen que registrar sus mediciones.

Descripción	Da de alta nuevos servicios en el sistema.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador, accedido al menú de registro de servicios, rellene el formulario y pulse el botón de 'registrar servicio'.
Postcondición	Se registra un servicio en el sistema en caso de que haya introducido todos los campos en el formulario, en otro caso se mostrará un error.

Tabla 2.8: Caso de uso: Dar de alta servicios

2.1.9 Consultar pacientes asignados a un servicio

Se mostrará un listado de pacientes a los que se le ha asignado ese servicio. Esta ventana permitirá realizar una búsqueda de pacientes por DNI, y ordenar el listado de pacientes por nombre, apellidos y la fecha de alta en la que fueron asignados al servicio seleccionado.

Descripción	Muestra el listado de pacientes a los que se le han asignado el servicio seleccionado.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador y haya pulsado en el menú sobre uno de los servicios que ha creado.
Postcondición	Cargar la lista de pacientes activos en ese servicio.

Tabla 2.9: Caso de uso: Consultar pacientes asignados a un servicio

2.1.10 Asignar servicio a paciente

Se mostrará un formulario que permitirá al sanitario asignarle un servicio de hospital al paciente. En este formulario el sanitario o administrador deberá de introducir el dni del paciente, el intervalo de tiempo en el que el usuario deberá de realizar sus mediciones, un valor límite para esas mediciones, las cuales si el paciente sobrepasa se generará una alarma en el sistema, y por último unas indicaciones que debe tener en cuenta el pacientes a la hora de tomar sus mediciones.

Descripción	Asignación de un servicio a un paciente.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador, haya pulsado en el menú sobre uno de los servicios que ha creado, presione el botón de asignación de servicio, rellene el formulario y pulse sobre el botón de 'asignar servicio'.
Postcondición	Se asignará el servicio al paciente siempre que se cumpla que: ▪ El sanitario o administrador haya rellenado todos los campos del formulario. ▪ El paciente al que se le va asignar el servicio ha sido previamente registrado en el sistema. En cualquier otro caso se mostrará un error.

Tabla 2.10: Caso de uso: Asignar servicio a paciente

2.1.11 Consultar servicio asignado a paciente

Se mostrará información tanto del servicio asignado como de las mediciones que ha estado realizando el paciente. Estas mediciones se mostrarán en una tabla, el sanitario o administrador podrá ver el valor de la medición, la fecha y hora en la que fue tomada la medición y si esta medición generó una alarma o no. El sanitario también podrá ordenar las mediciones por cada campo mencionado anteriormente. Por otra parte, desde esta ventana el sanitario podrá dar de baja al paciente del servicio asignado pulsando sobre el botón de dar de baja.

Descripción	Muestra información del servicio asignado, de las mediciones que ha realizado el paciente y permite dar de baja al paciente del servicio asignado seleccionado.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador, haya pulsado en el menú sobre uno de los servicios que ha creado y seleccione a un paciente de la lista.
Postcondición	Se cargará la información del servicio asignado y de las mediciones del paciente.

Tabla 2.11: Caso de uso: Consultar servicio asignado a paciente

2.1.12 Dar de baja a un paciente de un servicio

Se dará de baja un paciente de un servicio que le haya sido asignado previamente en caso de que el sanitario o administrador pulse sobre el botón de dar de baja, en la pantalla de consulta de servicio asignado.

Descripción	Da de baja un paciente de un servicio asignado.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador, haya pulsado en el menú sobre uno de los servicios que ha creado, seleccione a un paciente de la lista y pulse sobre el botón de 'Dar de baja'.
Postcondición	Se da de baja el paciente del servicio asignado.

Tabla 2.12: Caso de uso: Dar de baja a un paciente de un servicio

2.1.13 Ordenar por nombre, apellidos y fecha de alta del paciente

Descripción	Ordena la lista por nombre, apellidos y fecha de alta del paciente de manera ascendente o descendente.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador, entre en la sección de consulta de pacientes responsables del sanitario o en la sección de pacientes asignados a un servicio y pulse sobre el nombre de la columna a ordenar.
Postcondición	Se ordena el listado de manera ascendente en caso de dar un click, y de manera ascendente en caso de dar otro click en el nombre de la columna deseada.

Tabla 2.13: Caso de uso: Dar de baja a un paciente de un servicio

2.1.14 Buscar paciente por dni

Descripción	Busqueda de pacientes por dni.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador, entre en la sección de consulta de pacientes responsables del sanitario o en la sección de pacientes asignados a un servicio y introduzca un dni en el buscador.
Postcondición	Búsqueda del paciente en la lista por cada carácter que introduzca de manera automática.

Tabla 2.14: Caso de uso: Buscar paciente por dni

2.1.15 Consultar alarmas activas

Se mostrará un listado de alarmas activas generadas por los pacientes que están a cargo del sanitario, esta lista contendrá el dni del paciente que ha generado la alarma, el servicio sobre el que se ha generado la alarma y la fecha y hora en la que se generó.

Descripción	Muestra las alarmas activas y permite gestionarlas.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador y haya pulsado en el menú de alarmas.
Postcondición	Se cargarán las alarmas activas de las que es responsable el usuario que haya iniciado sesión.

Tabla 2.15: Caso de uso: Consultar alarmas activas

2.1.16 Gestionar alarmas

Descripción	Da de baja alarmas activas.
Actores	Sanitario y administrador.
Precondición	Que el usuario haya iniciado sesión como sanitario o administrador, haya pulsado en el menú de alarmas, pulse sobre una de ellas y le de al botón de 'gestionar alarma'.
Postcondición	Se da de baja una alarma activa.

Tabla 2.16: Caso de uso: Gestionar alarmas

2.1.17 Consultar servicios activos

Descripción	Se muestran los servicios activos del paciente en la parte inferior del menú lateral.
Actores	Paciente.
Precondición	Que el usuario haya iniciado sesión como paciente.
Postcondición	Cargar los servicios que tenga activos el paciente.

Tabla 2.17: Caso de uso: Consultar servicios activos

2.1.18 Consultar mediciones del servicio asignado

Carga la información necesaria para que el paciente pueda tomar y registrar una medición. Se muestran las indicaciones del sanitario que debe realizar el paciente, la fecha y la hora de cuando tiene que realizar la próxima medición, el histórico de mediciones que el paciente ha realizado en ese servicio y información adicional del servicio asignado como la fecha de alta, la descripción del servicio y el intervalo de tiempo entre mediciones.

Descripción	Muestra la información necesaria para que el paciente pueda realizar una medición y registrarla según las indicaciones del sanitario responsable.
Actores	Paciente.
Precondición	Que el usuario haya iniciado sesión como paciente y haya pulsado sobre un servicio asignado de la lista del menú.
Postcondición	Cargar las mediciones y información del servicio asignado.

Tabla 2.18: Caso de uso: Consultar mediciones del servicio asignado

2.1.19 Registro de mediciones

Será la encargada de registrar mediciones, el paciente verá las unidades en las que se tomarán las mediciones, las cuales indicó el sanitario a la hora de asignarle el servicio al paciente, y el paciente introducirá el valor de la medición.

Descripción	Registro de mediciones de un servicio asignado.
Actores	Paciente.
Precondición	Que el usuario haya iniciado sesión como paciente, haya pulsado sobre un servicio asignado de la lista del menú, presionado sobre el botón de registrar nueva medición, introduzca la medición y pulse sobre 'registrar medición'.
Postcondición	Se registrará una nueva medición en el servicio asignado. En caso de que el paciente no introduzca la medición y intente registrarla, se mostrará un error por pantalla.

Tabla 2.19: Caso de uso: Registro de mediciones

2.1.20 Ordenar por valor, fecha y hora de la medición

Descripción	Ordena la lista de mediciones de manera ascendente o descendente por el valor, fecha y hora de la medición.
Actores	Sanitario, administrador y paciente.
Precondición	Que el usuario haya iniciado sesión, y si ha iniciado sesión como: ▪ **Sanitario o administrador**. Acceder a la información del servicio asignado de un paciente. ▪ **Paciente**. Pulsar sobre uno de los servicios que tenga activos. Y pulsar sobre una de las columnas de la lista de mediciones del paciente.
Postcondición	Se ordena la lista de manera ascendente si se pulsa una vez sobre el nombre de la columna y se ordena de manera descendente en caso de pulsar otra vez.

Tabla 2.20: Caso de uso: Ordenar por valor, fecha y hora de la medición

2.1.21 Registrar sanitarios

Descripción	Registra nuevos sanitarios en el sistema.
Actores	Administrador.
Precondición	Que el usuario haya iniciado sesión como administrador, pulse sobre el botón de 'registro de pacientes' del menú lateral, rellene el formulario y pulse el botón de 'registrar sanitario'.
Postcondición	Da de alta un nuevo sanitario en caso de que el administrador haya rellenado el formulario y el sanitario no exista en el sistema.

Tabla 2.21: Caso de uso: Registrar sanitarios

3 Diseño de la solución

3.1 Arquitectura de la aplicación

Para poder desarrollar este proyecto se hace necesario seguir una arquitectura basada en el modelo cliente-servidor (GeeksForGeeks, 2022) en el que los clientes serán los distintos dispositivos (portátiles, sobremesa, dispositivos móviles, etc) que a través de la aplicación web consuman el servicio RESTful alojado en el servidor, véase la figura 3.1.

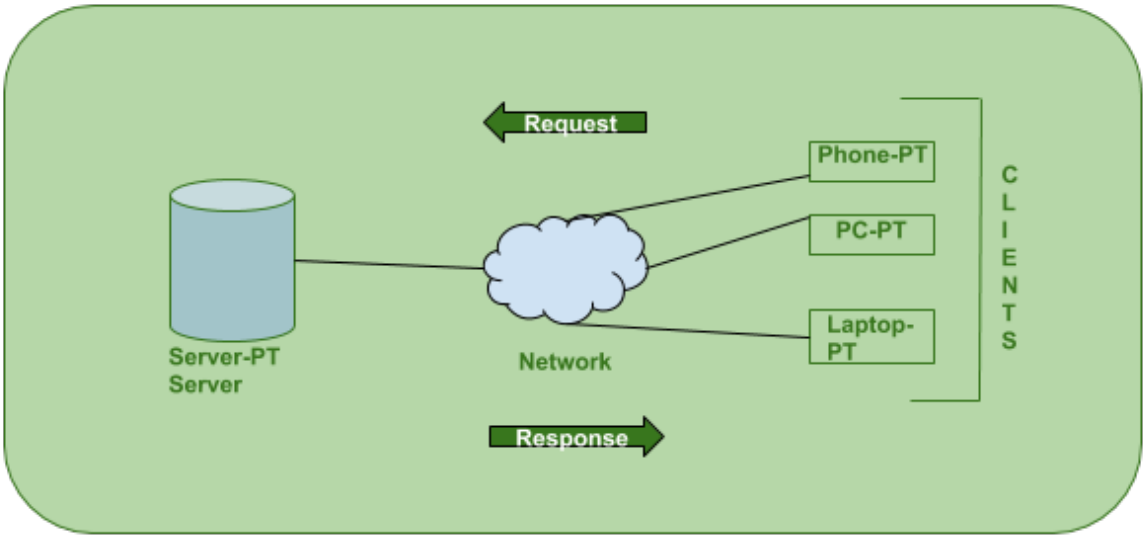

Figura 3.1: Arquitectura Cliente servidor, de Geeks for Geeks

Desde otro nivel de abstracción, se pueden diferenciar varias capas en este proyecto, cada una con un propósito único y una comunicación entre ellas las cuales articularán el proyecto. Estas son la capa de la interfaz gráfica del usuario (GUI), la capa de lógica de negocio, la capa de comunicación y la capa de acceso a datos, véase la figura 3.2.

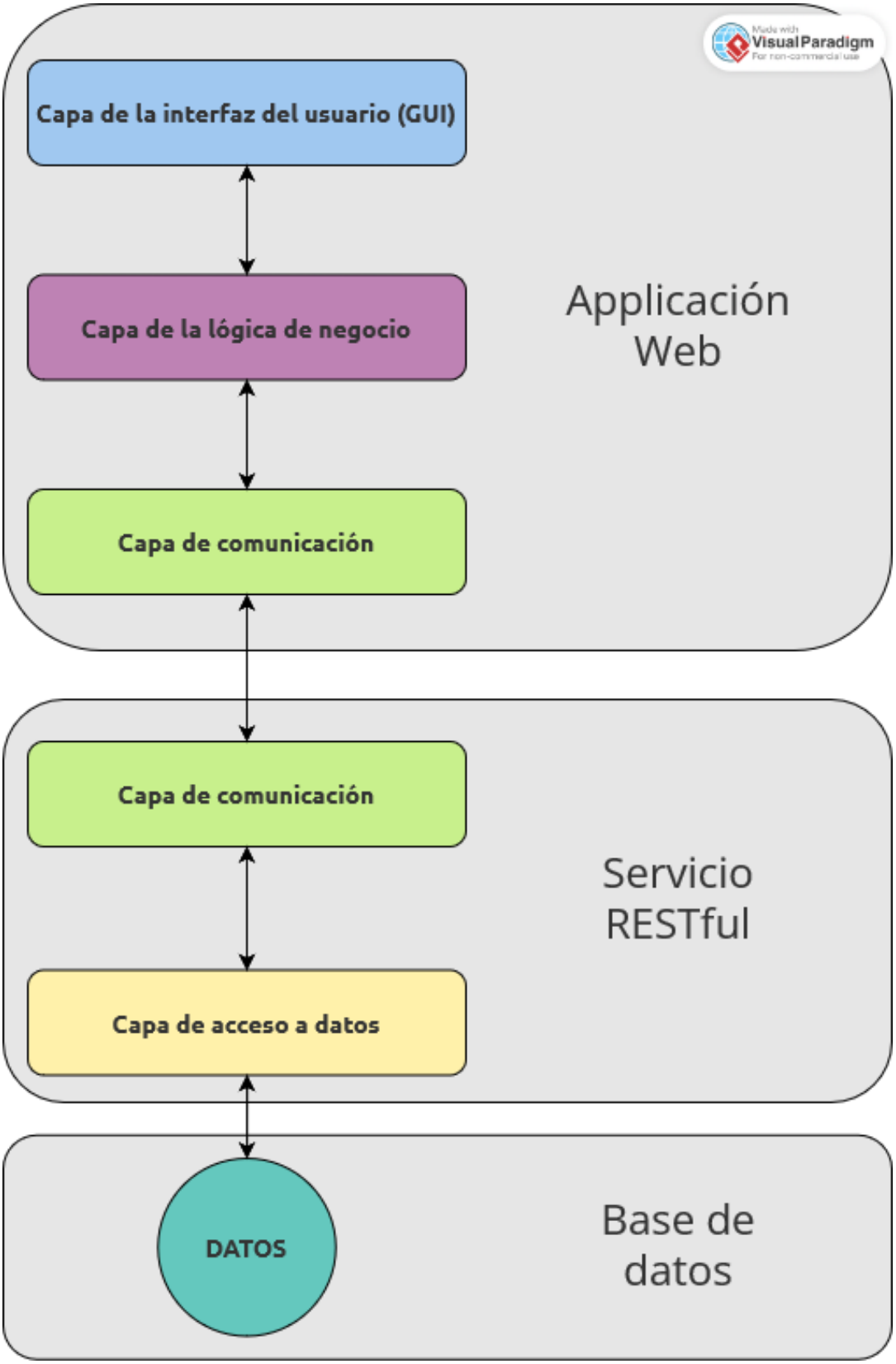

Figura 3.2: Comunicación entre capas, realizada en VisualParadigm

En primer lugar la capa de la interfaz gráfica del usuario es la parte visible y con la que interactúan los usuarios, es decir que esta será la encargada de mostrar la información de manera visual y permitirá la interacción del usuario con los distintos elementos de la aplicación de manera gráfica.

Por otra parte la capa de la lógica de negocio hará de intermediaria entre la GUI y la capa de comunicación, esta será la encargada de la validación y procesado de datos antes de mostrarlos en la GUI o de enviarlos a la capa de comunicación.

La capa de comunicación dentro de la aplicación web, será la encargada de recibir los datos ya procesados de la capa de la lógica de negocio y enviar las peticiones al servicio RESTful. Por otra parte, también interceptará las respuestas del servicio RESTful para posteriormente enviarlas a la capa de la lógica de negocio.

Dentro del servicio RESTful tendremos otra capa de comunicación la cual interceptará las peticiones de la capa de comunicación de la aplicación web para posteriormente pedirle a la capa de acceso a datos lo que necesita.

Por último la capa de acceso a datos, será la que acceda a la base de datos, tanto para obtener, modificar, eliminar o actualizar datos, en función de lo que la capa de comunicación le pida.

3.2 Tecnologías utilizadas

En esta subsección vamos a detallar las diferentes tecnologías que se han utilizado para poder desarrollar tanto la aplicación web como el servicio RESTful, véase la figura 3.3.

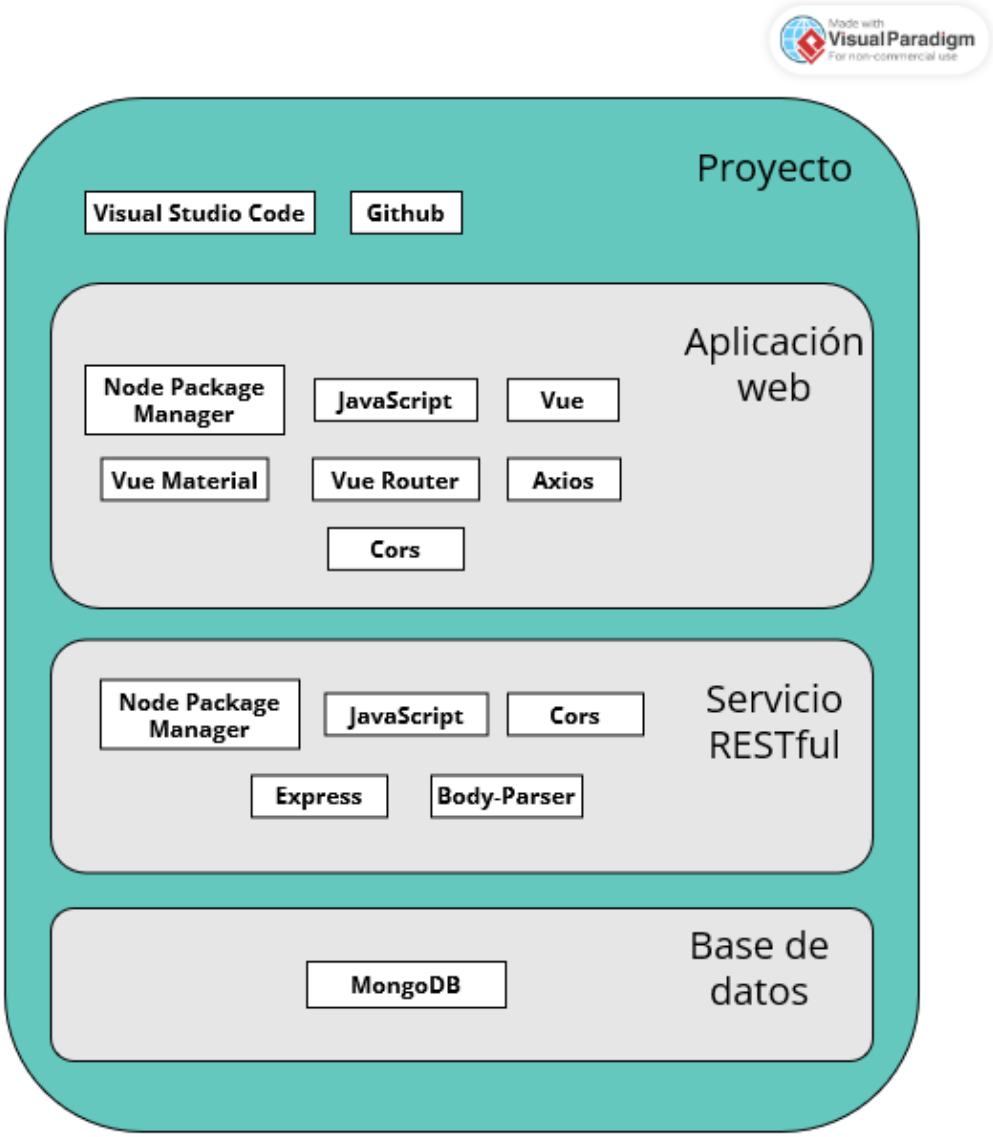

Figura 3.3: Tecnologías utilizadas, realizada en VisualParadigm

3.2.1 Node Package Manager

En primer lugar, a la hora de desarrollar una aplicación web, por simple que esta sea, resulta algo complejo el manejo de librerías de las que esta depende. Por otra parte, estas librerías pueden depender de otras y aún resulta más complejo si dependiendo de la versión de cada librería, necesitamos ciertas dependencias u otras. Para dar solución a este problema utilizamos la tecnología Node Package Manager, más conocido como NPM (Isaac Z. Schlueter, 2010).

Node Package Manager es la herramienta más utilizada encargada de automatizar la gestión de dependencias en proyectos de aplicaciones web (Module Counts), ya que a parte de solucionar el problema anterior, dispone de una inmensa cantidad de paquetes y librerías disponibles para importar directamente en nuestro proyecto.

Por estos motivos, se ha utilizado NPM tanto en la parte del cliente, como en la parte del servidor.

3.2.2 Visual Studio Code

En segundo lugar, se ha utilizado Visual Studio Code como IDE, siguiendo la documentación Visual Studio Code Team, 2023. Este es un entorno de desarrollo integrado ligero el cual permite escribir y depurar nuestro código, por lo que ha resultado necesario en el desarrollo de este proyecto.

3.2.3 Github

Se ha utilizado también Github como controlador de versiones tanto en la aplicación web como en el servicio RESTful. Esta tecnología resulta muy útil para tener un seguimiento de las versiones de este proyecto, para poder compartirlo, etc.

