Diseño de un Sistema Web para la Gestión del Servicio de Camilleros para El Hospital Universitario de Santander (HUS)

María Paula Riveros Gómez

Proyecto de Grado realizado en la modalidad de práctica empresarial como requisito para optar al título de Ingeniera de Sistemas e Informática

Director

Gabriel Rodrigo Pedraza Ferreira, PhD

Doctor en Ciencias de la Computación

Tutor

Ingeniero Ever Ernesto Barrera Vargas

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2021

Dedicatoria

A Dios, por brindarme la sabiduría, la fuerza y la paciencia para poder llegar a este punto, tomando las mejores decisiones y superando los obstáculos, permitiéndome crecer a nivel profesional y personal. Por no abandonarme nunca.

A mis padres, por su apoyo y sus esfuerzos incansables para darme siempre lo mejor. Por su paciencia y amor. Por su confianza en mí y luchar conmigo para lograr esta meta. Por ellos y para ellos.

“Siempre parece imposible hasta que se hace”

Nelson Mandela

Agradecimientos

A Dios por todas sus bendiciones, por ser mi guía y fortaleza en mi vida, permitiéndome superar cada momento de dificultad y poder ser cada vez mejor, porque me ha permitido tener cerca, y bien, a mis seres queridos.

A mis padres por su amor incondicional, por el inmenso apoyo que me han dado y creer en mí. Gracias por absolutamente todo.

A mi hermano por llenarme de alegría, por entenderme y brindarme los mejores consejos, por su compañía.

A mis amigos por cada momento; por las largas horas de estudio y de trabajo en equipo, por su compañía y perseverancia para obtener los mejores resultados. Por la ayuda que me brindaron cuando no entendía algo, por su paciencia y sus explicaciones. Por permanecer.

Tabla de contenido

Lista de Tablas

Lista de Figuras

Lista de Apéndices

Resumen

Título: Diseño de un sistema web para la gestión del servicio de camilleros para el Hospital Universitario de Santander (HUS) [*]

Autor: María Paula Riveros Gómez [**]

Palabras Clave: Asignación, consulta, gestión, servicio de camilleros, sistema web, solicitud.

Descripción

El servicio de camilleros de un hospital se ocupa del traslado del personal en situación de enfermedad y/o discapacidad desde algunos servicios a otros; este constituye una tarea de gran importancia ya que así se pueden garantizar los procesos de cuidados adecuados para la salud. Para el Hospital Universitario de Santander (HUS) es de vital importancia contar con una buena gestión de estos servicios, por tal razón existe una central de camilleros en la que se lleva a cabo los procesos de solicitud y asignación de dichos servicios. Sin embargo, en ocasiones estas actividades de solicitud y asignación se llegan a retrasar debido a que no se tiene una buena organización respecto a la ocupación de los camilleros en el momento. Además, no es posible considerar los tiempos a ajustar para que los pacientes sean conducidos a los destinos en forma oportuna, ya que no se cuenta con la información necesaria. En este proyecto se propone desarrollar una herramienta, en este caso un prototipo de sistema web, que permita llevar el control de las asignaciones y solicitudes para los camilleros en el HUS, permitiendo a su vez consultar los tiempos de respuesta de los servicios para así, si es necesario, hacer mejoras posteriormente.

---

[*] Proyecto de grado en la modalidad de práctica empresarial

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Gabriel Rodrigo Pedraza Ferreira, PhD en Ciencias de la Computación. Tutor: Ingeniero Ever Ernesto Barrera Vargas.

Abstract

Title: Design of a web system for the management of the stretcher service for the Santander University Hospital (HUS) [*]

Author: María Paula Riveros Gómez [**]

Keywords: Assignment, management, query, request, stretcher service, web system.

Description

The stretcher-bearers service of a hospital takes care of the transfer of personnel in a situation of illness and/or disability from some services to others; This constitutes a task of great importance since this way the appropriate health care processes can be guaranteed. For the University Hospital of Santander (HUS) it is of vital importance to have a good management of these services, for this reason there is a central of stretcher-bearers in which the processes of requesting and assigning said services are carried out. However, sometimes these application and assignment activities are delayed because there is not a good organization regarding the occupation of the stretcher-bearers at the time. In addition, it is not possible to consider the times to be adjusted so that patients are taken to their destinations in a timely manner, since the necessary information is not available. In this project it is proposed to develop a tool, in this case a prototype of a web system, that allows to control the assignments and requests for the orderlies in the HUS, allowing in turn to consult the response times of the services in order to, if it is necessary, make improvements later.

[*] Degree Project in the entrepreneurial practice modality

[**] Faculty of Physical-Mechanical Engineeries. Department of Systems Engineering and Informatics. Director: Gabriel Rodrigo Pedraza Ferreira, PhD in Computer Science. Tutor: Engineer Ever Ernesto Barrera Vargas.

Introducción

En Colombia, para los hospitales es muy importante contar con un personal de apoyo y camilleria con el fin de ofrecer servicios a toda la zona del centro médico, garantizando una atención eficiente y controlada al usuario.

En el Hospital Universitario de Santander se cuenta con una central de camilleros, en donde se hace la gestión para llevar a cabo la ejecución de los servicios que se solicitan en la institución. Actualmente se ha intentado realizar dicha gestión de la manera más rápida y ordenada posible, teniendo en cuenta que se hace todo manualmente. Sin embargo, esto ha sido ineficiente, ya que a veces resulta demorado asignar un servicio a un camillero y el tiempo en que demora un camillero en realizar algún servicio es incierto; no es posible recopilar cierta información sobre los servicios debido a la gran cantidad de estos que se solicitan en tan solo un día.

Esto deja en claro la necesidad de mejorar la gestión de los servicios de camilleros en donde se pueda solicitar, asignar y consultar servicios de manera más rápida y eficiente, obteniendo los datos necesarios para hacer mejoras posteriormente. Por tal razón, se propone desarrollar una herramienta, en este caso un sistema o software, que permita llevar el control de las asignaciones y solicitudes para los camilleros en el Hospital Universitario de Santander, por medio de una interfaz rápida, sencilla y amigable con el usuario a la hora de realizar una solicitud. Permitiendo, además, filtrar los servicios de acuerdo con algún dato que puedan tener en común ciertos servicios y sacar conclusiones respecto a los tiempos de respuesta que se muestren.

La metodología aplicada consta de cinco fases, en la cuales se hace se recopilación de la información para establecer los requerimientos y el alcance del proyecto y proceder con el diseño e implementación del sistema validando el funcionamiento de este por medio de un plan de pruebas.

A lo largo de este trabajo se presenta, en los capítulos, el desarrollo de cada una de estas fases, junto con diagramas y tablas para una mejor comprensión de este.

1. Acerca del Proyecto

1.1. Planteamiento del problema

La E.S.E Hospital Universitario de Santander, es una Institución Pública de orden Departamental, prestadora de servicios de salud de mediana y alta complejidad con estándares de Calidad, que busca mejorar continuamente sus procesos de atención y contribuir al mejoramiento de la calidad de vida de la comunidad del nororiente colombiano, mediante el trabajo de un equipo humano calificado, con apoyo tecnológico, desarrollando actividades asistenciales junto con el personal auxiliar encaminadas a promover y asegurar el cuidado integral del paciente, garantizando la adecuada prestación del servicio y cumpliendo con la misión, visión, y objetivos de la Institución.

Muchas veces se puede generar confusión y errores al momento de la asignación y control del trabajo de los camilleros, llegando a dificultar y retrasar la realización de las tareas asignadas debido a que no se tiene una buena organización respecto a la ocupación de los camilleros en el momento. Por esta razón, aparece una necesidad de seguir garantizando el compromiso del Hospital para con los pacientes, la cual requiere ser atendida y así evitar cualquier malentendido o confusión en cuenta a los camilleros, fomentando una buena organización hasta en los momentos de emergencia.

1.2. Justificación

Teniendo en cuenta la problemática mencionada anteriormente y en búsqueda de la implementación de los estándares superiores de calidad establecidos en el Sistema Único de Acreditación, se propone desarrollar una herramienta, en este caso un sistema o software, que permita llevar el control de las asignaciones y solicitudes para los camilleros en el Hospital

Universitario de Santander, por medio de una interfaz rápida, sencilla y amigable con el usuario a la hora de realizar una solicitud.