3.2.4 JavaScript

Por otra parte, se ha utilizado JavaScript (Brendan Eich, 1995) tanto en el desarrollo de la aplicación web como en el servidor, ya que a día de hoy se ha convertido prácticamente en un estándar del desarrollo web por diversas razones:

- **Compatibilidad.** Hoy en día JavaScript está presente en todos los navegadores modernos, por lo que se hace difícil encontrar problemas de compatibilidad en cualquier dispositivo.
- **Actualizaciones en tiempo real.** Javascript permite actualizar ciertas partes específicas de una página web sin tener que refrescarla por completo y sin dejar la página bloqueada mientras se realizan peticiones a un servidor.
- **Integración con servicios.** Permite una integración sencilla para el consumo de APIs y servicios.

- **Comunidad.** Al ser el lenguaje más utilizado en el desarrollo web, presenta una comunidad de desarrolladores muy activa y, gracias a esto, un ecosistema que cada vez se hace más grande y robusto de bibliotecas y marcos de trabajo.

3.2.5 Vue

Existen diferentes frameworks webs muy potentes a día de hoy que nos facilitan la construcción de las interfaces del usuario, como por ejemplo Angular (AngularJS Team, 2010), React (Jordan Walke, 2011) o Vue.js (Vue.js Team, 2017). En este proyecto se ha utilizado el framework de Vue.js por diferentes motivos:

- **Tamaño ligero.** Una de las ventajas que presenta Vue.js frente a otros frameworks es su ligereza, esto permite que las aplicaciones desarrolladas se carguen más rápidamente, lo cual hace que el usuario tenga una mejor experiencia.
- **Escalabilidad.** Por otra parte Vue.js ofrece diversas herramientas que facilitan la escalabilidad de la aplicación si en un futuro este proyecto crece mucho.
- **Facilidad de aprendizaje.** Otro punto fuerte de esta tecnología es su suave curva de aprendizaje, esto puede permitir que más gente que vea este proyecto, que tenga nociones básicas de desarrollo, pueda entenderlo y en caso de que quiera, mejorarlo.

3.2.6 Vue Material

Junto a Vue.js se ha utilizado Vue Material (Marcos Moura, 2015) para facilitarnos el desarrollo de las interfaces de usuario. Vue Material es una api que facilita la integración de Material Design (Google Team, 2014) en proyectos Vue.js.

Por otra parte, Material Design es un sistema de diseño 'open source' construído por el equipo de diseñadores y desarrolladores de Google, el cual a parte de ser utilizado en sus proyectos web como Gmail, Youtube, Google Maps o Google Drive, también lo podemos encontrar en páginas web populares como WhatsApp Web o Pinterest.

Lo que destaca de Material Design es que mantiene un diseño limpio y claro, y al ser este bastante utilizado en páginas web populares hace que al usuario le resulte familiar la estructura y el diseño de la misma. Por estas razones se ha escogido utilizar estas tecnologías a la hora de contruír las interfaces de la aplicación web.

3.2.7 Vue Router

Para construir la aplicación web se ha seguido una arquitectura Single page Application (SPA), es decir con un único documento HTML (HyperText Markup Language), ya que esta presenta ciertas ventajas frente las arquitecturas Multi page application (MPA), la cual presenta más de un documento HTML, y nos resultan más beneficiosas que sus desvantajas (Shahin, 2017), por ejemplo:

- **Rapidez.** Las SPA reducen la cantidad de datos transferidos ya que sólo se envían exclusivamente los datos necesarios, esto resulta en un menor consumo de ancho de banda, y por ende unos tiempos de respuesta más rápidas.
- **Mejor experiencia del usuario.** Al tener un único documento HTML y múltiples vistas, hace que la navegación por parte del usuario sea más fluída.
- **Reusabilidad.** Al disponer de un front-end y back-end desacoplados, se puede reutilizar el back-end para otras plataformas.

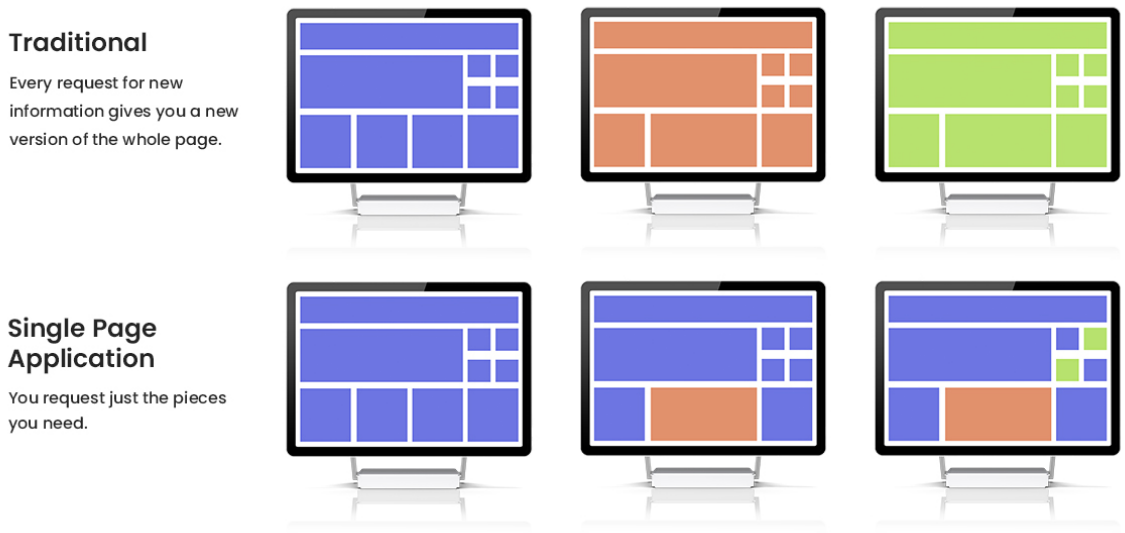

Figura 3.4: Arquitectura Cliente servidor, de Excellent WebWorld

Al seguir una arquitectura SPA, con un único documento HTML y varias vistas, las cuales serán incluidas todas ellas en el HTML, y cada una de estas representarán un componente Vue, resulta necesario mantener un flujo de navegación entre estos componentes, y aquí es donde entra el uso de Vue Router (Eduardo San Martin Morote, 2014).

Vue Router es una librería oficial de Vue.js, la cual nos permite definir un conjunto de rutas y asignarlas a los distintos componentes (o vistas) Vue, lo cual hace que al cambiar de ruta (o url), Vue router detecte estos cambios de forma automática y cambie de componente.

De esta forma, esta librería nos permite tener una navegación fluída y dinámica en una SPA, necesarias para obtener una buena gestión del enrutamiento de nuestra aplicación.

3.2.8 Axios

Por otra parte, al tener una aplicación cliente la cual accede a un servidor mediante el protocolo HTTP (Hypertext Transfer Protocol), necesitamos una tecnología que nos facilite la contrucción y el envío de peticiones HTTP a nuestro servidor, y por otra parte que intercepte y transforme las respuestas del servidor. Para solucionar este problema, se hace uso de Axios (Matt Zabriskie, 2014).

Axios es un cliente desarrollado para Node.js, es decir, disponible en Node Package Manager, que permite construír y envíar peticiones usando el protocolo HTTP, y también es capaz de interceptar estas respuestas por parte del servidor.

3.2.9 Express

En la parte del servidor necesitamos una tecnología que sea capaz de interceptar las peticiones enviadas desde el cliente para poder gestionarlas y devolver una respuesta de vuelta al cliente. Para ello, se utiliza Express (OpenJS Foundation, 2017).

Express es un framework de Node.js flexible y minimalista, que nos ofrece un enrutador que permite definir rutas y manejar las peticiones HTTP del cliente. Este framework es muy utilizado para construir servicios RESTful debido a su enfoque simple y su capacidad de manejar solicitudes HTTP, por este motivo el uso del mismo.

3.2.10 Body-Parser

En la parte del servidor, también se ha utilizado el middleware Body-Parser (Douglas Wilson, 2014). Este middleware de Node.js es utilizado para poder analizar los cuerpos de las solicitudes HTTP: POST, PUT y DELETE.

Esta tecnología nos va a resultar de utilidad ya que al realizar las peticiones HTTP mencionadas anteriormente, se envía información en formato JSON (JavaScript Object Notation), elegido ya que se ha convertido prácticamente en un estándar en el envío de información entre cliente y servidor por su simplicidad, legibilidad, ligereza, flexibilidad, etc. Y esta información resulta imprescindible para poder saber que datos queremos insertar, actualizar o eliminar en la base de datos.

3.2.11 Cors

Al ejecutar desde un mismo ordenador el cliente y el servidor en dos dominios distintos (servidor: http://localhost:8081 y cliente: http://localhost:8082), se está incumpliendo la restricción de seguridad same-origin policy. Esta norma, que aplican todos los navegadores modernos, viene a evitar que múltiples aplicaciones web ataquen a un mismo dominio, es decir que una aplicación web solamante puede realizar peticiones HTTP asíncronas al mismo dominio del que procede. Esta restricción de seguridad fue creada para proteger al usuario de posible ataques.

Por este motivo, se utiliza la librería CORS disponible en NPM (Douglas Wilson y Troy Goode, 2013), este mecanismo, que implementan navegador y servidor, basado en cabeceras HTTP que nos permite solicitar recursos desde un dominio diferente al dominio de origen.

3.2.12 MongoDB

Hoy en día existen multitud de sistemas de bases de datos tanto realcionales como no relacionales o NoSQL, en el caso de este proyecto se ha escogido una base de datos NoSQL debido a su flexibilidad, escalabilidad horizontal, alto rendimiento y velocidad, etc.

Al buscar sistemas de datos no relacionales, se hace inevitable pensar en MongoDB (Dwight Merriman y Eliot Horowitz, 2009), este sistema de base de datos 'open source' es un repositorio basado en documentos muy utilizado hoy en día, que a parte de suscribir las características mencionadas anteriormente, presenta unos accesos a base de datos muy potentes, a parte de una comunidad muy grande y activa.

3.3 Aplicación web

Se ha creado el proyecto a partir de la plantilla Webpack template, haciendo uso del comando:

```
1 vue init webpack nombreDelProyecto
```

Esto se hace porque esta plantilla viene con una estructura de paquetes ya predefinida y que se hace muy fácil de comprender, véase la figura 3.5.

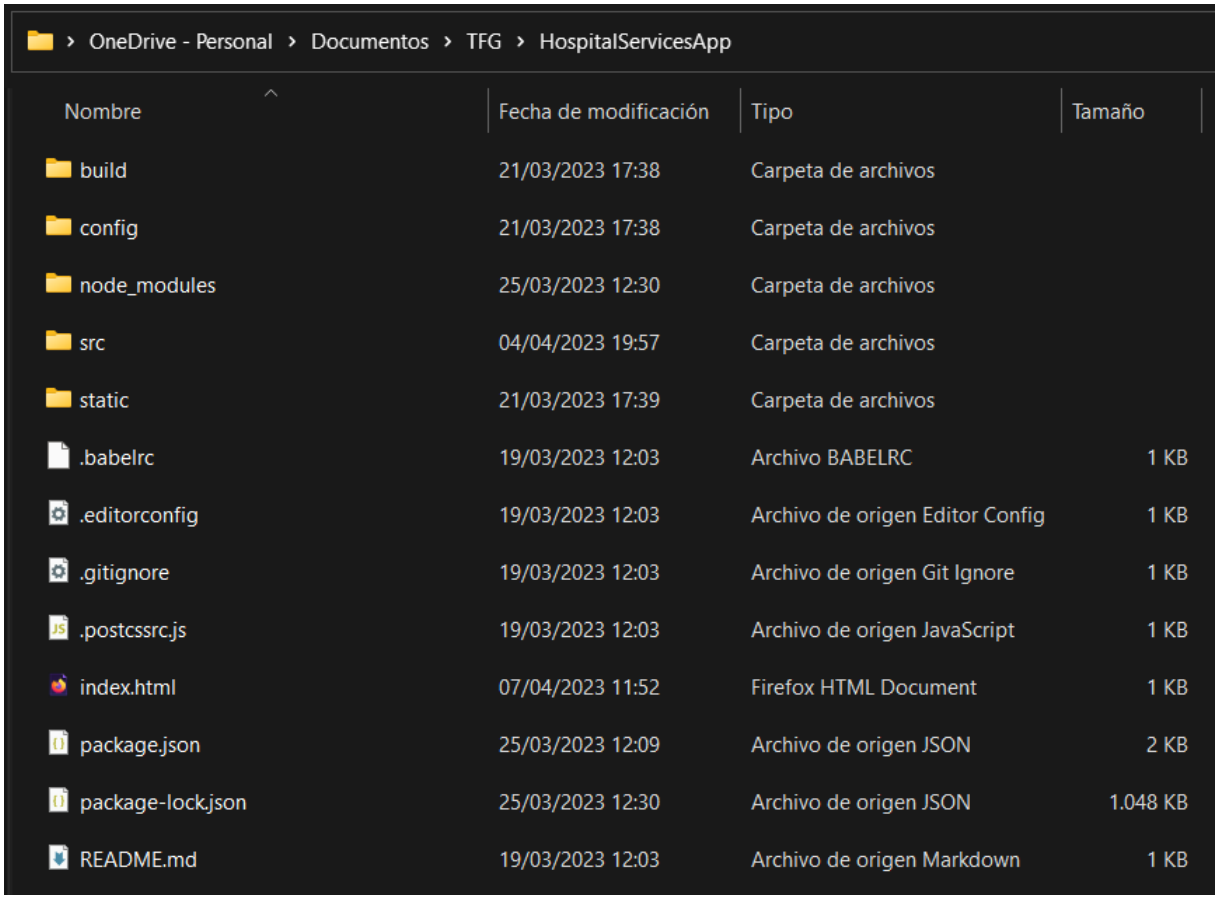

Figura 3.5: Directorio raíz de la aplicación web

Esta estructura está compuesta por un fichero package.json en la ruta raíz del proyecto, este fichero es muy importante ya que contiene todas las dependencias necesarias para el correcto funcionamiento del proyecto.

Para añadir dependencias al proyecto se ha utilizado el comando:

```
1 npm install nombreDeLaDependencia
```

El proyecto también contiene un fichero index.html con un contenedor principal, en el cual se irán añadiendo componentes en el proceso de compilado, el cual está ubicado en la ruta raíz del proyecto.

Tenemos un fichero main.js en la carpeta '/src', el cual crea una instancia de Vue, en la que se carga el componente raíz App.vue (el cual esta dentro de '/src'), véase la figura 3.6.

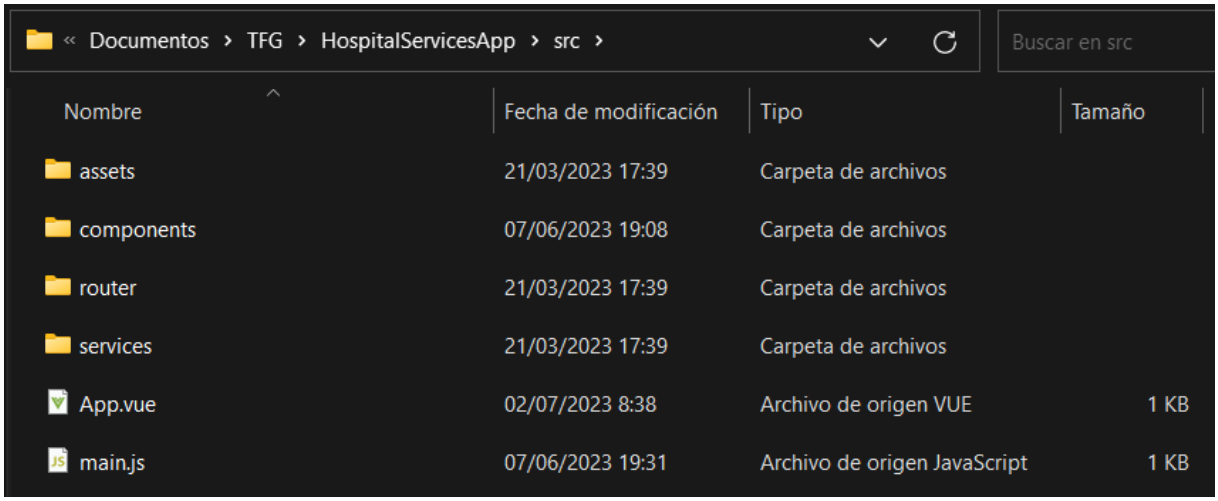

Figura 3.6: Directorio src de la aplicación web

Dentro de '/src' está la carpeta '/router' en la cual está el fichero index.js que contiene todas las rutas de navegación de vue-router.

Dentro de '/src' también tenemos la carpeta '/cli' en la que se almacena la librería que se comunicará con el servidor.

Por último dentro de '/src', tenemos la carpeta '/components', donde se almacenan todos los componentes Vue necesarios para el correcto funcionamiento de la aplicación, los cuales son (incluyendo App):

- **App.** Componente encargado de redirigir al login en caso de que el usuario no tenga un token asignado, o de lo contrario si se ha logueado y el servidor le ha otorgado un token redirigir al usuario a la intranet del sanitario o del paciente.
- **Login.** Encargado de validar los datos del usuario, y en caso de ser válidos asignarle el token proporcionado por el servidor al usuario.
- **SanitarioHome.** Componente que contiene la cabecera y el menú lateral del sanitario o administrador. Encargado de iniciar vue-router en el cual se irá mostrando el contenido de la sección seleccionada.
- **PacienteHome.** Componente que contiene la cabecera y el menú lateral del paciente. Encargado de iniciar vue-router en el cual se irá mostrando el contenido de la sección seleccionada.
- **Home.** Primer componente a mostrar en la pila de vue-router. Encargado de dar la bienvenida al usuario a la aplicación web.
- **User.** Componente encargado de mostrar la información personal del usuario y que permite modificar la contraseña del usuario.

- **Alarmas.** Encargado de mostrar las alarmas activas al sanitario o administrador y permite darlas de baja.
- **HistoricoPaciente.** Componente encargado de mostrarle al sanitario y al administrador la información del paciente seleccionado y de su histórico de servicios.
- **MedicionesServicio.** Componente encargado de mostrar la información del servicio asignado al paciente y de registrar mediciones.
- **NuevoServicio.** Componente que le permite al sanitario y al administrador dar de alta servicios.
- **Pacientes.** Componente encargado de mostrar el listado de pacientes de los que es responsable el sanitario o administrador que ha iniciado sesión.
- **Servicio.** Encargado de mostrar información del servicio seleccionado y de mostrar el listado de pacientes a los que se le ha asignado el servicio seleccionado.
- **ServicioAsignadoInfo.** Encargado de mostrar tanto al paciente, como al sanitario, como al administrador información de un servicio asignado a un paciente.

Cada uno de estos componentes esta formado por un *<template>* en el cual se establece la estructura de la parte visual del componente, y por otra parte por un *<script>* en el que se especifica la lógica que va a tener el componente. Y de manera opcional por un *<style>* para especificar reglas de diseño para la estructura especificada, como márgenes, colores, etc. Véase el componente Login de ejemplo:

```
<template>
  <div id="login">
    <!-- LOGO -->
    <div class="table-container">
      <table>
        <tr>
          <td>
            <md-icon class="md-size-5x">
                local_hospital
            </md-icon>
          </td>
          <td>
            <md-icon class="md-size-5x">
                notifications
            </md-icon>
          </td>
        </tr>
        <tr>
          <td>
            <md-icon class="md-size-5x">
                supervisor_account
            </md-icon>
          </td>
          <td>
            <md-icon class="md-size-5x">
```

```
26                  medical_information
27              </md-icon>
28            </td>
29          </tr>
30        </table>
31      </div>
32      <!-- /LOGO -->
33
34      <!-- FORMULARIO -->
35      <md-card>
36        <md-card-header>
37          <div class="md-title">Login</div>
38        </md-card-header>
39
40        <md-card-content>
41          <div class="form">
42            <md-field md-inline>
43              <label>DNI</label>
44              <md-input id="dni" v-model="form.dni"/>
45            </md-field>
46
47            <md-field>
48              <label>Password</label>
49              <md-input v-model="form.password" type="password"/>
50            </md-field>
51
52            <md-switch v-model="form.esSanitario">
53              Iniciar sesión como Sanitario
54            </md-switch>
55
56            <br />
57
58            <md-button class="md-raised md-accent" @click="enter">
59              Iniciar sesión
60            </md-button>
61          </div>
62        </md-card-content>
63      </md-card>
64      <!-- /FORMULARIO -->
65    </div>
66  </template>
67
68  <script>
69    export default {
70      name: "Login",
71      data: () => ({
72        form: {
73          dni: "",
74          esSanitario: false,
75          password: "",
76        },
77      }),
78      methods: {
79        getType(){
80          let type = ‘’
```

```
81          if(this.form.esSanitario === true){
82            type = 'S';
83          }else{
84            type = 'P';
85          }
86          return type;
87        },
88        enter() {
89          console.log(this.$user);
90          this.$model.login(
91            this.form.dni,
92            this.form.password,
93            this.getType(),
94            (err, token, usuario) => {
95              if (err) {
96                alert("Error" + err.stack);
97              } else {
98                this.$set(this.$user, "token", token);
99                for (var att in usuario){
100                 this.$set(this.$user, att, usuario[att]);
101               }
102               console.log("Usuario logeado");
103             }
104           }
105         );
106       },
107     },
108   };
109 </script>
110
111 <style>
112   .table-container {
113     display: flex;
114     justify-content: center;
115     align-items: center;
116     height: 40vh;
117   }
118   #login {
119     margin: 20%;
120   }
121 </style>
```

Listing 3.1: Componente Login.vue

Para encapsular todo lo anterior, la aplicación web está formada por un fichero main.js el cual carga el componente App.vue en el que se muestra el componente Login.vue mientras que el usuario no tenga un token (el cual proporciona el servidor si la autenticación es correcta). Una vez el usuario tenga el token, el componente App.vue te redirige al componente SanitarioHome.vue o PacienteHome.vue, en el que en ambos se carga tanto la cabecera como el menú lateral y se inicializa vue-router para que cargue en la pantalla el componente correspondiente en función de la ruta que carguemos, véase la figura 3.7.

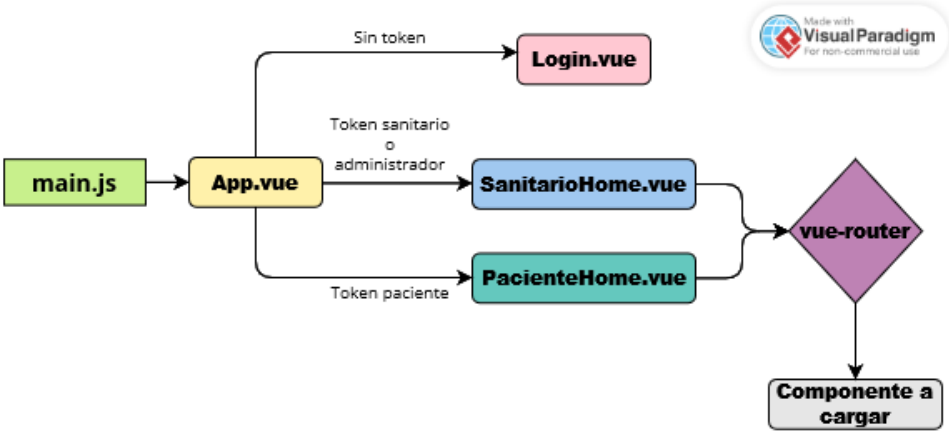

Figura 3.7: Estructura básica de la aplicación web, realizada en VisualParadigm

Por otra parte, la mayoría de estos componentes necesitan acceder al servicio RESTful y estos lo hacen a través de una librería llamada cli.js (esta sería la capa de comunicación). Para que un componente pueda utilizar esta librería, solamente tiene que ejecutar el método que necesite (ya que esta librería se importa de manera global en el fichero *main.js*), esta generará la petición al servidor y le devolverá una respuesta, véase la figura 3.8.

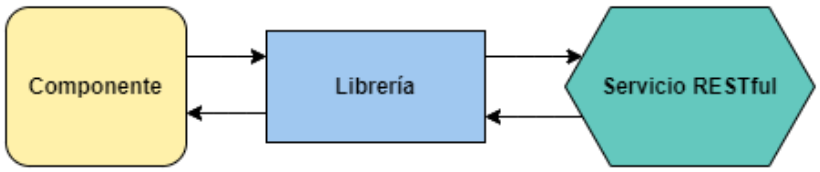

Figura 3.8: Comunicación de componentes con el servicio, realizada en VisualParadigm

3.3.1 Capa de la Interfaz gráfica de usuario (GUI)

La primera capa de la aplicación web es la interfaz gráfica de usuario, esta es la capa con la que interactúa el usuario. Es decir, esta capa contendrá todos los elementos visuales necesarios (formularios, listas, botones, etc) que permitirán visualizar y manejar los datos de forma clara para los diferentes usuarios.

En esta sección se detallarán los distintos *<template>* de los componentes creados y como se relacionan entre ellos (véase la figura 3.9 y la figura 3.10), los cuales todos ellos forman la GUI. Cabe destacar que para contruir estas interfaces se ha utilizado VueMaterial (Marcos Moura, 2015).

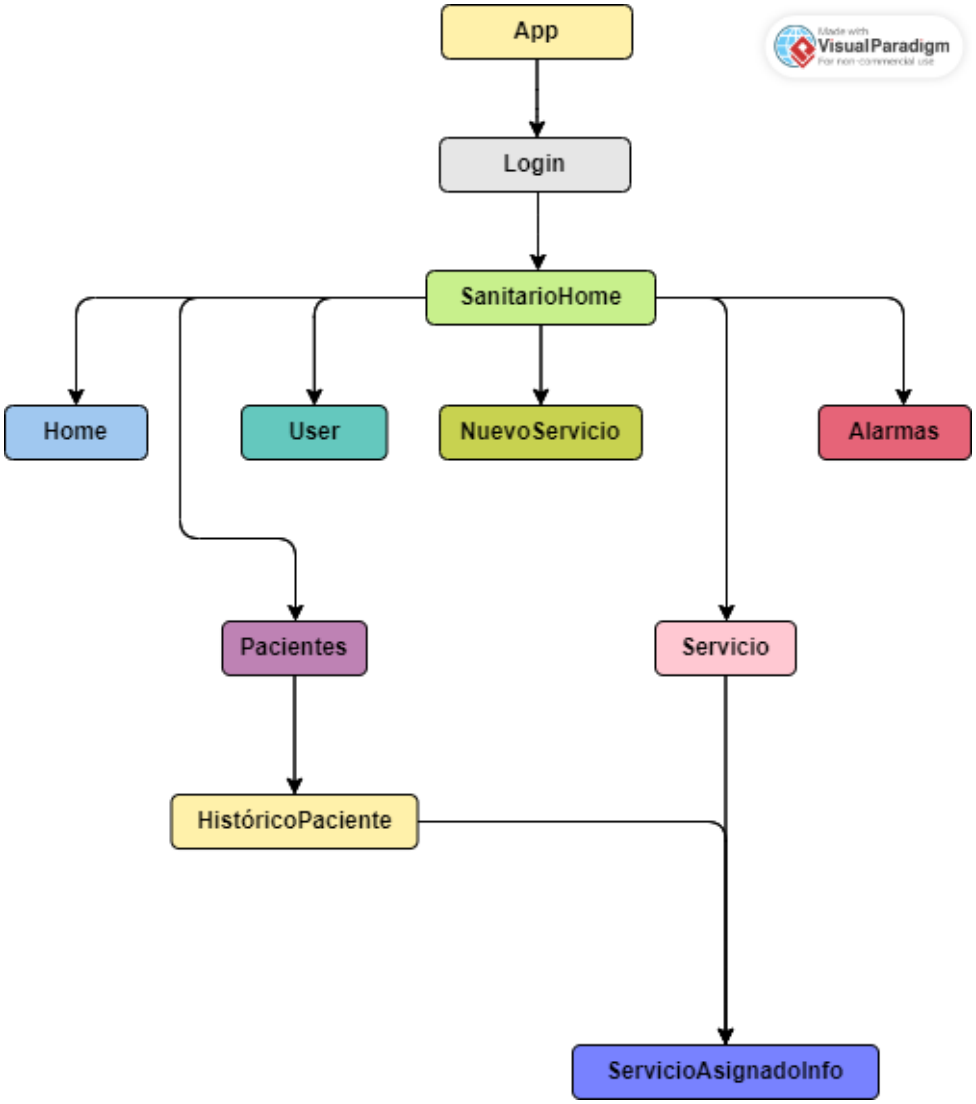

Figura 3.9: Mapa de ventanas del sanitario y administrador, realizada en VisualParadigm

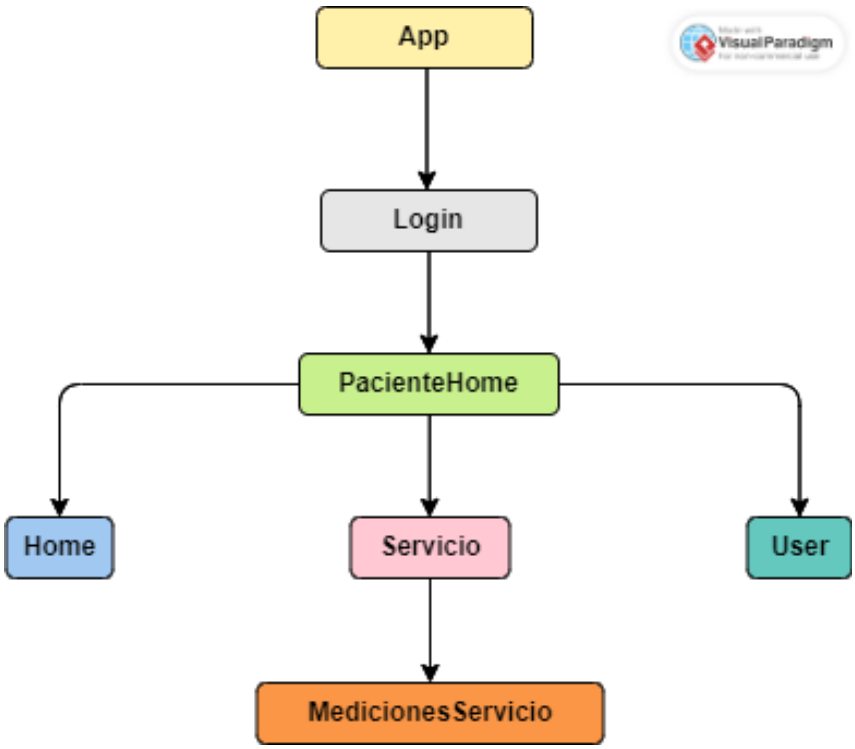

Figura 3.10: Mapa de ventanas del paciente, realizada en VisualParadigm

Tanto en el componente SanitarioHome como en PacienteHome, se inicializa vue-router el cual servirá para mantener una navegación dentro de la aplicación en la que los quitando los componentes App, Login, SanitarioHome y PacienteHome, los demás componentes irán asociados a una ruta y estos irán cambiando en función de la ruta activa (la cual iremos modificando desde la lógica de negocio). Estas rutas se especifican como ya hemos comentado en el fichero */router/index.js* y son las que podemos ver en la tabla 3.1.

RUTA	Componente
/home	Home
/sanitario/pacientes	Pacientes
/sanitario/alarmas	Alarmas
/sanitario/pacientes/*:dniPaciente*	HistoricoPaciente
/sanitario/pacientes/*:dniPaciente*/*:nombreServicioAsignado*	ServicioAsignadoInfo
/sanitario/servicios	Servicios
/sanitario/servicios/asignados/*:nombreServicioAsignado*	Servicio
/sanitario	User
/paciente	User
/paciente/servicios/*:nombreServicio*	MedicionesServicio

Tabla 3.1: Rutas de VueRouter

Por otra parte, se ha especificado que cualquier ruta que no este definida en la tabla 3.1, se redirija al componente Home. Esto se ha realizado indicando que cuando la ruta es *, se realice un *redirect* a dicho componente.

App

La interfaz App está formada unicamente por un *v-if* y dos *v-else-if* en los que se comprueban si el json *user* declarado de manera global en main.js contiene un token. En caso de que no tenga token se carga el componente Login.vue, de otra manera se comprueba si el json *user* tiene la variable *type* a *S* (Sanitario), *A* (Administrador) o *P* (Paciente), y redirige al usuario al componente SanitarioHome.vue o a PacienteHome.vue, tal y como se ve en el siguiente fragmento de código:

```
1 <template>
2   <div id="app">
3     <Login v-if="!user.token"></Login>
4     <SanitarioHome v-else-if="user.token && user.type === 'S' ||
5         user.type === 'A'"/>
6     <PacienteHome v-else-if="user.token && !user.type === 'P'"/>
7   </div>
8 </template>
```

Listing 3.2: Template del componente App.vue

Login

Esta interfaz está formada por un título para que el usuario conozca en que parte de la aplicación está, dos campos de texto, uno para introducir el dni del usuario y otro para la contraseña el cual ocultará los carácteres por razones obvias, un selector para que el usuario decida si va a iniciar sesión como paciente o como sanitario (aquí también entra el administrador), ya que un sanitario podría ser dado de alta como paciente. Y por último un botón para poder iniciar sesión.

SanitarioHome

Una vez accedido como sanitario o administrador, se muestra la interfaz 'SanitarioHome', esta permanece visible durante todas las interfaces.

Está formada por una cabecera en la cual se muestra información del usuario conectado y también contendrá un botón para cerrar la sesión.

También contiene un desplegable lateral por la parte izquierda de la pantalla en el cual se muestra un listado de las distintas funcionalidades que ofrece la aplicación (Información del usuario, Alarmas, Pacientes y Creación de servicios) y justo debajo de este, se muestra otro listado de los servicios de los cuales es responsable el sanitario.

En la parte superior del desplegable se habilita la sección 'Registro de pacientes' en caso de que haya iniciado sesión un administrador.

Por último, en el centro de la pantalla se irán mostrando las diferentes interfaces dependiendo de la parte de la aplicación en la que nos situemos.

Esta estructura formada por cabecera, menú lateral y contenido en el centro es una estructura de VueMaterial, en la que se encapsula todo dentro de un *<md-app></md-app>*, se establece la cabecera dentro de *<md-app-toolbar></md-app-toolbar>*, el menú lateral dentro de *<md-app-drawer></md-app-drawer>* y el contenido dentro de *<md-app-conent></md-app-conent>* en el que en nuestro caso se inicializa vue-router.

Home

Es la primera interfaz que muestra 'SanitarioHome', este simplemente mostrará al usuario un texto de bienvenida con el logo de la aplicación.

User

Formada por diferentes campos de texto los cuales mostrarán la información personal del usuario conectado, también hay un campo de texto y un botón en el cual el usuario podrá modificar su contraseña.

Alarmas

Interfaz encargada de visualizar las alarmas activas, también se muestra un botón el cual el sanitario pulsará para gestionar la alarma seleccionada.

HistoricoPaciente

Interfaz en la que se muestra información del paciente seleccionado, esta es sus datos personales y un listado de los servicios activos e inactivos los cuales se le ha asignado.

NuevoServicio

Formada por un formulario de tres campos, estos son el nombre, descripción y las unidades de las mediciones del servicio. Encargada de dar de alta nuevos servicios, si el servicio se ha registrado correctamente, se muestra una ventana flotante confirmando el registro.

Pacientes

Interfaz que muestra el listado de pacientes los cuales es responsable un doctor. Desde esta interfaz, el sanitario puede dar de alta pacientes en el sistema pulsando sobre el botón de 'Registro de pacientes', en el que se abre una ventana flotante con un formulario con los datos del paciente a registrar. Una vez registrado el paciente, se muestra otra ventana flotante de confirmación.

Servicio

Interfaz encargada de mostrar información del servicio y del listado de pacientes a los que se le ha asignado ese servicio. Desde esta interfaz, el sanitario puede asignar el servicio seleccionado a pacientes pulsando sobre el botón de 'Asignación de servicio', en el que se abre una ventana flotante con un formulario donde se indica el dni del paciente al que se le va asignar el servicio, el intervalo de tiempo entre mediciones y indicaciones que debe realizar el paciente antes de realizar mediciones. Si el servicio se ha asignado correctamente, se muestra un 'PopUp' de confirmación.

ServicioAsignadoInfo

Interfaz encargada de mostrar información del servicio asignado a un paciente seleccionado. Desde esta interfaz, el sanitario puede dar de baja a ese paciente del servicio asignado pulsando el botón de 'Dar de baja', en el que se abre un 'PopUp' advirtiendo que va a dar de baja un servicio asignado. Si el sanitario confirma nuevamente, da de baja el servicio asignado a ese paciente.

PacienteHome

Al igual que la interfaz 'SanitarioHome', esta interfaz permanece visible durante todas las interfaces una vez conectado como paciente.

También cuenta con una cabecera en la que se muestra información del usuario conectado y que también contiene un botón para cerrar la sesión.

Dispone de un menú lateral en el que a diferencia de la interfaz 'SanitarioHome', esta contiene únicamente en la parte superior la sección de 'Información del usuario', y en la parte superior el listado de servicios que le han asignado y que están activos.

MedicionesServicio

En esta interfaz, se muestra información del servicio asignado al paciente necesaria para tomar la medición, desde esta interfaz se muestra el listado de mediciones que ha estado realizando el paciente, y también puede registrar nuevas pulsando en el botón de 'Registrar nueva medición'. Si el paciente pulsa sobre este botón, se abre una ventana flotante en el que el paciente puede introducir el valor de la medición en las unidades que especificó el sanitario.

3.3.2 Capa de Lógica de negocio

La segunda capa está también ubicada en la aplicación web y esta es la lógica de negocio. Esta se encargará de toda la lógica de la aplicación, esto es el procesado de datos que le llega tanto de la GUI como de la capa de comunicación, y la navegación dentro de la aplicación.

En esta sección veremos las diferentes lógicas que se han implementado en cada interfaz (en caso de que exista tal lógica en el componente) y que todas ellas forman la capa de lógica de negocio.

Login

Recibe los datos de la GUI, crea un json con todos estos datos para formar una entidad usuario y le manda esta información a la capa de comunicación. En caso de que esta última nos responda de forma correcta, almacenamos la información del usuario en una variable global llamada 'user' incluída un token, el cual será necesario para poder realizar acciones en la aplicación y se le comunicará a la GUI, de otra forma se generará un error y se le comunicará a la GUI.

En caso de que haya iniciado sesión un sanitario o administrador, se carga la ruta del 'SanitarioHome', si ha iniciado sesión como paciente, se carga la ruta del 'PacienteHome'.

SanitarioHome

Al crearse esta interfaz, la capa de negocio le solicitará a la capa de comunicación las alarmas activas y los servicios que están a cargo del sanitario y le enviará los datos a la GUI.

Se vaciará el contenido del usuario en caso de cierre de sesión, de esta forma Vue.js detectará que el usuario ya no tiene token y la GUI se encargará de mostrar de nuevo la ventana de 'Login'.

También se comprueba si el usuario conectado es un administrador o no, en caso de que lo sea se habilita la sección en el menú de 'Registrar sanitario', en la que si se selecciona se abre un 'PopUp' con un formulario para dar de alta sanitarios.

Se cargará la ruta de 'User', 'Alarmas', 'Pacientes', 'NuevoServicio' o 'Servicio' en función de las peticiones de la capa de la interfaz de usuario. La forma de cargar una ruta es añadiendola a la pila de vue-router con el comando *push*, por ejemplo:

```
1 servicioSeleccionado(servicio){
2       this.servicioSelected = servicio;
3       this.$router.push({name: "Servicio",
4         params:{
5                     idServicioAsignado: servicio.nombre
6                }
7         }).catch(err => {
```

```
8                 console.log(err.name);
9       });
10  }
```

Listing 3.3: Método onClick de un servicio

Por otra parte al inicializarse vue-router se especifican dos métodos en caso de que se produzcan dos eventos. El primero es un evento (addServicioEvent) que genera el componente 'NuevoServicio', el cual cuando se active se llama al método addServicio el cual añade el nuevo servicio a la lista del menú lateral. Por otra parte se ejecuta el metodo 'alarmaGestionada' en caso de que se produzca el evento 'alarmasEvent' el cual genera el componente Alarmas al gestionar una alarma. Lo que hace el método es reducir en uno el contador de alarmas activas que se muestra en el menú lateral.

Se pasa como parámetro al vue-router las alarmas activas, esto se hace ya es necesario obtenerlas en este componente para mostrar el contador de numero de alarmas activas, por lo que para evitarnos otra segunda llamada al servicio, se le pasan a vue-router como parámetro para su uso en el componente 'Alarmas'.

Y por último se pasa también como parámetro el servicio seleccionado a vue-router, el cual es necesario para el componente 'Servicio'.

Alarmas

En este componente se cargan las alarmas que hemos recibido del componente 'SanitarioHome' en la lista y en caso de que se gestione una alarma, se llama a la función necesaria de la capa de comunicación y en caso de que se gestione de manera correcta la alarma, producimos el evento 'alarmasEvent' el cual captará el componente 'SanitarioHome' para reducir el numero de alarmas activas. El evento lo producimos de la siguiente manera:

```
1  this.$emit('alarmasEvent');
```

Listing 3.4: Producir evento 'alarmasEvent'

Pacientes

Al cargarse este componente, este obtiene los pacientes los cuales es responsable el sanitario que ha iniciado sesión, y se muestran en la lista. Al pulsar sobre un paciente, se llama al método 'abrirHistoricoPaciente' el cual carga la ruta del componente 'HistoricoPaciente' con un 'push'.

Al pulsar sobre el botón de registro de pacientes se cambia el valor de la variable 'registroPaciente' a true, al ser esta variable reactiva se abre el popUp de registro de usuarios y al guardar el paciente se vacía el formulario y se vuelve a poner a false esta variable para ocultar el popUp. Esta es la lógica seguida para mostrar y ocultar popUps en la aplicación.

Por otra parte el método que se ejecuta cada vez que se escribe algo en el buscador de la lista, está basado en el método buscar de Table search and empty state de Vue Material.

HistoricoPaciente

En este componente se muestra información tanto del paciente como del histórico de servicios que le han sido asignados. Por lo que al ser montado este componente, se obtiene esta información a través de la capa de comunicación. Al recibir el histórico de servicios, se hace necesario formatear la fecha de alta y de baja de cada servicio para que se muestre con un formato más entendible, esto se hace con el siguiente método:

```
formateaFechaYhora(fechaAformatear){
    if(fechaAformatear === null){
        return null;
    }
    var proxMed = new Date(fechaAformatear);
    var dia = proxMed.getDate();
    var mes = proxMed.getMonth() + 1;
    var anio = proxMed.getFullYear();
    var horas = proxMed.getHours();
    var minutos = proxMed.getMinutes();

    var fechaFormateada = ('0' + dia).slice(-2) + '/' +
    ('0' + mes).slice(-2) + '/' + anio;

    var horaFormateada = ('0' + horas).slice(-2) + ':' +
    ('0' + minutos).slice(-2);

    return fechaFormateada + ' ' + horaFormateada;
}
```

Listing 3.5: Producir evento 'alarmasEvent'

NuevoServicio

En este componente después de registrar un nuevo servicio, se limpian los valores del formulario, y se genera el evento 'addServicioEvent' para que el componente 'SanitarioHome' lo detecte y lo añada a la lista inferior del menú lateral. Para generar el evento pasando el servicio como parámetro se utiliza:

```
this.$emit('addServicioEvent', this.servicio);
```

Listing 3.6: Producir evento 'addServicioEvent' pasando como parámetro un servicio

Servicio

En este componente, se ha utilizado 'watch' para detectar cuando se modifica el servicio seleccionado y actualizar los datos del mismo. Esto se ha tenido que realizar de esta manera, ya que antes se obtenían los pacientes al montarse el componente, pero al pasar entre servicios, al utilizar el mismo componente Vue (Servicio) no se obtenían los pacientes. Por este motivo se utiliza 'watch', y cuando se cambian las pantallas entre servicios, se actualizan la lista de pacientes del mismo.

ServicioAsignadoInfo

En este componente, se comprueba al cargar la información si el servicio asignado al paciente, ha sido dado de baja o no, esto se hace para deshabilitar el botón de dar de baja en caso de que el servicio ya lo esté. Al dar de baja un servicio asignado también se deshabilita este botón modificando el valor de la variable 'deshabilitaBaja' a true.