1.3. Objetivos

Objetivos General

Diseñar un sistema web para la gestión del servicio de camilleros que soporte las actividades de solicitud, asignación y consulta de tiempos de respuesta del servicio de camilleros en el Hospital Universitario de Santander.

Objetivos Específicos

- Establecer los requerimientos adecuados que debe llevar el sistema Web para la gestión de camilleros.
- Definir la arquitectura y diseñar los componentes que harán parte del sistema Web para la gestión de camilleros del Hospital Universitarios de Santander, con un soporte adecuado en cuanto a la funcionalidad, persistencia de datos y la integridad con otros sistemas.
- Construir un prototipo de aplicación web que permita ejecutar el diseño realizado.
- Validar el correcto funcionamiento y el cumplimiento de los requerimientos del prototipo realizado.

1.4. Metodología

La metodología que se usó para el desarrollo de la aplicación se compone de cinco fases:

1. **Ambientación Tecnológica:** Inicialmente se investigó el marco de referencia bajo el cual se enmarcaba el desarrollo del proyecto, y de esta manera tener un dominio total y profundo del tema, conociendo también las herramientas a disponibilidad y necesarias para llevar a cabo los objetivos del proyecto.

2. **Identificación de características del sistema web:** En esta fase se tuvo en cuenta el dominio y el alcance del proyecto, estableciendo las características, funcionalidades y limites que iba a tener el proyecto y aclarando las tecnologías que a utilizar y la arquitectura de la aplicación a desarrollar.
3. **Diseño del sistema web:** En esta fase se estableció el diseño de la aplicación web, con los componentes necesarios para que funcionará de manera óptima y eficaz, siendo a su vez amigable con el usuario.
4. **Prototipado:** En este punto se implementó el diseño del sistema web creado, para esto se desarrolló un prototipo el cual cumple de manera parcial o total las funcionalidades del sistema web.
5. **Pruebas de validación:** Era necesario conocer el límite de la aplicación y saber si cumplía con los requerimientos y funcionalidades necesarias para solucionar el problema en cuestión. De modo que se realizaron pruebas al prototipo para estos factores.

2. Marco de referencia

2.1. Marco conceptual

2.1.1. *Hospital Universitario de Santander*

La E.S.E Hospital Universitario de Santander, es una Institución Pública de orden Departamental, prestadora de servicios de salud de mediana y alta complejidad con estándares de Calidad, dedicada a contribuir al mejoramiento de la calidad de vida de la comunidad del nororiente colombiano, mediante el trabajo de un equipo humano calificado, con apoyo tecnológico, a través de un proceso administrativo transparente y el compromiso con la academia, apoyado en la investigación y generación de conocimiento.

Historia: El Hospital Ramón González Valencia, nació por la necesidad de tener en la ciudad un Hospital que diera cubrimiento en salud a todo el Nororiente Colombiano. En julio de 1973, es inaugurado por el presidente Misael Pastrana Borrero, en memoria del Ex-presidente RAMÓN GONZÁLEZ VALENCIA.

En vista de tantos conflictos laborales y la falta de recursos el Hospital Ramón González Valencia fue liquidado el 4 de febrero de 2005, para dar paso a la Empresa Social del Estado Hospital Universitario de Santander, una nueva institución creada mediante el Decreto 0025, que comenzó a operar saneada, con menos funcionarios y con más eficiencia en la atención a la comunidad no sólo de Santander, sino de departamentos vecinos.

El nacimiento de esta nueva institución significa para el Gobierno Departamental y para quienes laboran allí, la oportunidad para rescatar la Red Pública Hospitalaria y brindar una atención integral a los usuarios. El Hospital Universitario de Santander sirve como modelo de Hospital

Público del país; esto significa que la labor que se adelanta en él es tomada como referencia por otros Hospitales para su funcionamiento y atención.

Misión: Prestar Servicios de Salud Integrales de mediana y alta complejidad en las modalidades ambulatoria e internación y así brindar atención humanizada, segura y eficiente con la tecnología adecuada y talento humano calificado. Además de fortalecer la formación académica orientada a la investigación e innovación para satisfacer las necesidades de los usuarios.