PacienteHome

Al crearse este componente se solicita a la capa de comunicación los servicios activos que tiene el paciente que ha iniciado sesión para cargarse la lista del menú lateral.

Al igual que el componente 'SanitarioHome', se vaciará el contenido del usuario en caso de cierre de sesión, de esta forma Vue.js detectará que el usuario ya no tiene token y la GUI se encargará de mostrar de nuevo la ventana de 'Login'.

Y se cargará la ruta de 'User' o 'Servicio' en función de las peticiones de la interfaz de usuario haciendo uso de 'push'.

MedicionesServicio

Este componente formatea las fechas que muestra la GUI como ya se ha detallado en la subsección 3.3.2.

Por otra parte al dar de alta una medición se obtiene la fecha y la hora de la próxima medición haciendo uso del método obtenerProximaMedicion(), el cual obtiene el intervalo de mediciones del servicio asignado, el cual esta formado por un numero y una letra (horas 'h', días 'd', semanas 's'), y lo que hace es separar primero el valor de la letra y añadir el tiempo necesario para generar la nueva fecha:

```
1 obtenerProximaMedicion(){
2     let prxMed = new Date()
3     let indice = 0;
4     for (var i=0; i<this.servicioAsignado.intervalo.length; i++)
          {
5         if (isNaN(parseInt(this.servicioAsignado.intervalo[i])))
              {
6             indice = i;
```

```
7                 break;
8             }
9         }
10        var valor = parseInt(this.servicioAsignado.intervalo
11                        .substring(0,indice));
12        var intervalo = this.servicioAsignado.intervalo
13                        .substring(indice);
14        switch(intervalo){
15            case 'h':
16                prxMed.setHours(prxMed.getHours() + valor);
17                break;
18            case 'd':
19                prxMed.setDate(prxMed.getDate() + valor);
20                break;
21            case 's':
22                prxMed.setDate(prxMed.getDate() + 7);
23                break;
24        }
25        return prxMed;
26    }
```

Listing 3.7: Método para obtener la próxima medición

3.3.3 Capa de Comunicación

La capa de comunicación ubicada en la aplicación web es la encargada de recibir los datos ya procesados de la capa de la lógica de negocio y de crear y mandar peticiones a la capa de comunicación ubicada en el servicio RESTful.

Por otra parte, esta también será la encargada de recibir las respuestas que provienen del servicio RESTful y comunicarselas a la capa de la lógica de negocio.

Las rutas creadas siempre van con el prefijo 'http://*dominio*:*puerto*/**hospitalServices**', a partir de esta se irán añadiendo rutas, todas ellas siguiendo el modelo establecido en la tesis doctoral de Fielding, 2000.

Las rutas creadas por esta capa, en función de las peticiones de la capa de la lógica de negocio, y las cuales serán enviadas al servicio RESTful son las que aparecen en la tabla 3.2:

RUTA	TIPO	Acción
/hospitalServices/sessions	POST	Iniciar sesión
/hospitalServices/users/*idUser*	PUT	Actualizar contraseña
/hospitalServices/users	POST	Añadir usuario
/hospitalServices/pacientes	GET	Obtener pacientes
/hospitalServices/servicios	POST	Añadir servicio
/hospitalServices/servicios/asignados	POST	Asignar un servicio a un paciente
/hospitalServices/servicios/asignados /*idServicioAsignado*	GET	Obtener los pacientes de un servicio asignado
/hospitalServices/servicios/asignados /*idServicioAsignado*	PUT	Dar de baja un servicio asignado a un paciente
/hospitalServices/servicios/asignados /*idServicioAsignado*/mediciones	POST	Añadir una medición
/hospitalServices/servicios/asignados /*idServicioAsignado*/*idPaciente*	GET	Obtener el servicio asignado de un paciente
/hospitalServices/sanitarios/*idSanitario*/servicios	POST	Obtener servicios del sanitario
/hospitalServices/pacientes/*idPaciente*/servicios /asignados	GET	Obtener servicios asignados del paciente
/hospitalServices/alarmas	GET	Obtener las alarmas responsables del sanitario
/hospitalServices/alarmas/*idAlarma*	PUT	Dar de baja una alarma

Tabla 3.2: Tipos de ruta en función de la acción

Una vez llega a esta capa la respuesta por parte del servicio RESTful, la forma de actuar independientemente de la ruta enviada (vistas en la tabla 3.2) es similar. Se comprueba si el mensaje recibido es un error o en caso contrario la respuesta contiente datos, y se le envía esta información de vuelta a la GUI.

3.4 Servicio RESTful

El servicio RESTful desarrollado se basa en dos capas, la capa de comunicación en la que el servicio está escuchando peticiones de clientes y respondiendo a estos. Y por otra parte la capa de acceso a datos, esta es una librería la cual tiene todas las funciones necesarias por los clientes para poder manejar los datos, véase la figura 3.11.

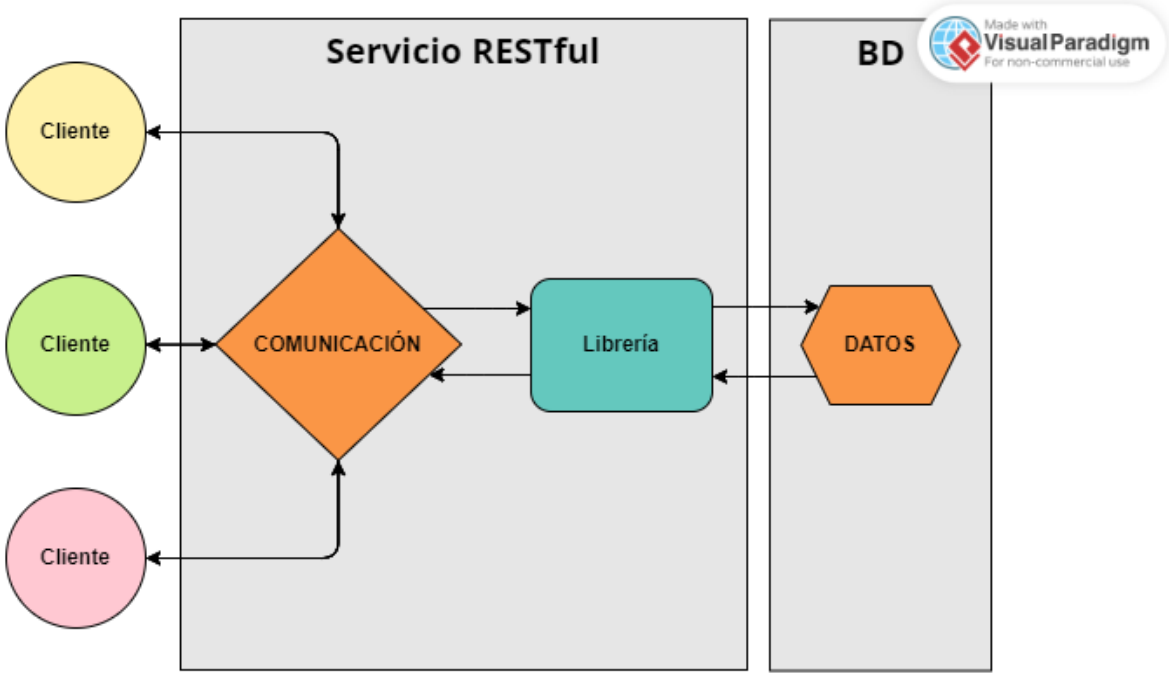

Figura 3.11: Clientes accediendo a bd a través del servicio RESTful, realizada en VisualParadigm

3.4.1 Capa de Comunicación

En la capa de comunicación se utiliza Express para poner al servidor en escucha en un puerto específico el cual atacarán los clientes. A este servidor se le indica que va a utilizar body-parser y cors los cuales son necesario como ya hemos visto en la sección 3.2.

```
1  const express = require('express');
2  const cors = require('cors');
3  const bodyParser = require('body-parser');
4  const port = 8081;
5
6  let app = express();
7
8  app.use(bodyParser.json());
9  app.use(cors());
10
11 app.listen(port, () =>{
12     console.log('=========[SERVER STARTED p:${port}]=========');
13 });
```

Listing 3.8: Inicialización del servidor

Cuando el servidor recibe una petición comprueba en primer lugar si tiene token, si en la petición del cliente, la cual no es la del login, no se pasa un token ni dentro de la query (cuando es una petición GET) ni dentro dentro del json params que se ubica dentro de body (cuando es una petición POST, PUT y DELETE), la petición se rechaza enviándole

al cliente un mensaje de 'Token not found', en cualquier otro caso se deja pasar para el tratamiento de esa petición:

```
app.use(function (req, res, next) {
    console.log('authorize ' + req.method +
    + ' ' + req.originalUrl);

    if ((req.path == '/hospitalServices/sessions' &&
         req.method == 'POST')) {
        next();
    } else if (!req.query.token && !req.body.params.token){
        res.status(401).send('Token not found');
    } else next();
});
```

Listing 3.9: Autorización de peticiones en el servicio RESTful

Una vez pasado este nivel de autorización, todas le peticiones se gestionan de una manera similar. Se llama a la función de la librería de acceso a datos que corresponda, y en tanto si esta nos devuelve datos o un error, se responde a cada cliente con la información o el error correspondiente.

Como se ha comentado antes la única diferencia es como se accede a los datos que nos envía el cliente, en caso de ser una petición *get* se obtienen los datos del campo *query* dentro de la *request* y en cualquier otro caso se obtienen dentro del campo *params* ubicado dentro del body de la *request*:

```
//GET PACIENTES
app.get('/hospitalServices/pacientes', function (req, res) {
    console.log('list pacientes');
    model.listaPacientesPorSanitario(req.query.token,
        req.query.dni, (err, pacientes) => {
        if (err) {
            console.log(err.stack);
            res.status(400).send(err);
        } else {
            res.send(pacientes);
        }
    });
});

// ADD PACIENTE
app.post('/hospitalServices/pacientes', function (req, res) {
    console.log('add paciente ' +
    + JSON.stringify(req.body.params.user));
    model.addPaciente(req.body.params.token,
                      req.body.params.user, (err, user) => {
        if (err) {
            console.log(err.stack);
            res.status(400).send(err);
        } else res.send(user);
    });
});
```

Listing 3.10: Obtención de datos en petición GET y POST

Por último, como se ha comentado anteriormente, esta capa hace de intermediaria entre los clientes y la capa de acceso a datos, por lo que dependiendo de la petición del cliente esta capa realizará una llamada a la librería la cual accede a los datos, estas distintas redirecciones, son las que se pueden ver en la tabla 3.3.

PETICIÓN DEL CLIENTE	TIPO	FUNCIÓN CAD
/hospitalServices/sessions	POST	login()
/hospitalServices/users/*idUser*	PUT	updatePassword()
/hospitalServices/users	POST	addUser()
/hospitalServices/pacientes	GET	listaPacientesPorSanitario()
/hospitalServices/servicios	POST	addServicio()
/hospitalServices/servicios/asignados	POST	addServicioAsignado()
/hospitalServices/servicios/asignados /*idServicioAsignado*	GET	getPacientesPorServicioAsignado()
/hospitalServices/servicios/asignados /*idServicioAsignado*	PUT	deleteServicioAsignado()
/hospitalServices/servicios/asignados /*idServicioAsignado*/mediciones	POST	addMedicion()
/hospitalServices/servicios/asignados /*idServicioAsignado*/*idPaciente*	GET	getServicioAsignadoPaciente()
/hospitalServices/sanitarios /*idSanitario*/servicios	POST	getServiciosSanitario()
/hospitalServices/pacientes/*idPaciente* /servicios/asignados	GET	getServiciosAsignadosPaciente()
/hospitalServices/alarmas	GET	getAlarmas()
/hospitalServices/alarmas/*idAlarma*	PUT	gestionarAlarma()

Tabla 3.3: Función de la capa de acceso a datos (CAD) según la petición del cliente

3.4.2 Capa de Acceso a datos

Esta capa en forma de librería, es la encargada de acceder a la base de datos y por lo tanto de obtener y manipular los datos. A esta capa solo tiene acceso la capa de comunicación del servicio RESTful y está formada por diferentes funciones las cuales tienen todas la misma estructura, en las que se validan al principio los parámetros de entrada, se abre una conexión a la base de datos de Mongo DB, se realizan las operaciones necesarias en la base de datos, se cierra la conexión y se responde con un callback. Estas funciones son las que se van a detallar en esta subsección.

login()

Este método es el único que no necesita un token, se recibe como entrada el dni, password y si el usuario se está conectando como paciente, sanitario o administrador, se comprueba si existe tal documento en la base de datos y en caso afirmativo se devuelve como *token* el *ObjectId* de el usuario encontrado.

addUser()

Método encargado de añadir un nuevo paciente o sanitario en la base de datos, se le pasa como entrada el token del usuario conectado, el usuario a insertar y un callback. Se comprueba que todos los campos del json usuario tienen datos, se comprueba que el token del usuario conectado es válido y corresponde con el de un

- **Sanitario.** En caso de que se vaya a añadir un paciente.
- **Administrador.** En caso de que se vaya a añadir un sanitario.

Se comprueba que el dni del paciente o del sanitario a insertar no exista en la base de datos, y si cumple todos los requisitos anteriores se añade un documento en la tabla 'usuarios' con la siguiente estructura:

```
{
  "_id": {
   "$oid": "ObjectId generado automáticamente por MongoDB (token
      )"
  },
  "nombre": "Nombre del paciente",
  "apellido1": "Primer apellido del paciente",
  "apellido2": "Segundo apellido del paciente",
  "password": "Password del paciente",
  "dni": "DNI del paciente",
  "type": "P",
  "fechaDeAlta": Fecha de alta del paciente
}
```

Listing 3.11: Documento que representa un usuario de tipo paciente

```
{
  "_id": {
   "$oid": "ObjectId generado automáticamente por MongoDB (token
      )"
  },
  "nombre": "Nombre del sanitario",
  "apellido1": "Primer apellido del sanitario",
  "apellido2": "Segundo apellido del sanitario",
  "password": "Password del sanitario",
  "dni": "DNI del sanitario",
  "type": "S",
```

```
11    "fechaDeAlta": Fecha de alta del sanitario,
12    "pacientes": [],
13    "servicios": []
14  }
```

Listing 3.12: Documento que representa un usuario de tipo sanitario

Y se devuelve la información del paciente o sanitario registrado en el callback, si una de las comprobaciones anteriores falla se devuelve el callback con un error.

addServicio()

Método encargado de registrar un nuevo servicio en la base de datos. Se le pasa como parámetros de entrada el token del usuario conectado, un objeto de tipo servicio y un callback, se comprueba que el objeto servicio tiene todos los campos necesarios, se comprueba que el token es válido y que es de un sanitario o administrador y si es así se inserta en la tabla 'servicios' un documento con la siguiente estructura:

```
1  {
2    "_id": {
3      "$oid": "ObjectId generado automáticamente por MongoDB"
4    },
5    "nombre": "Nombre del servicio",
6    "descripcion": "Descripción del servicio",
7    "unidades": "Unidades en las que se tomarán las mediciones
8                 de ese servicio"
9  }
```

Listing 3.13: Documento que representa un servicio

se inserta en el array 'servicios' del documento que hace referencia al 'usuario' del sanitario el identificador del nuevo servicio y se devuelve la información del servicio nuevo en el callback, si una de las comprobaciones anteriores falla se devuelve el callback con un error.

addServicioAsignado()

Método encargado de registrar un nuevo 'servicioAsignado'. Se le pasa como parámetros de entrada el token del usuario conectado, el 'servicioAsignado' a registrar y un callback. En caso de que falte algún campo del 'servicioAsignado' a añadir, el token no sea válido o no pertenezca a un sanitario o administrador, se devuelve el callback con un error. En otro caso, se inserta en la tabla 'serviciosAsignados' un elemento con la siguiente estructura:

```
1  {
2    "_id": {
3      "$oid": "ObjectId generado automáticamente por MongoDB"
4    },
5    "servicio": {
6      "$oid": "ObjectId del servicio a asignar"
```

```
  },
  "fechaAlta": {
    "$date": "Fecha de alta"
  },
  "fechaBaja": null,
  "proximaMedicion": {
    "$date": "Fecha y hora de la proxima medicion"
  },
  "intervalo": "Intervalo de tiempo entre mediciones",
  "valorMax": "Valor límite para la generación de alarma",
  "activo": true,
  "observaciones": "Indicaciones del sanitario",
  "mediciones": [Array de mediciones],
  "sanitarioResponsable": "DNI del sanitario responsable",
  "paciente": "DNI del paciente al que se le ha asignado el
     servicio"
}
```

Listing 3.14: Documento que representa un servicio asignado

Y se devuelve en el callback el 'servicioAsignado' insertado.

addMedicion()

Método encargado de añadir una medición a un servicio asignado, se pasan como parámetro de entrada el token del usuario conectado, el identificador del servicio asignado, un objeto medición y la fecha y hora de la próxima medición. Se comprueba que el token es válido y que pertenece a un usuario, que el paciente pertenece al servicio asignado al que se le quiere añadir la medición y que el objeto de tipo medición contiene los datos de la fecha y hora a la que se ha registrado la medición y el valor de la medición.

En caso de que alguna comprobación de las anteriores no se cumpla, se devuelve el callback con el error registrado.

En caso de que se cumpla todo lo anterior, se comprueba si el valor de la medición es superior al campo 'valorMax' del documento 'servicioAsignado' (visto en la subsección 3.4.2). En caso de que el valor de la medición sea inferior, se añade al objeto medición el campo alarma con valor a false, y se añade al array 'mediciones' la medición.

En caso de que el valor de la alarma sea superior al campo 'valorMax', se añade al objeto medición el campo alarma con valor a true, y se inserta en la tabla 'alarmas' un nuevo documento con la siguiente estructura:

```
{
  "_id": {
    "$oid": "ObjectId generado automáticamente por MongoDB"
  },
  "servicioAsignado": {
    "$oid": "ObjectId del servicio asignado"
  },
```

```
8     "fechaGenerada": {
9       "$date": "Fecha y hora a la que se generó la alarma"
10    },
11    "fechaGestionada": null
12  }
```

Listing 3.15: Documento que representa una alarma

Por último, se le devuelve el callback con la medición insertada.

updatePassword()

Encargado de actualizar la contraseña de un usuario. Se pasan como parámetros de entrada el token del usuario, la nueva contraseña y el callback.

Se comprueba que la contraseña y el token son válidos, y se modifica el campo password del usuario que coincida con el token recibido.

listaPacientesPorSanitario()

Método encargado de obtener los pacientes de los que es responsable un sanitario o administrador, se le pasa como parámetro de entrada el token del sanitario o administrador.

Se comprueba que el token es válido y que pertenece a un sanitario o administrador, en caso de que se cumpla lo anterior se devuelve el callback con el array de pacientes, en cualquier otro caso, se devuelve el callcbock con el error generado.

getPacientesPorServicioAsignado()

Método encargado de obtener los pacientes a los que se le ha asignado un servicio específico. Se pasa como parámetros el token del usuario conectado, el identificador del servicio y un callback.

Se comprueba que el token es válido y que pertenece a un sanitario o administrador y se comprueba que el identificador del servicio es válido.

Si se cumple lo anterior se devuelve a la capa de comunicación el callback con el array de pacientes, en otro caso se devuelve el callback con el error generado.

deleteServicioAsignado()

Método encargado de realizar un borrado lógico de un servicio asignado, se realiza un borrado lógico para poder mantener el histórico de un paciente. Se pasan como parámetros de entrada el token del usuario conectado, el identificador del servicio asignado y un callback.

Se comprueba de que el token del usuario sea válido y que este pertenezca a un sanitario o administrador, se comprueba que el identificador del servicio asignado sea válido y de que el sanitario que va a dar de baja un servicio asignado sea el responsable del mismo.

En caso de que se cumpla lo anterior se modifica el campo activo a false del documento 'servicioAsignado' (visto en la subsección 3.4.2) y se devuelve el callback con el identificador del 'servicioAsignado'. En cualquier otro caso, se devuelve el callback con el error generado.

getServicioAsignadoPaciente()

Método encargado de obtener el servicio asignado de un paciente. Se pasan como parámetros de entrada el token del usuario conectado, el dni del paciente, el identificador del servicio y un callback.

Se comprueba que el token del usuario sea válido y que pertenezca a un sanitario, administrador o al paciente el cual tiene el servicio asignado, y se comprueba que el identificador del servicio sea válido.

Si son válidas las comprobaciones anteriores se devuelve el callback con la informaciónd del servicio asignado, si no es válida alguna comprobación, se devuelve el callback con el error que se ha generado.

getServiciosSanitario()

Método encargado de obtener los servicios creados por un sanitario. Se pasan como parámetros el token del usuario conectado y un callback.

Se comprueba que el token es válido y que pertenece a un sanitario o administrador, en ese caso se devuelve un array con los servicios que ha creado el sanitario o administrador, en otro caso se devuelve el callback con el error generado.

getAlarmas()

Método encargado de obtener las alarmas responsables de un sanitario o administrador. Se pasan como parámetros el token del usuario conectado y un callback.

Se comprueba que el token del usuario sea válido y que pertenezca a un sanitario o administrador, si esto se cumple se devuelven todas las alarmas las cuales es responsable el sanitario responsable y que tengan el campo 'fechaGestionada' de la tabla alarmas a null (esto quiere decir que aún no ha sido gestionada), en otro caso se devuelve el callback con el error generado.

gestionarAlarma()

Encargado de dar de baja una alarma, se le pasan como parámetros de entrada el token del usuario conectado, la alarma a gestionar y un callback.

Se comprueba que el token sea válido y que pertenezca a un sanitario o administrador, en este caso se modifica el campo 'fechaGestionada' a la fecha que nos viene como parámetro dentro del campo alarma que nos pasan como entrada, en cualquier otro caso, se devuelve el callback con el error generado.

3.5 Base de datos

La base de datos está formada por cuatro colecciones (o tablas) las cuales almacenan todos los datos necesarios para el correcto funcionamiento de este proyecto, estas son: usuarios, servicios, serviciosAsignados y alarmas (figura 3.12). En esta sección entraremos en más detalle en cada una de estas colecciones.

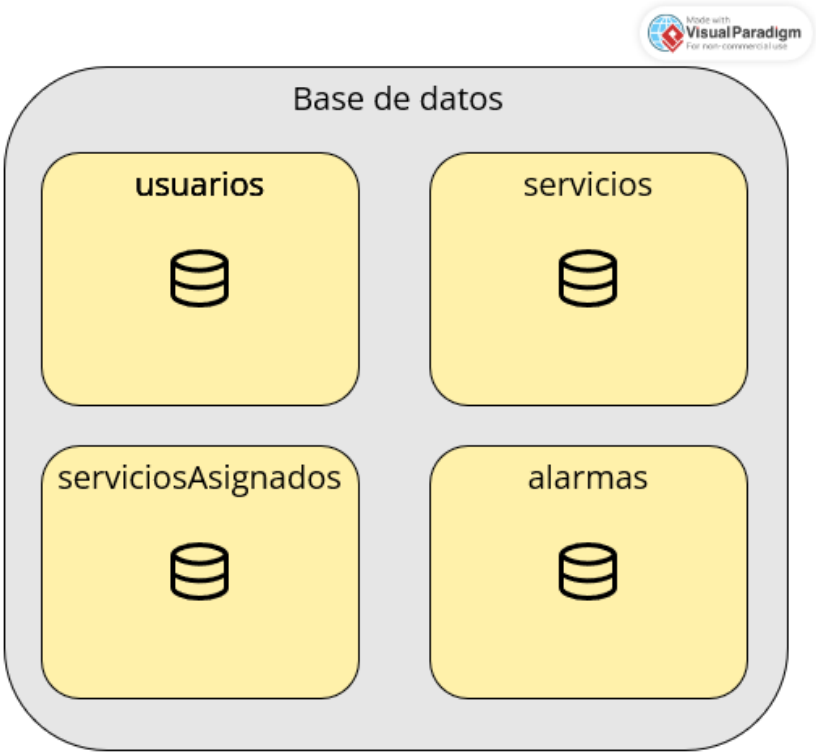

Figura 3.12: Colecciones de la base de datos, realizada en VisualParadigm

3.5.1 usuarios

En esta colección se almacenan todos los usuarios del sistema. Dependiendo de si se va a almacenar un paciente o un sanitario, el documento a insertar tendrá unas columnas o otras (la estructura de un usuario de tipo administrador, es la del sanitario salvo el valor de la columna *type*), véase la tabla 3.4 y la tabla 3.5.

Paciente

COLUMNA	TIPO	DESCRIPCIÓN
_id	ObjectId	Identificador del paciente
nombre	String	Nombre del paciente
apellido1	String	Primer apellido del paciente
apellido2	String	Segundo apellido del paciente
password	String	Contraseña del paciente
dni	String	DNI del paciente
fechaDeAlta	Date	Fecha en la que fue dado de alta el paciente
type	String	Tipo de usuario ('P': Paciente)

Tabla 3.4: Columnas de un usuario de tipo paciente

Sanitario o administrador

COLUMNA	TIPO	DESCRIPCIÓN
_id	ObjectId	Identificador del sanitario o administrador
nombre	String	Nombre del sanitario o administrador
apellido1	String	Primer apellido del sanitario o administrador
apellido2	String	Segundo apellido del sanitario o administrador
password	String	Contraseña del sanitario o administrador
dni	String	DNI del sanitario o administrador
fechaDeAlta	Date	Fecha en la que fue dado de alta el sanitario o administrador
type	String	Tipo de usuario ('S': Sanitario, 'A': Administrador)
pacientes	Array	Array de DNIs de los pacientes de los que es responsable el sanitario o administrador
servicios	Array	Array de identificadores de los servicios de los cuales es responsable el sanitario o administrador

Tabla 3.5: Columnas de un usuario de tipo sanitario o administrador

3.5.2 servicios

Colección que almacena todos los servicios creados por un sanitario o administrador, esta tabla almacena la información que se puede ver en la tabla 3.6.

COLUMNA	TIPO	DESCRIPCIÓN
_id	ObjectId	Identificador del servicio
nombre	String	Nombre del servicio
descripcion	String	Descripción del servicio
unidades	String	Unidades en las que se tomarán las mediciones del servicio

Tabla 3.6: Columnas de un usuario de tipo sanitario o administrador

3.5.3 serviciosAsignados

En esta colección se almacenan los datos necesarios para asignar un servicio a un paciente, véase la tabla 3.7.

COLUMNA	TIPO	DESCRIPCIÓN
_id	ObjectId	Identificador del servicioAsignado
servicio	ObjectId	Identificador del servicio que se le va asignar al paciente
sanitarioResponsable	ObjectId	Identificador del sanitario que le asignó el servicio al paciente
paciente	ObjectId	Identificador del paciente al que se le ha asignado el servicio
fechaAlta	Date	Fecha en la que el servicio fue asignado a un paciente
fechaBaja	Date	Fecha en la que el servicio fue dado de baja al paciente
proximaMedicion	Date	Fecha en la que el paciente debe de tomar la siguiente medición
intervalo	String	Intervalo de tiempo en el que el paciente debe de realizar las mediciones (Ej: 6h, 1s, etc)
valorMax	Int32	Valor límite en el que si una medición lo sobrepasa, se genera una alarma
activo	Boolean	Indica si el servicio asignado esta activo o no
observaciones	String	Indicaciones del doctor, que tiene que tener en cuenta el paciente a la hora de realizar mediciones
mediciones	Array	Array que almacena todas las mediciones realizadas por el paciente en el servicio. Cada medición almacena el valor de esta, la fecha en la que fue tomada y indica si al registrarla generó una alarma.

Tabla 3.7: Columnas de un 'servicioAsignado'

3.5.4 alarmas

Por último, esta colección se encarga de almacenar alarmas. Las columnas de la colección las podemos ver en la tabla 3.8.

COLUMNA	TIPO	DESCRIPCIÓN
_id	ObjectId	Identificador de la alarma
servicioAsignado	ObjectId	Identificador del 'servicioAsignado' donde se ha generado la alarma
fechaGenerada	Date	Fecha de cuando se generó la alarma
fechaGestionada	Date	Fecha de cuando un sanitario o administrador dio de baja la alarma (si está activa, este campo es null)

Tabla 3.8: Columnas de un 'servicioAsignado'

4 Resultados

4.1 Login

Lo primero que se encuentra el usuario al entrar a la aplicación web es la pantalla del login, véase la figura 4.1, en esta pantalla el usuario tiene que introducir sus credenciales y seleccionar si quiere acceder como sanitario o no (el administrador también tiene que marcar el 'slider' para acceder).

Se tiene que marcar la opción de acceso ya que un sanitario o administrador puede ser dado de alta como paciente.

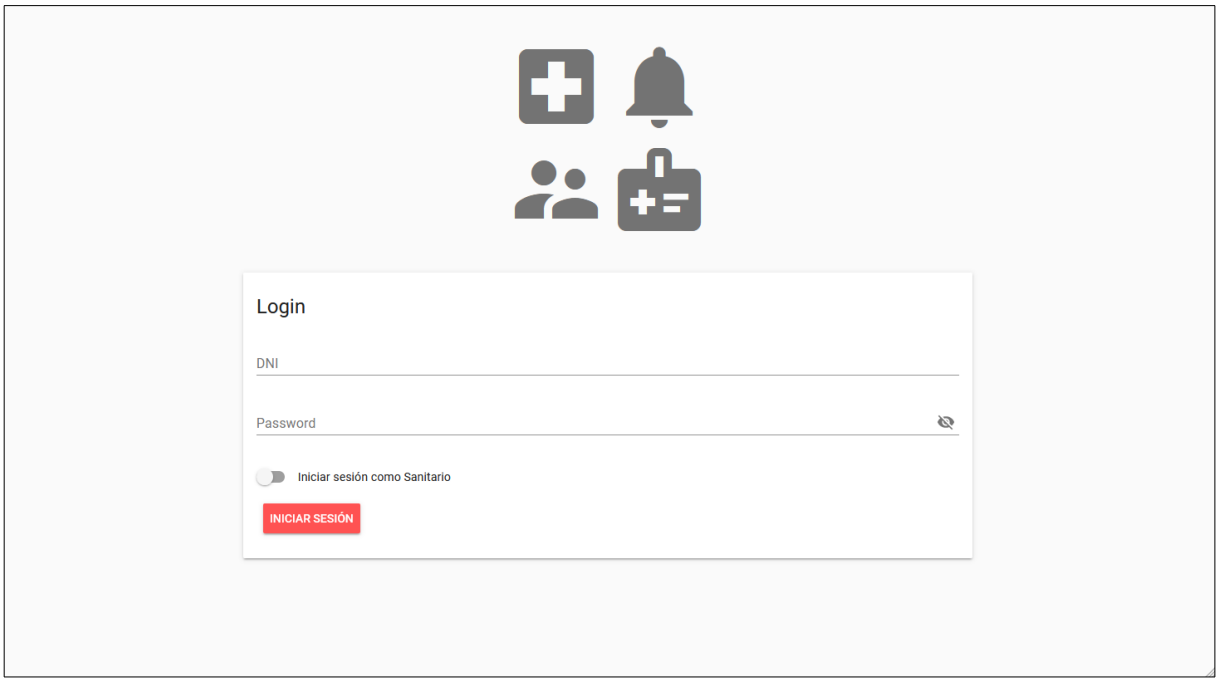

Figura 4.1: Pantalla: Login

4.2 Intranet del sanitario

Una vez accedido el usuario a la intranet del sanitario, se le mostrará la pantalla del 'Home' del sanitario, como se puede ver en la figura 4.2, en la que se le da la bienvenida a la aplicación.

En esta pantalla se puede observar una cabecera en la que se muestra el dni del sanitario y el modo en el que ha accedido a la aplicación, en este caso como sanitario. También podrá observar un botón en la parte derecha de la cabecera para poder cerrar la sesión.

Por otra parte, se muestra un menú lateral el cual puede ser ocultado. Este menú esta separado en dos partes por una línea horizontal, la parte superior de este menú esta formado por distintas secciones de la aplicación, las cuales son: 'Información del usuario', 'Alarmas', 'Pacientes' y 'Creación de servicios'. Y en la parte inferior del menú, se mostrará el listado de servicios que ha creado y por lo tanto es responsable el sanitario.

Destacar que en la parte superior del menú, en la sección de alarmas se verá, en caso de que existan alarmas activas, un contador con el número de estas.

Figura 4.2: Pantalla: Home del sanitario

Si el sanitario accede a la pantalla de 'Información del usuario', tal y como se puede ver en figura 4.3, puede ver sus datos personales y por otra parte puede modificar la contraseña del usuario con el que ha accedido.

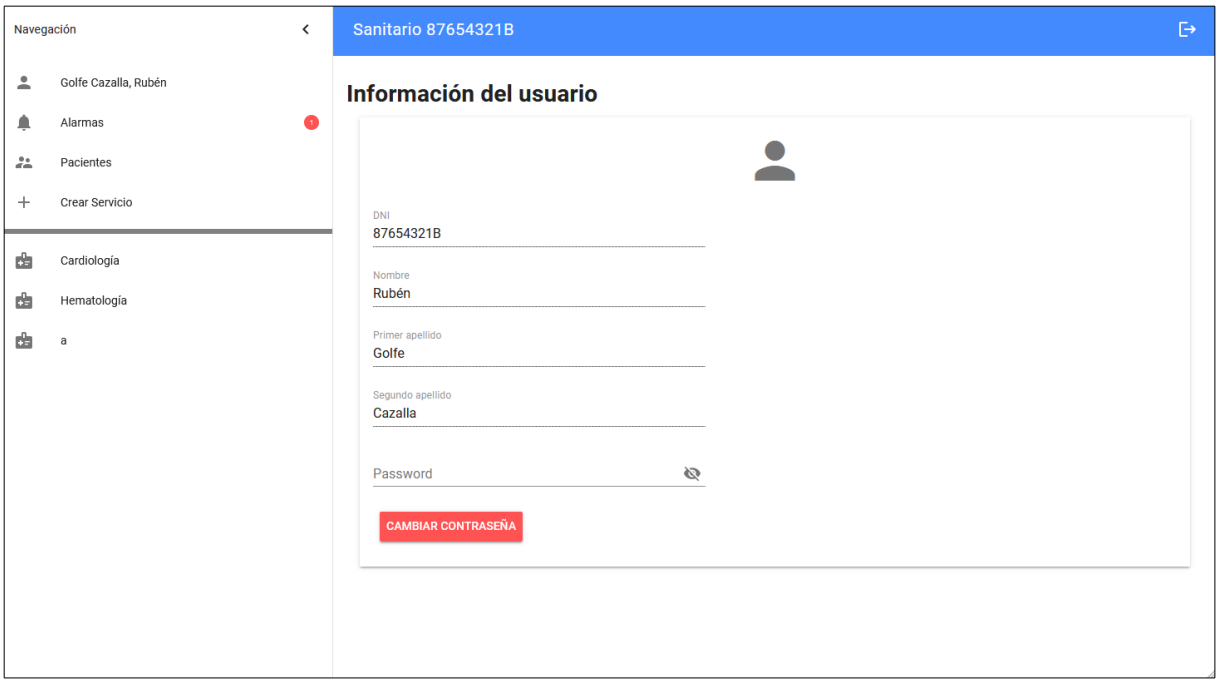

Figura 4.3: Pantalla: Información del usuario

Cuando el sanitario accede a la pantalla de alarmas, se muestra, como se puede ver en figura 4.4, el listado de alarmas activas que hay. En este listado, el sanitario puede observar el servicio de donde se ha generado la alarma, el paciente que ha generado la alarma, y la fecha y hora en la que se generó la alarma.

Por otra parte, aunque el listado está ordenado por defecto por la fecha y hora de generación de la alarma de manera descendente (para que al sanitario le aparezcan en primer lugar las alarmas que llevan más tiempo sin ser atendidas), este listado se puede ordenar, de manera ascendente y descendente, por cada uno de sus campos pulsando en el nombre de la columna.

El sanitario puede dar de baja alarmas, seleccionando primero la alarma que quiere dar de baja y pulsando el botón de 'Gestionar alarma'.

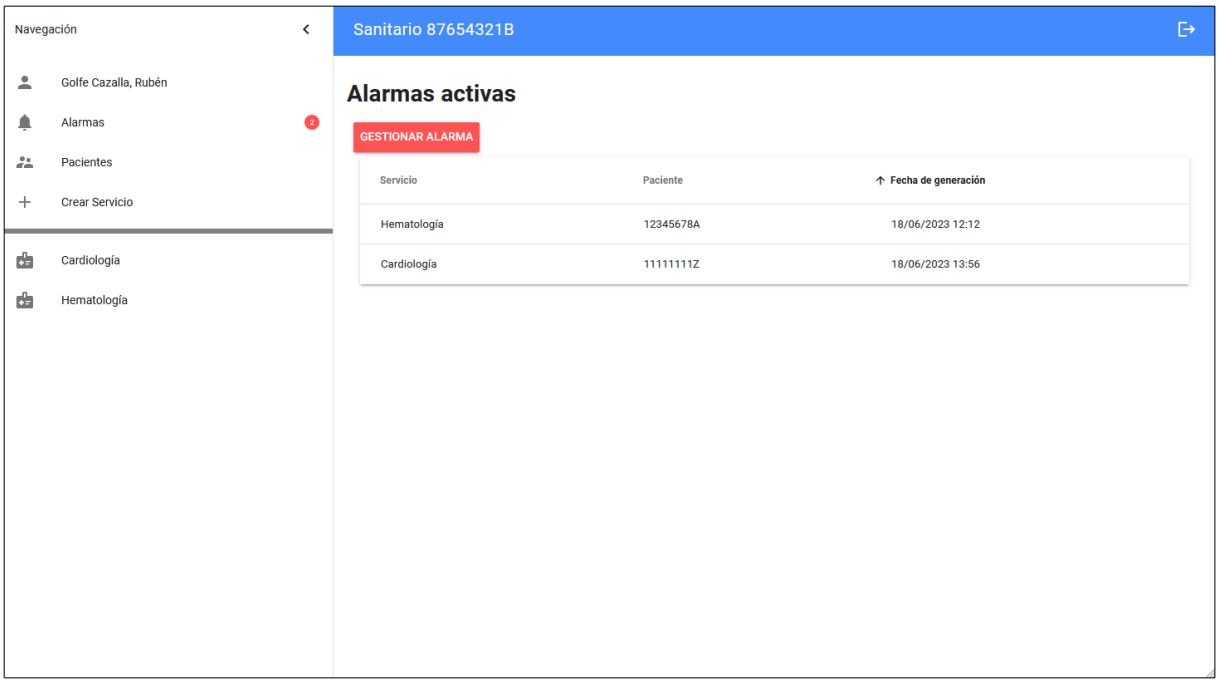

Figura 4.4: Pantalla: Alarmas

Cuando el sanitario accede a la sección de 'Pacientes', se le mostrará el listado de pacientes de los cuales es o ha sido responsable, véase la figura 4.5. En esta lista se puede ver el dni, el nombre, los apellidos y la fecha de alta en la que fue registrado el paciente en el sistema.

Esta lista a parte de poder ser ordenada de manera ascendente o descendente por cada una de sus columnas, también se puede filtrar por el dni del paciente introduciendolo en el campo de texto, la lista se va filtrando a medida que se va escribiendo en el campo de texto.

Desde esta pantalla, el sanitario puede registrar a pacientes en el sistema pulsando en el botón de 'Registro de pacientes'.

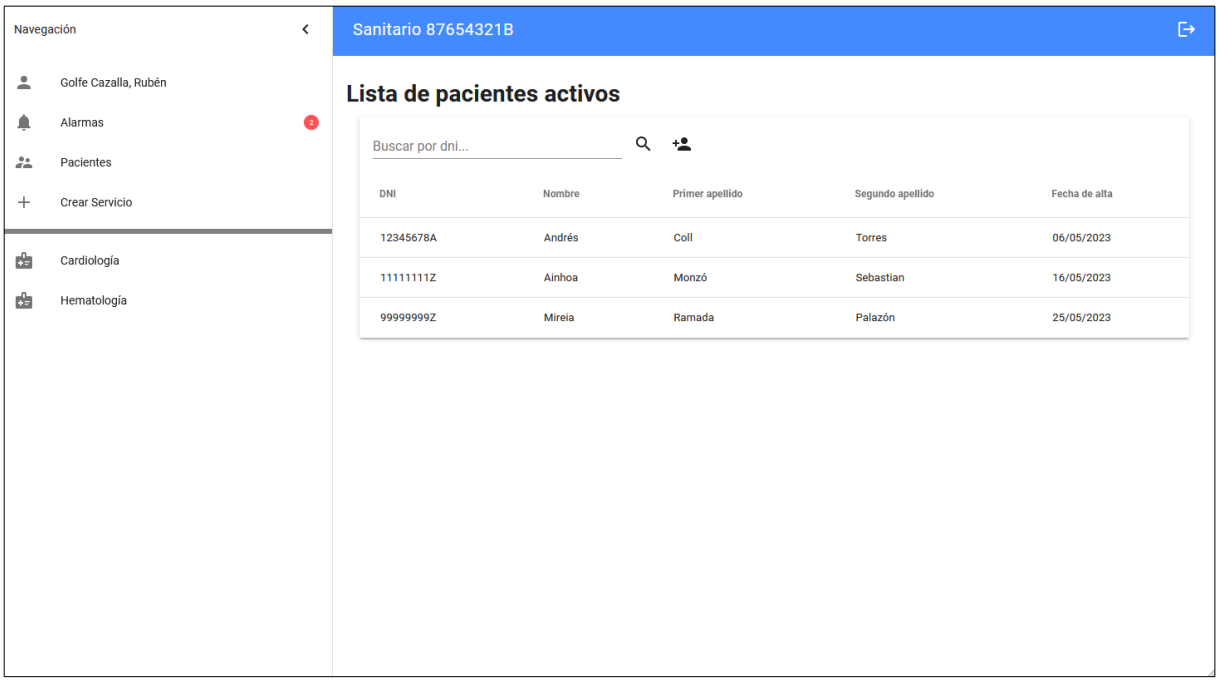

Figura 4.5: Pantalla: Pacientes

Si el sanitario pulsa sobre el botón de 'Registro de pacientes', se le abrirá una ventana flotante o 'PopUp' con un formulario para poder dar de alta pacientes en la aplicación.

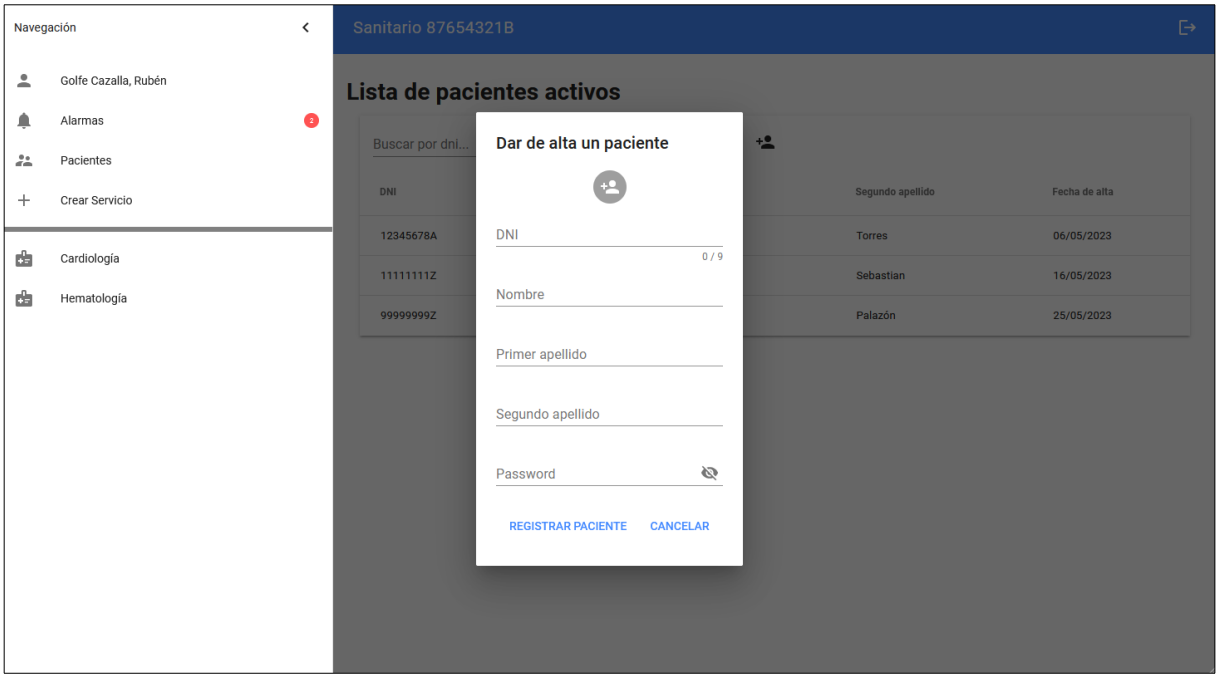

Figura 4.6: PopUp: Alta de pacientes

Si el paciente se ha podido registrar correctamente, se mostrará otra ventana flotante de confirmación, véase la figura 4.7. Esto se realiza para que el sanitario tenga *feedback* de la operación que acaba de realizar.