Visión: En el año 2022, será la institución pública líder en la prestación de Servicios de Salud de mediana y alta complejidad destacándose por su calidad, excelencia, competitividad, sostenibilidad financiera y formación del talento humano en salud; generando conocimiento e innovación y a su vez siendo amigable con el medio ambiente.

2.1.2. *Gestión Hospitalaria*

Atender a un paciente en un centro de salud es mucho más que solo diagnosticar, tratar y curar; dar una atención medica de calidad implica una gestión eficiente de la información hospitalaria, coordinando la atención medica con los servicios de laboratorio, los inventarios y la administración, todo esto acorde a la información clínica del paciente. Por esta razón, es fundamental garantizar una gestión hospitalaria eficiente, en donde se diseñen y desarrollen estrategias para obtener una adecuada relación entre calidad, precio y el trabajo duro para cumplir la efectividad en los servicios de salud. De manera que se debe estar atento a todos los procesos del hospital para tener la capacidad de identificar los problemas y, con ayuda de la tecnología, solucionarlos.

La Gestión Hospitalaria se preocupa por mantener tres conceptos: Gestión, calidad y mejora continua.

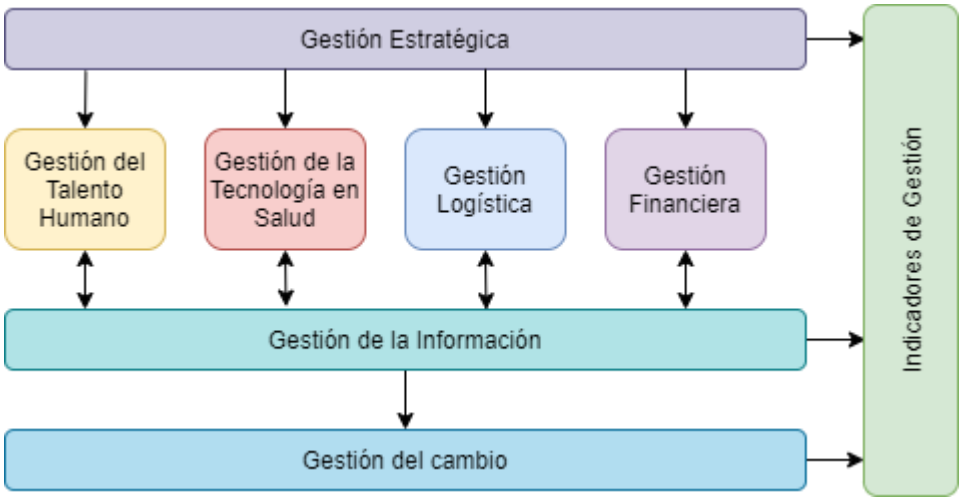

Figura 1. Gestión Hospitalaria
Adaptado de Organización para la Excelencia de la Salud (2019). Gestión Hospitalaria.

Para la gestión hospitalaria es importante realizar el diseño de procesos en cada área, incluyendo procesos operativos transparentes con el que fin de facilitar y optimizar la asistencia, además de, en lo posible, alcanzar una organización sustentable, logrando una disminución de los problemas y una mejora en el funcionamiento de la institución. Dichos procesos están involucrados en la atención al paciente, desde que entra hasta que se va. Una vez obtenido lo anterior, es posible realizar un análisis de la calidad técnica de cada proceso, su efectividad clínica (resultado), la satisfacción (calidad), los tiempos (eficacia) y el consumo de recursos (costo).

Esta actividad de gestión hospitalaria se encuentra en constante evolución, lo cual ha permitido llegar a una etapa de transformación digital, en donde se han aplicado las Tecnologías de la Información (TI) permitiendo a las instituciones de salud comunicarse y trabajar de manera más eficiente.

Algunos aspectos son muy importantes en esta gestión, ya que se garantiza una visión completa del hospital, permitiendo un mayor control de los procesos:

- Conocer el perfil de los pacientes: saber en qué marco se encuentra la institución, si posee un historial completo y digitalizado de la situación clínica de los pacientes y cuáles son las patologías más comunes.
- Recepción y atención de calidad: el ingreso del paciente al hospital es muy importante ya que cualquier falla en esta atención genera una imagen negativa de la institución. Para evitar esto, lo mejor es implementar procesos claros y entrenar a los colaboradores. La aplicación de un sistema de gestión hospitalaria reduce el tiempo de la atención y mejora la organización.
- Mejora continua: siempre es importante actualizar y, si es necesario, mejorar la gestión con apoyo en la metodología de procesos hospitalarios con el fin de mantener una gestión eficiente, innovadora y de calidad.
- Entrenamiento: motivar a los empleados a mantenerse actualizados con el fin de mantener la calidad de la asistencia.
- Proceso de intercambio de informaciones: integrar las tecnologías optimiza la comunicación entre los departamentos del hospital facilitando los procesos operacionales y asistenciales.