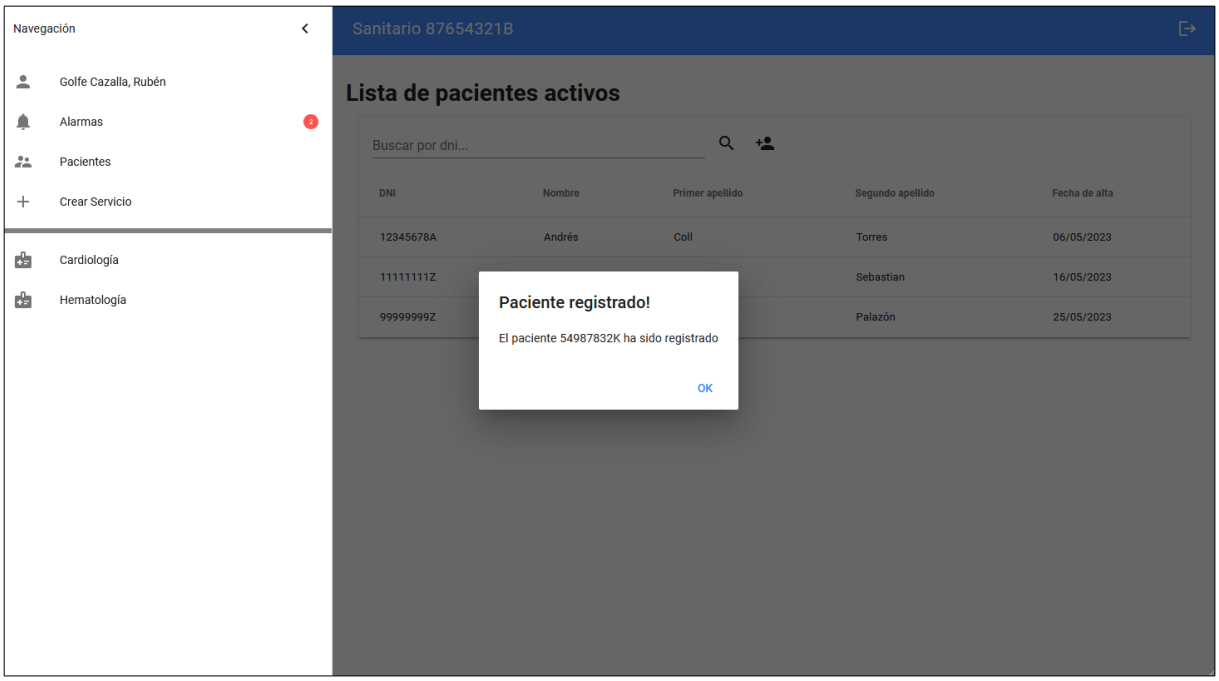

Figura 4.7: PopUp: Confirmación del registro de un paciente

Estando en la pantalla de 'Pacientes' (figura 4.5), si el sanitario selecciona a un paciente de la lista, accede a la pantalla del 'Histórico del paciente'.

En esta pantalla, tal y como se puede ver en figura 4.8 se ve la información personal del paciente seleccionado, y por otra parte un listado de los servicios que se le han asignado activos e inactivos.

En la lista de servicios asignados del paciente, el sanitario puede ver el nombre del servicio que le han asignado, el dni responsable de ese servicio, la fecha en la que se le asignó ese servicio al paciente y la fecha de baja del servicio. En caso de que un servicio asignado no tenga fecha de baja, significa que el servicio sigue activo para el paciente.

Esta lista puede ser filtrada por el nombre del servicio, y por otra parte puede ser orednada de manera ascendente y descendente por cada una de sus columnas, aunque por defecto la lista está ordenada por la fecha de alta para que el sanitario pueda ver al principio de la lista los servicios más recientes.

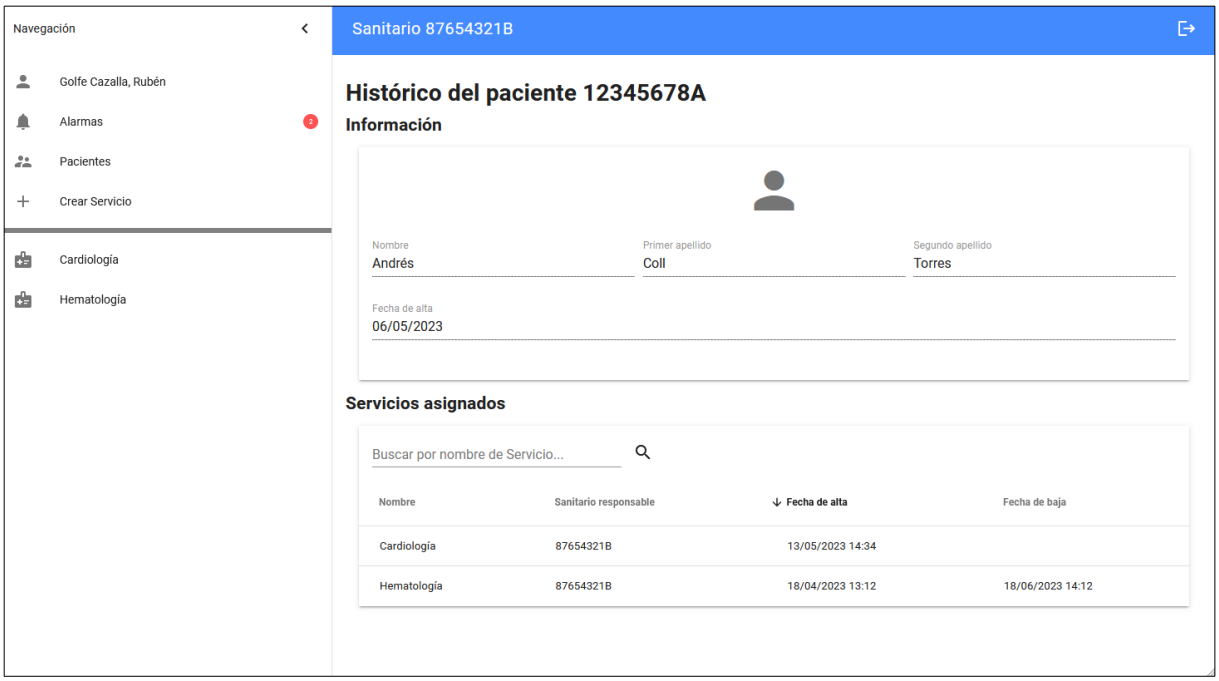

Figura 4.8: Pantalla: Histórico del paciente

Si el sanitario pulsa sobre uno de los servicios del paciente, accede a la pantalla de 'Información del servicio asignado', véase figura 4.9.

En esta pantalla, el sanitario puede ver información acerca del servicio como su nombre y la descripción. También puede ver información del servicio asignado como la fecha en la que se le fue asignado el servicio al paciente, la fecha de baja en caso de que el servicio esté inactivo, el intervalo de tiempo en el que el paciente tiene que realizar mediciones y el valor máximo que estableció el sanitario responsable, el cual si se supera se generará una alarma.

También se muestra el listado de mediciones que ha ido registrando el paciente en el que se puede ver el valor de la medición, la fecha en la que fue registrada la medición y un campo que indica si el registro de esta medición generó una alarma o no. Este listado está ordenado por defecto por la fecha de registro de la medición (para que aparezcan en primer lugar las últimas mediciones tomadas por el paciente), pero también puede ser ordenado de manera ascendente o descendente por cualquiera de sus otras columnas.

En caso de que el servicio asignado este activo y el responsable de este sea el sanitario que ha accedido a la aplicación, se habilita el botón 'Dar de baja', el cual como su propio nombre indica da de baja a ese paciente del servicio asignado seleccionado.

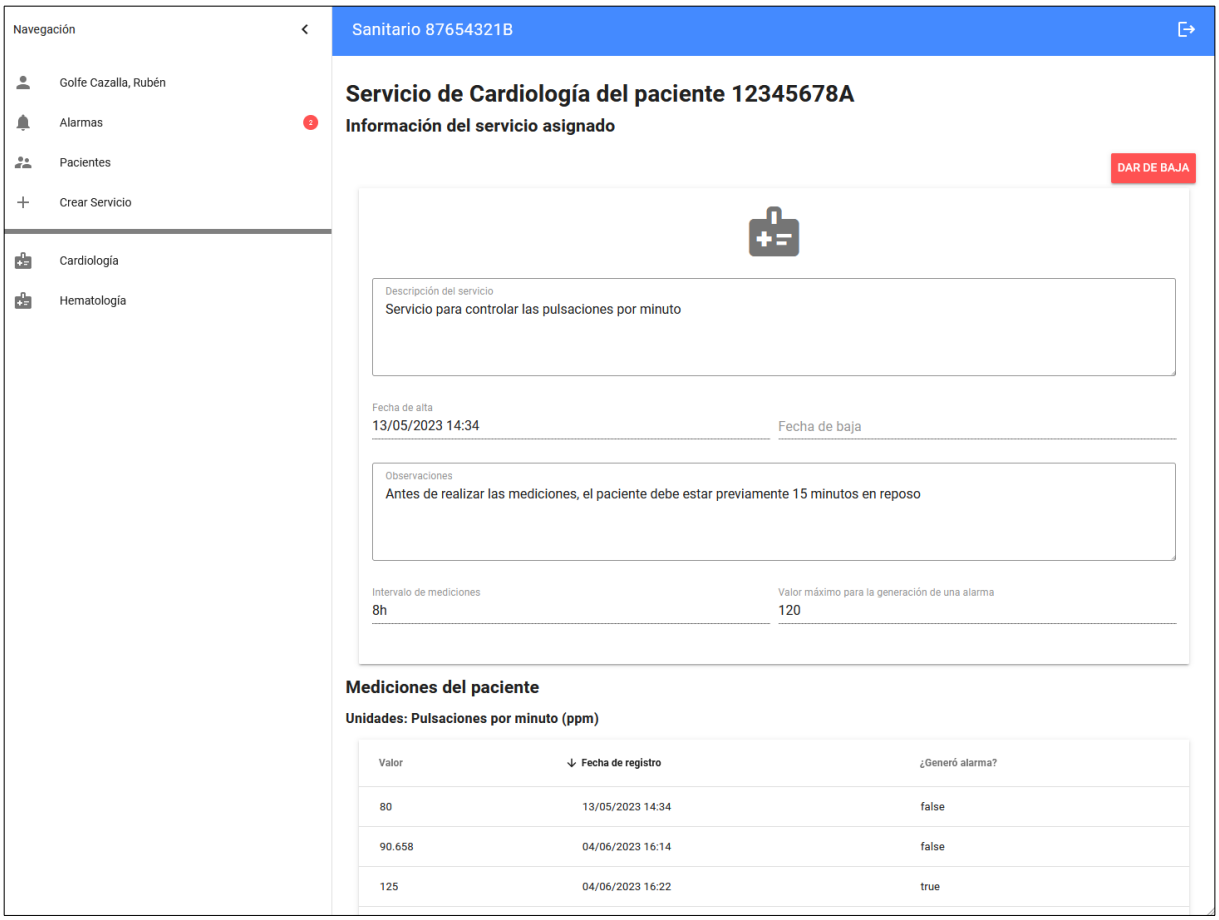

Figura 4.9: Pantalla: Información del servicio asignado

En caso de que en el listado de mediciones no se haya registrado ninguna, por ejemplo porque al paciente se le acaba de asignar un servicio, se muestra en el listado de mediciones un mensaje informativo al sanitario, tal y como se puede ver en figura 4.10.

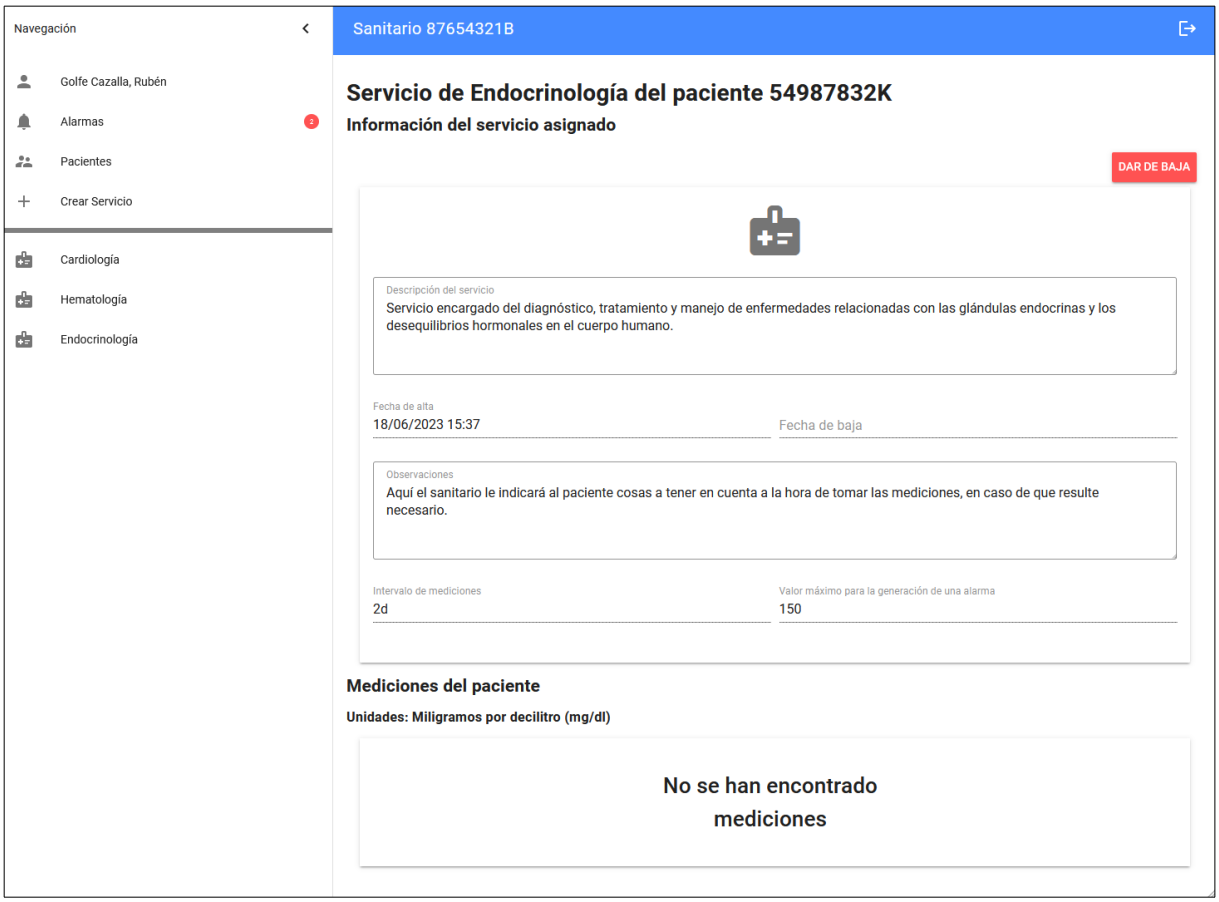

Figura 4.10: Pantalla: Información servicio asignado a paciente sin mediciones

Si el sanitario pulsa el botón de 'Dar de baja', se abre una ventana flotante, como se puede ver en la figura 4.11, para advertirle al usuario de si esta seguro de realizar esa acción. Si el sanitario confirma nuevamente sobre el 'PopUp' el paciente se da de baja del servicio.

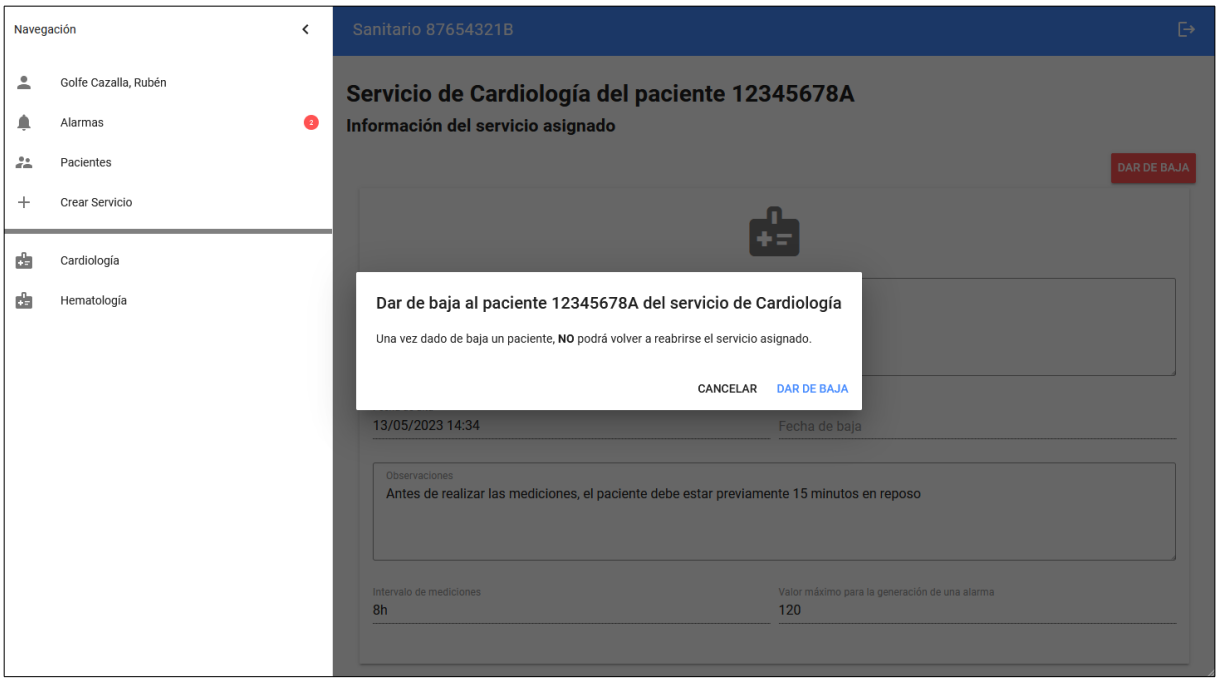

Figura 4.11: PopUp: Confirmación baja de un servicio asignado

Volviendo al menú lateral, si el sanitario selecciona la sección 'Crear Servicio', accede a la pantalla de 'Alta de servicios', tal y como se puede ver en la figura 4.12.

Esta pantalla está formada por un pequeño formulario con tres campos: el nombre del servicio, una descripción del servicio y las unidades en las que serán tomadas las mediciones de ese servicio.

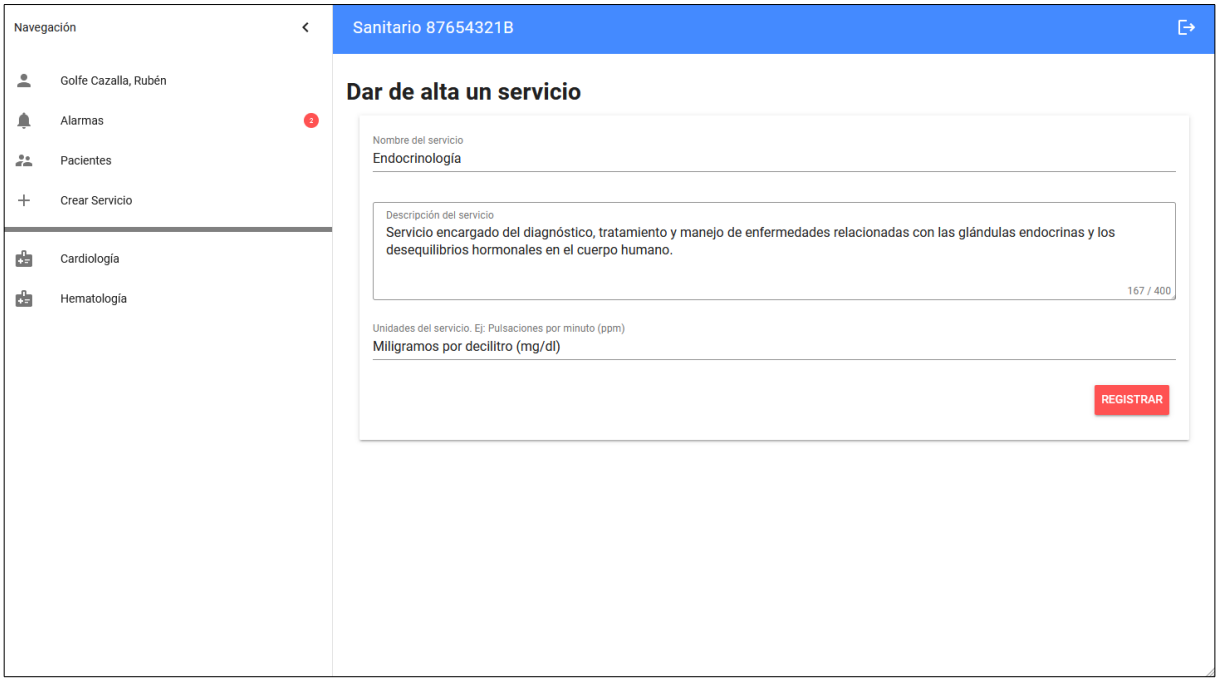

Figura 4.12: Pantalla: Alta de servicios

Una vez registrado un servicio por el sanitario, se muestra una ventana flotante en la que se confirma el registro del servicio, véase la figura 4.13.

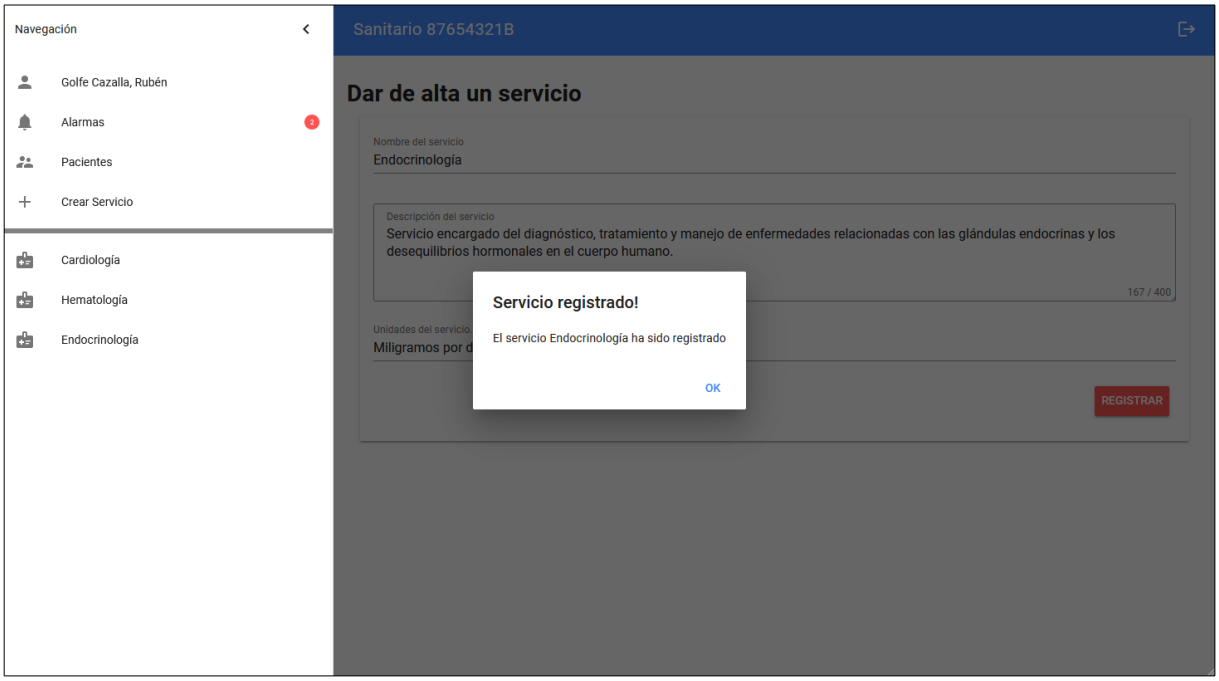

Figura 4.13: PopUp: Confirmación alta de servicios

Si el sanitario pulsa sobre cualquiera de los servicios que aparecen en la parte inferior de su menú, este accede a la pantalla de ese servicio, véase la figura 4.14.

En esta pantalla se ve la descripción del servicio seleccionado y un listado de pacientes a los que le ha asignado ese servicio, en el que el sanitario puede ver el dni, nombre, apellidos y la fecha de alta en la que se le asignó a ese paciente el servicio.

El sanitario puede buscar un paciente por dni, y puede ordenar la lista de pacientes de forma ascendente y descendente por cada una de las columnas, al pulsar sobre el nombre de estas.

En caso de que no hayan pacientes se mostrará un mensaje en el listado, tal y como aparece en la figura 4.14.

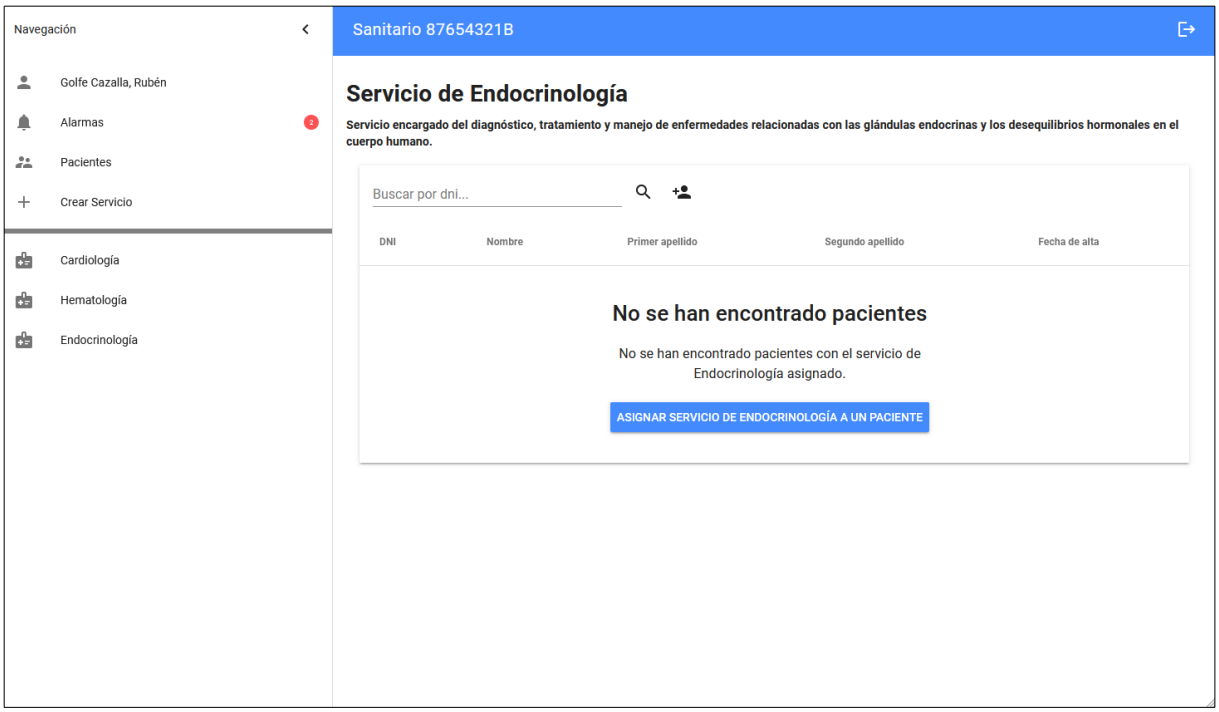

Figura 4.14: Pantalla: Servicio

Si el sanitario pulsa sobre el botón de asignar servicio a paciente que lo puede encontrar en la parte derecha del buscador del dni o en medio de la tabla en caso de que esta esté vacía, se abre la ventana flotante 'Asignación de servicio a paciente', tal y como aparece en la figura 4.15.

En este 'PoUp' el sanitario se encuentra con un formulario a rellenar, el cual está formado por el dni del paciente al que se le va a asignar el servicio, el intervalo de tiempo en el que el paciente deberá realizar las mediciones, en este se introduce el valor y la unidad de tiempo que seleccionará a través de un desplegable en el cual están los valores 'Horas', 'Días' y 'Semanas'. El sanitario también tiene que especificar el valor máximo que se va a permitir como medición, y por lo tanto si el paciente sobrepasa, se generará automáticamente una alarma.

Por último, también especificará unas indicaciones para realizar las medidas correctamente al paciente, estas aparecen a la hora de realizar la medición en la parte del paciente.

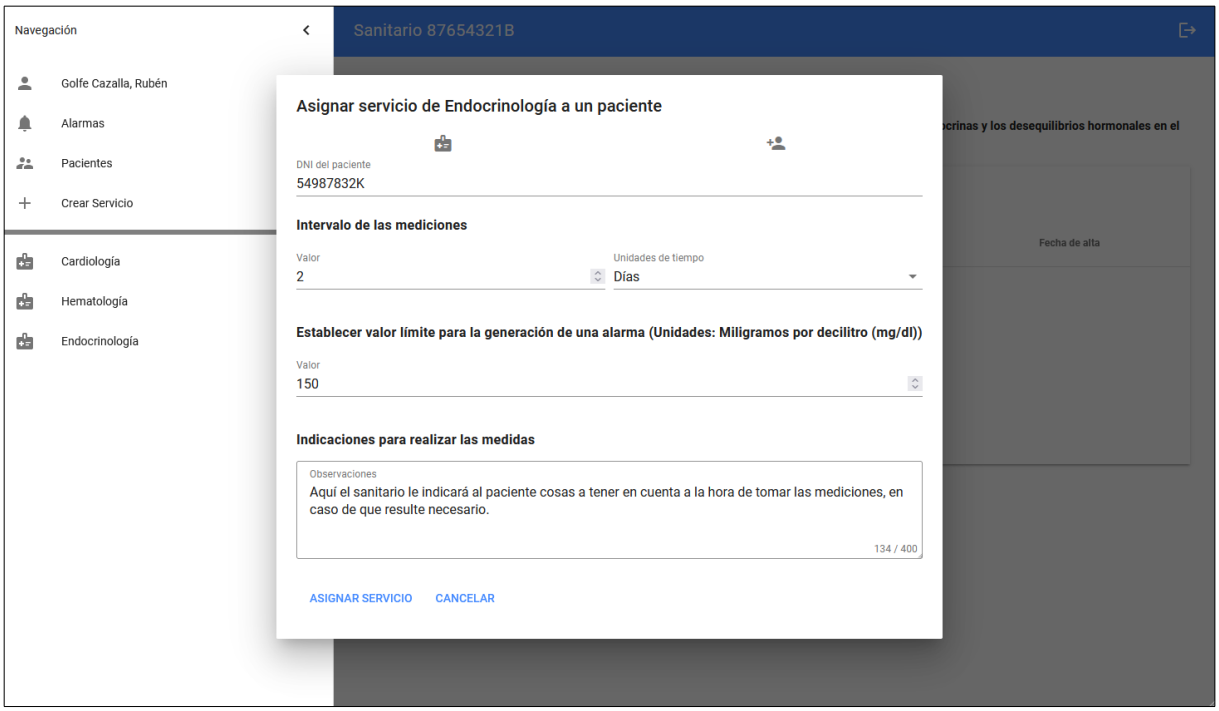

Figura 4.15: PopUp: Asignación de servicio a paciente

Si el servicio se ha asignado correctamente, se muestra un mensaje de confirmación como se puede ver en la figura 4.11.

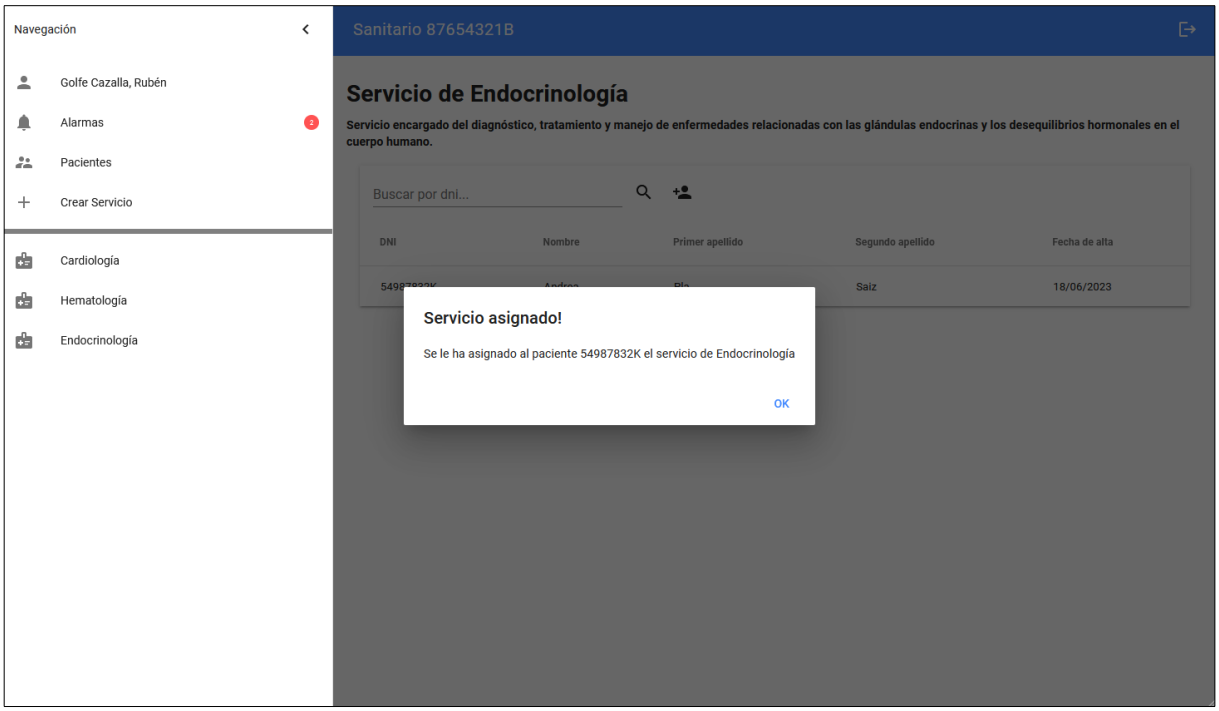

Figura 4.16: PopUp: Confirmación servicio asignado a paciente

4.3 Intranet del administrador

Como podemos comprobar en la figura 4.17, la pantalla de 'Home' del administrador es exactamente la misma que la del sanitario, con la única diferencia de que tiene habilitada la sección 'Registrar pacientes' en el menú lateral y que en la cabecera se muestra 'Administrador/Sanitario X'.

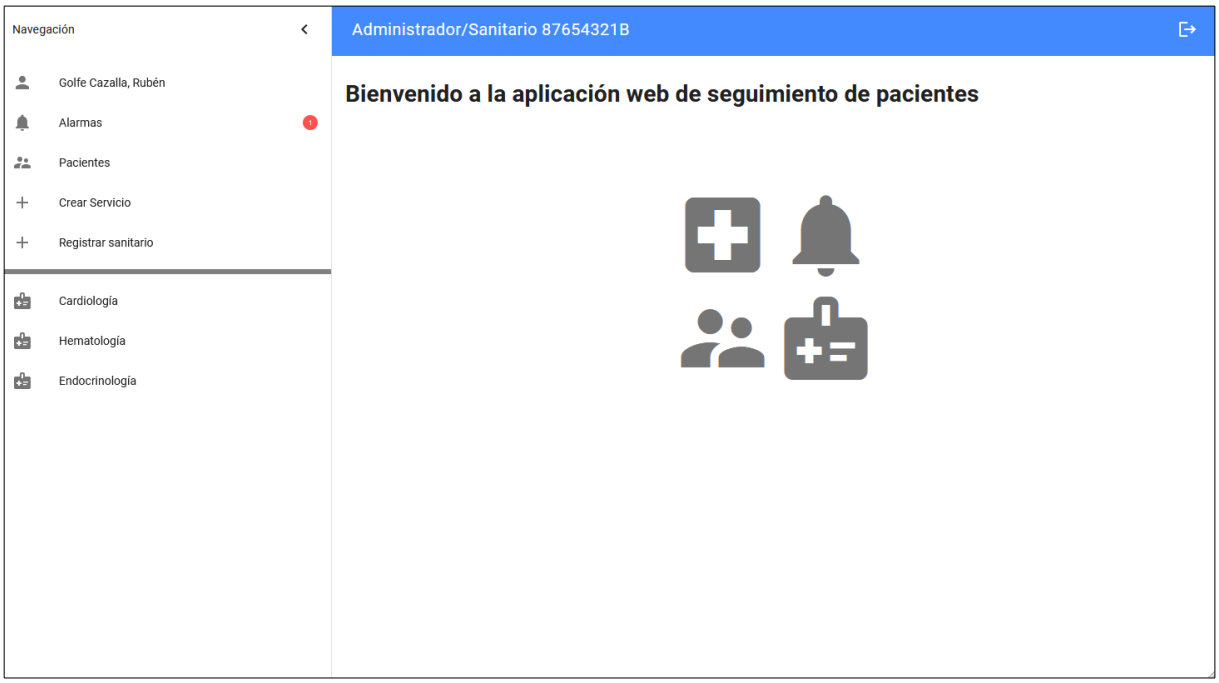

Figura 4.17: Pantalla: Home del administrador

Si el administrador pulsa sobre este botón, se abre un 'PopUp' con un formulario para dar de alta nuevos sanitarios, véase la figura 4.18

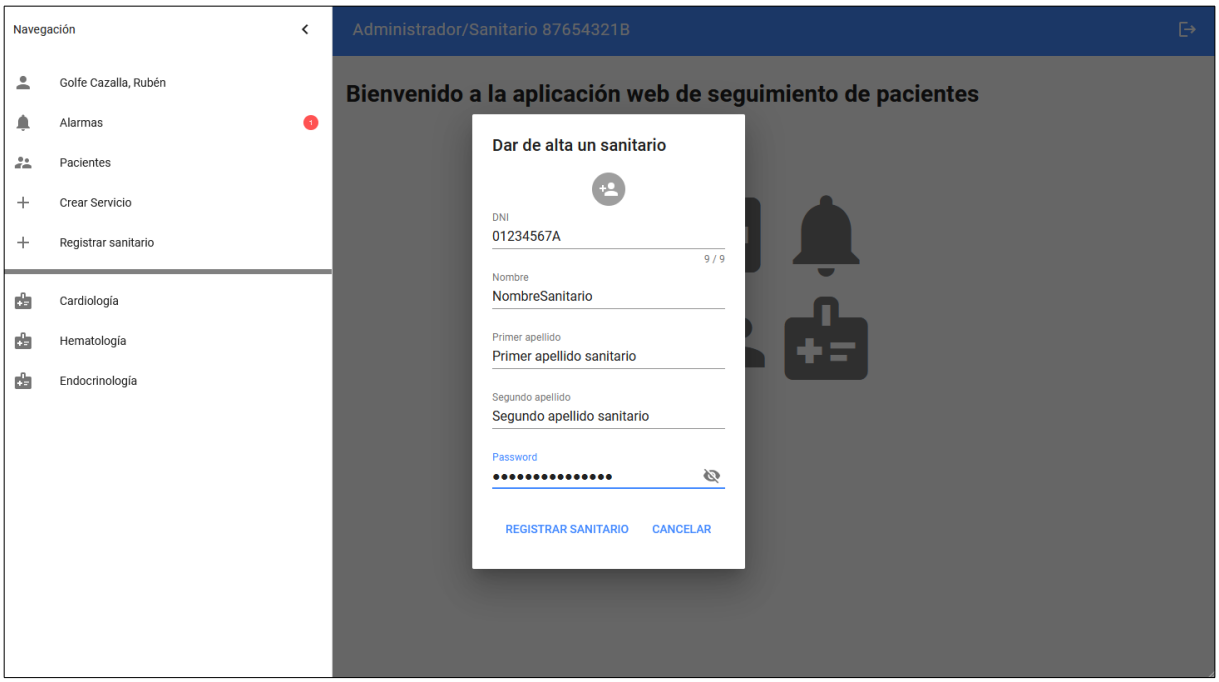

Figura 4.18: PopUp: Registro de sanitarios

Cuando el administrador registre correctamente un sanitario, se muestrá otro 'PopUp' de confirmación, véase la figura 4.19

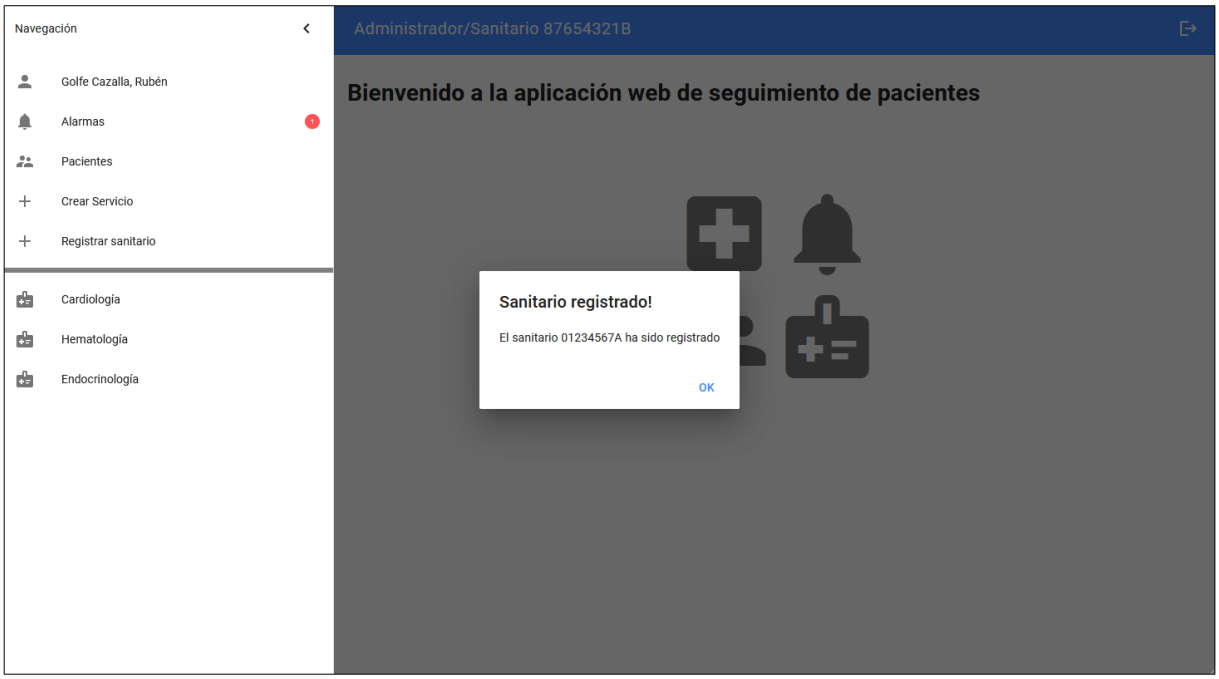

Figura 4.19: PopUp: Confirmación de sanitario registrado

4.4 Intranet del paciente

Si el usuario ha iniciado sesión como paciente, accederá a la pantalla del 'Home' del paciente, véase la figura 4.20.

El paciente puede ver una cabecera al igual que el sanitario, en la que se le indica en que modo ha iniciado sesión, y en la parte derecha el botón para cerrar la sesión.