De este modo, el Hospital Universitario de Santander (HUS) dispone de un mapa de procesos (figura) el cual orienta cada una de las actividades necesarias para el desarrollo de los cada uno de los procesos estratégicos, de apoyo y evaluación dentro del hospital, centrados al cumplimiento de los procesos misionales, siendo estos últimos los que permiten el cumplimiento de la misión del HUS.

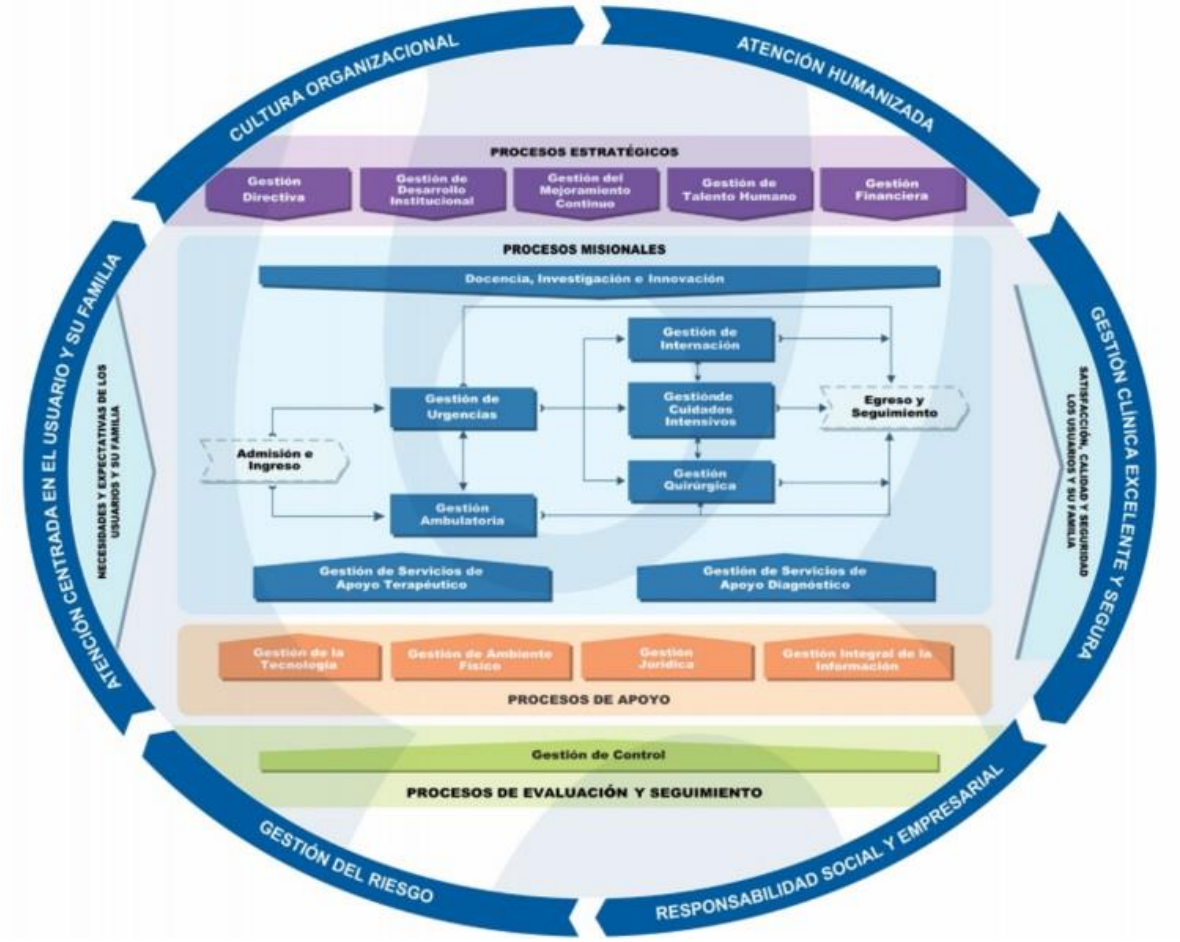

Figura 2. Mapa de procesos del HUS.
Tomado de MAPA DE PROCESOS ESE HUS.

2.1.3. *Servicio de camilleros*

El servicio de camilleros se ocupa del traslado del personal en situación de enfermedad y/o discapacidad, garantizando los cuidados que se deben tener en cuenta en cada caso. Esta es una actividad muy importante que se debe hacer con precisión, destreza y conocimiento para lograr la adecuada recuperación de la persona o paciente, aplicando técnicas manuales y mecánicas para trasladar al paciente de un lugar a otro.

De modo que, el camillero requiere de conocimientos específicos sobre el grado de criticidad de cada paciente y así saber cómo se debe trasladar a cada uno y en qué circunstancias

debe realizar el traslado, acompañado ya sea por enfermera o médico. Esto conlleva además a manejar principios de control de infecciones por el contacto con muchos pacientes en diferentes condiciones de salud, además el trato con los mismo debe ser primordial para su seguridad. Como trabaja en un medio en el que se usa un lenguaje específico, debe conocer los significados de algunas palabras y códigos que habitualmente usan los profesionales dentro de cada institución y dentro de cada servicio.

De igual forma, es importante que el camillero conozca el funcionamiento de las herramientas que tiene a disposición para el traslado de pacientes como, por ejemplo, silla de ruedas, camilla, tabla rígida, tanque de oxígeno, etc.

Con el fin de tener un orden y llevar a cabo la gestión para la realización de los servicios de camilleros, existe una central de camilleros en el Hospital Universitario de Santander (HUS). Esta central se basa principalmente en el registro (de una solicitud) y asignación (del camillero) de los servicios y, si es posible, la consulta de estos.

2.1.4. *Solicitud*

Esta actividad dentro del servicio de camilleros se refiere a la solicitud de un servicio desde cualquier área del HUS y en cualquier momento.

Actualmente, en la central de camilleros del HUS, esta actividad se hace de manera manual, en donde se reciben llamadas telefónicas o correos electrónicos solicitando algún servicio. Esta información se escribe en un pequeño formulario en físico para luego ser entregada al camillero.

2.1.5. *Asignación*

Esta actividad se encarga de la asignación de un camillero a un servicio que se ha solicitado. Posterior a esto, también se encarga de la asignación de horas conforme se va realizando el servicio.

En el momento, esto también se hace de manera manual al igual que la solicitud de un servicio; el nombre del camillero se agrega en el formulario, el cual se llenó con la información de cierto servicio, y las horas se escriben en el mismo formulario a medida que se va realizando el servicio.

2.1.6. *Consulta*

La actividad de consulta consiste en proporcionar a la institución la información completa de los servicios solicitados, además de poder consultar los tiempos de cada servicio y el tiempo promedio según el tipo de servicio.

Actualmente, en el HUS, esta información que se pide se encuentra de manera física, por lo que es muy complicado saber los tiempos de respuesta de los servicios y el promedio de un grupo de servicios, ya que obtener estos datos tomaría mucho tiempo.

2.1.7. *Arquitectura de Software*

La arquitectura de software es el conjunto de estructuras necesarias para razonar acerca del sistema. Es importante establecer estas estructuras, ya que con esto podemos saber cómo se van a organizar cada una de las partes del sistema y cómo estas se conectarán. Una segunda definición proveniente del libro “Design It: From Programmer to Software Architect”, el cual dice que la arquitectura de software es el conjunto de decisiones de diseño importante para organizar el software y promover los atributos de calidad deseados.

La arquitectura de software nos permite identificar el camino a seguir para poder cumplir con los requerimientos del sistema, de modo que, se debe analizar cada uno de los requerimientos para saber qué se va a hacer y cómo se va a hacer, además de empezar a establecer temas tales como servidores, tecnologías, bases de datos entre otras cosas.