Por otra parte, se muestra un menu lateral el cual esta formado por dos partes. En la parte superior tiene disponible la sección de 'Información del usuario' y en la parte inferior tiene disponible los diferentes servicios los cuales se le han asignado.

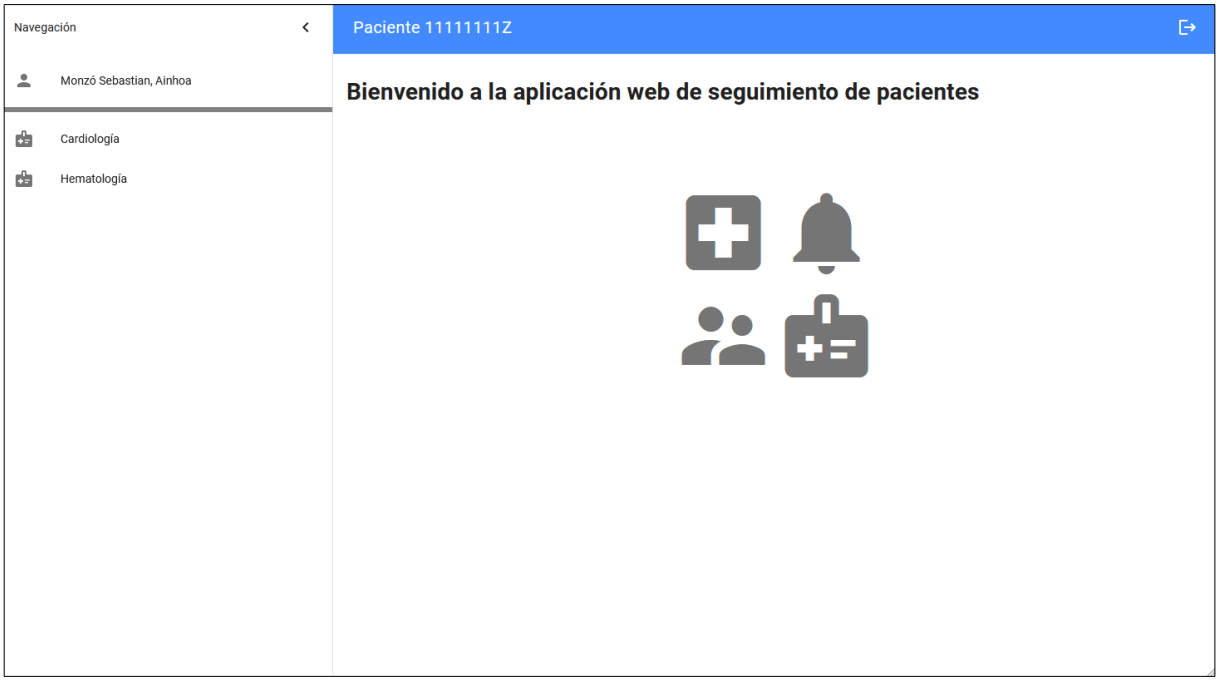

Figura 4.20: Pantalla: Home del paciente

Si el paciente accede a la pantalla de información del usuario, tal y como se puede ver en figura 4.21, este puede ver su información personal y puede modificar su contraseña la cual le asignó su doctor.

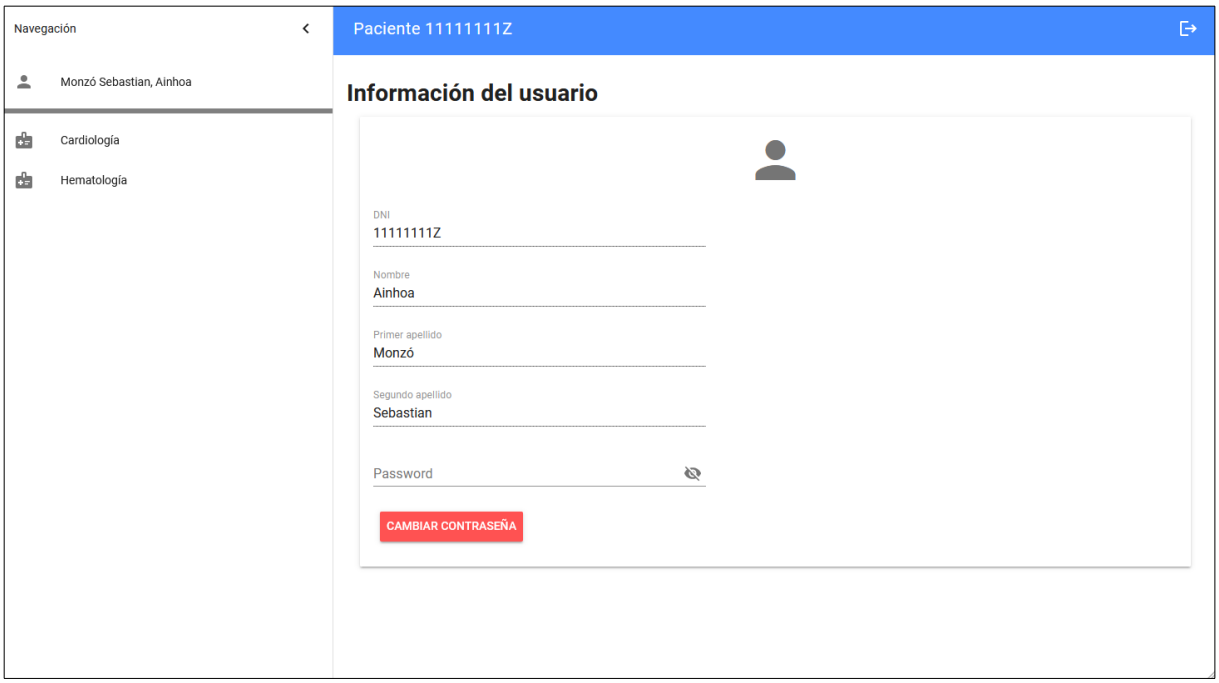

Figura 4.21: Pantalla: Información del usuario

Por otra parte, cuando el paciente pulsa sobre uno de los servicios que se le ha asignado, accede a la pantalla de 'Servicio', véase la figura 4.22.

En esta pantalla el paciente puede ver información del servicio que se le ha asignado, esta es la descripción del servicio, la fecha de alta y de baja del servicio, observaciones o indicaciones que la haya puesto el sanitario, para que tenga en cuenta el paciente antes de tomar la medición. También puede ver el dia y la hora en la que tiene que realizar la proxima medición y el intervalo de tiempo que tiene que pasar entre mediciones.

En la parte inferior de la pantalla esta el listado de mediciones registradas por el paciente, en la que el paciente puede ver el valor de la medición y la fecha y hora de cuando se registró la medición.

El botón de registro de mediciones está en la parte superior de la lista de mediciones, este botón sólo está disponible si al paciente le toca realizar la medición, en otro caso el botón esta deshabilitado.

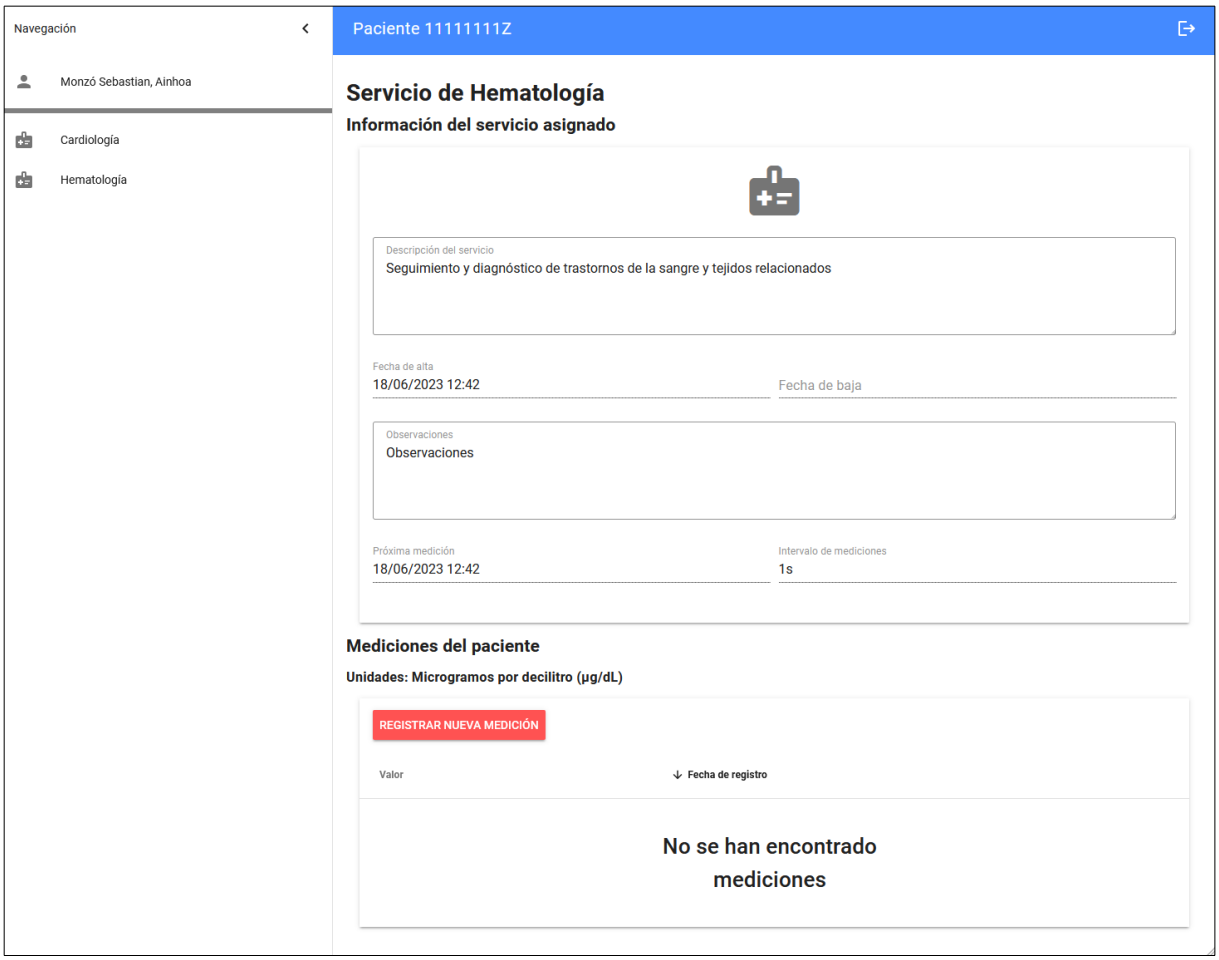

Figura 4.22: Pantalla: Servicio del paciente

Si el paciente pulsa el botón de 'Registrar nueva medición' se abre una ventana flotante, como se puede ver en la figura 4.23.

En este 'PopUp' el paciente sólo tiene que indicar el valor de la medición en las unidades que le haya especificado el sanitario.

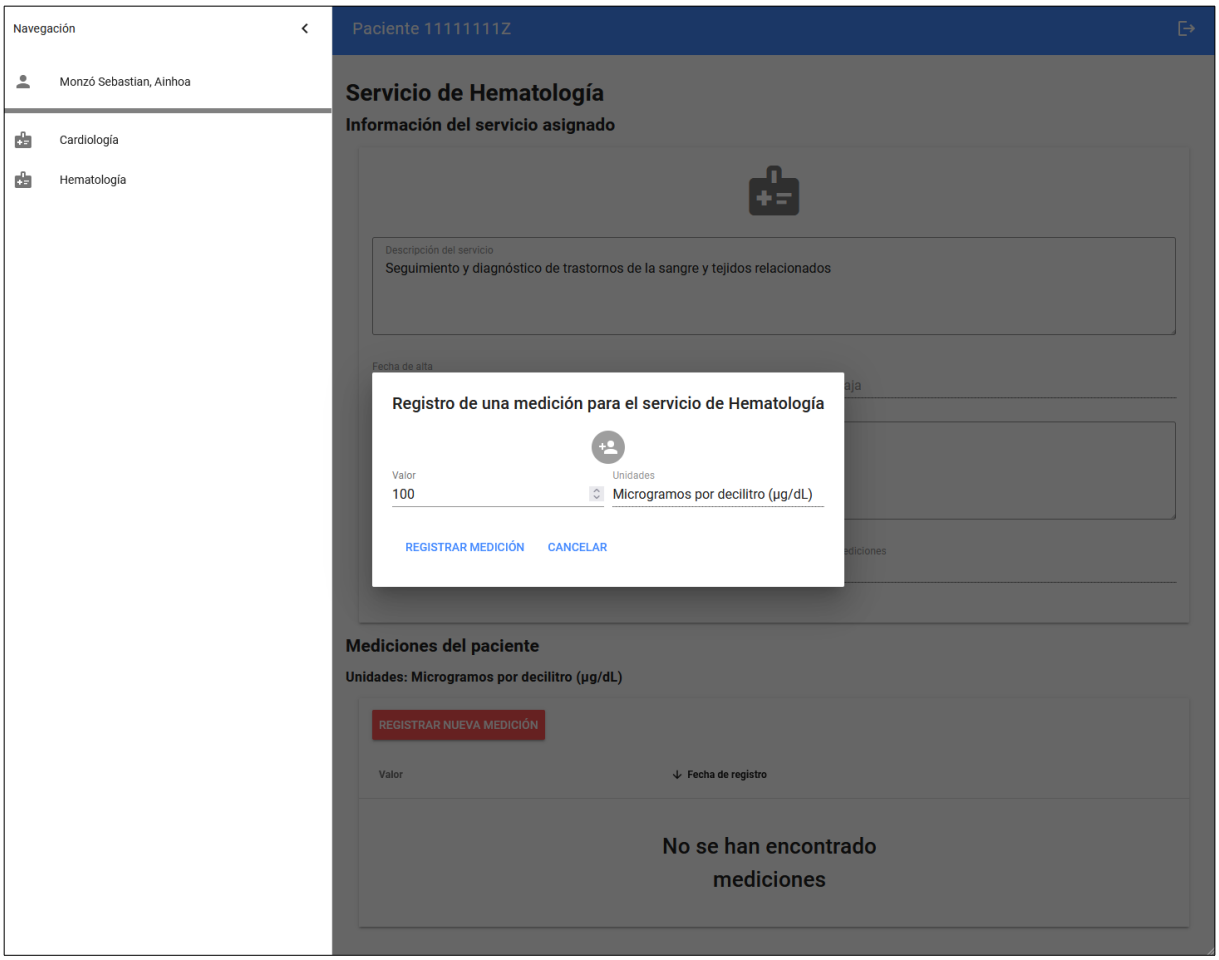

Figura 4.23: PopUp: Registro de mediciones

Si la medición se ha registrado correctamente, se le informa al paciente mostrando por pantalla la ventana flotante de la figura 4.24.

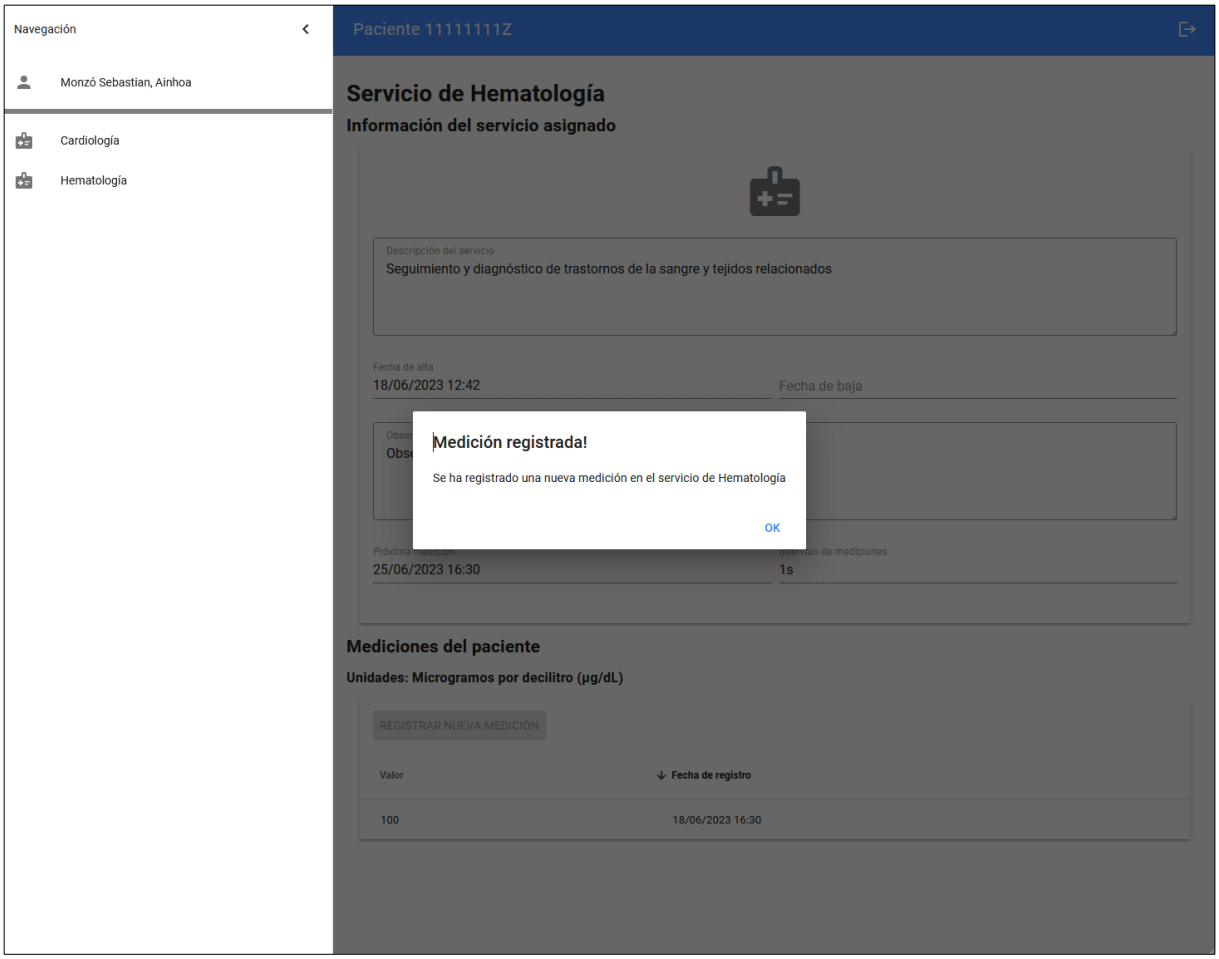

Figura 4.24: PopUp: Confirmación del registro de mediciones

5 Conclusiones y trabajos futuros

Con este proyecto de final de carrera he conseguido aprender los principios de diferentes tecnologías, que hasta antes de empezar con este proyecto no había tocado. Había oído hablar de varias de ellas, pero no les había metido mano a estas como en este proyecto, con esto me ha gustado saber que no importa que tecnologías nuevas vayan surgiendo en un futuro, ya que invirtiendo tiempo he sido capaz de desenvolverme con ellas (obviamente sin considerarme alguien que sabe mucho de estas, ya que para esto se necesita mucho más).

Por otra parte, con este proyecto me he dado cuenta de que el tiempo necesario para realizar un proyecto grande es mucho más de lo que esperaba. Aunque la aplicación web y el servicio RESTful desarrollados en este proyecto son funcionales, se han quedado ciertas cosas por implementar y muchas para mejorar, como por ejemplo dar una mejor gestión de los sanitarios a los administradores, crear una paginación en las listas de datos, mantener la sesión al refrescar la página, implementar validadores en los formularios, mejoras en la parte visual, y un largo etc.

Voy a seguir invirtiendo tiempo en este proyecto para dejarlo lo mejor posible, ya que me han resultado bastante interesantes muchas de las tecnologías utilizadas. Por lo que en un futuro bastante cercano, voy a seguir trabajando en este proyecto para llegar a corregir y mejorar a parte de lo mencionado anteriormente, las nuevas problemáticas que vayan surgiendo, ya que en este tipo de proyectos siempre surgen.

Por último, aunque voy a seguir trabajando en este proyecto, como he dicho antes, también he querido subir este proyecto de manera pública a mi github personal, para que cualquier usuario con más experiencia (que no será difícil) o con nuevas ideas, pueda también ampliar este proyecto.

Bibliografía

AngularJS Team. (2010). *AngularJS*. https://angular.io/. (Vid. pág. 24)

Brendan Eich. (1995). *JavaScript*. https://developer.mozilla.org/en-US/docs/Web/javascript. (Vid. pág. 23)

Douglas Wilson. (2014). *Body-Parser*. https://www.npmjs.com/package/body-parser. (Vid. pág. 26)

Douglas Wilson y Troy Goode. (2013). *Body-Parser*. https://www.npmjs.com/package/cors. (Vid. pág. 27)

Dwight Merriman y Eliot Horowitz. (2009). *Mongo DB*. https://www.mongodb.com/es. (Vid. pág. 27)

Eduardo San Martin Morote. (2014). *Vue Router*. https://router.vuejs.org/. (Vid. pág. 25)

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. University of California, Irvine. (Vid. pág. 43).

GeeksForGeeks. (2022). *Client-Server Model*. https://www.geeksforgeeks.org/client-server-model/. (Vid. pág. 19)

Google Team. (2014). *Material Design*. https://m3.material.io/. (Vid. pág. 24)

Isaac Z. Schlueter. (2010). *Node Package Manager*. https://www.npmjs.com/. (Vid. pág. 23)

Jordan Walke. (2011). *React*. https://react.dev/. (Vid. pág. 24)

Marcos Moura. (2015). *Vue Material*. https://www.creative-tim.com/vuematerial/. (Vid. págs. 24, 33)

Matt Zabriskie. (2014). *Axios*. https://axios-http.com/. (Vid. pág. 26)

OpenJS Foundation. (2017). *Express*. https://expressjs.com/. (Vid. pág. 26)

Shahin, T. (2017). Comparison between SPA and MPA: Competition to get the best ranking on SEO [pg. 16-17]. (Vid. pág. 24).

Visual Studio Code Team. (2023). *Documentación de Visual Studio Code*. https://code.visualstudio.com/docs. (Vid. pág. 23)

Vue.js Team. (2017). *Documentación de Vue.js 2.x*. https://v2.vuejs.org/. (Vid. pág. 24)