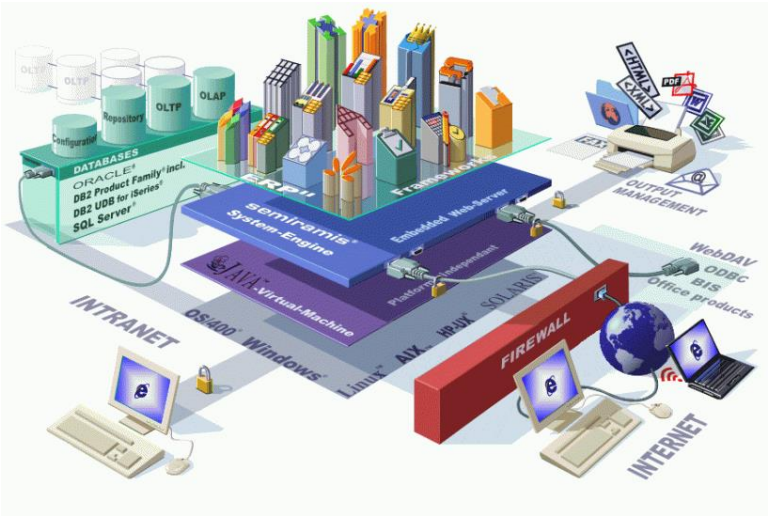

Figura 3. Arquitectura de software.
Tomado de *https://instintobinario.com/arquitectura-en-tres-capas/*

Definir las tecnologías es uno los pasos más importantes de la arquitectura de software, pero esto no implica que el tomar una decisión sea algo permanente que impida alguna modificación en el futuro. De modo que, uno de los objetivos de la arquitectura de software es crear una estructura de la aplicación que sea fácilmente escalable y que no esté fuertemente acoplada (que todo dependa de todo, lo que evita hacer cambios de manera sencilla).

Existen 4 etapas principales y otras también importantes en el proceso de definir la estructura de la aplicación, basadas en la arquitectura de software:

- ✓ Requerimientos: lenguajes, marcos de trabajo, especificaciones de equipo.
- ✓ Análisis: profundización en los procesos de negocio, casos de uso.
- ✓ Diseño: definiciones de tecnologías adecuadas.
- ✓ Documentación: información ordenada que sirva como referencia a todos, siendo este el marco de referencia de los involucrados.
- ✓ Desarrollo: asignación de tareas para hacer, si es necesario, correcciones al sistema final.
- ✓ Pruebas: verificar si hay algo en el diseño que pueda no funcionar y corregirlo.
- ✓ Implementación: se realiza el sistema en producción.

Pero la arquitectura es algo más que una estructura; el IEEE Working Group on Architecture la define como "el concepto de más alto nivel de un sistema en su entorno". Esto también incluye el ajuste con la integridad del sistema, con las restricciones económicas, con las preocupaciones estéticas y con el estilo. No se limita a un enfoque interior, si no que tiene en cuenta el sistema en su totalidad dentro del entorno de usuario y el entorno de desarrollo, un enfoque exterior.

2.1.8. *Desarrollo Web*

Este concepto suele confundirse en ocasiones con el Diseño Web, sin embargo, estos dos términos hacen referencia a actividades muy distintas. El desarrollo web define la programación necesaria para construir una aplicación o sitio web. Este se divide, de forma general, en Frontend (la parte cliente) y Backend (la parte servidor).

El Frontend es el desarrollo web en el ámbito del cliente, es decir, en el navegador web; es esa parte del programa o dispositivo a la que un usuario puede acceder directamente. Aquí se realiza la composición, diseño e interactividad usando HTML, CSS y JavaScript, obteniendo el aspecto visual del sitio web, los menús desplegables y el texto. Dominando estas tecnologías es posible usar algunos frameworks, librerías o preprocesadores para crear todo tipo de interfaces de usuario. Algunos de ellos son React, Vue, Angular, Svelte, Bootstrap, Foundation, Sass, Less, Stylus y PostCSS.

Por otro lado, en el Backend se realiza lo que no se ve, es decir, la parte en donde se almacenan los datos, ya que sin datos no hay Frontend. El Backend consiste en el servidor que acoge la web, una aplicación para ejecutarlo y una base de datos; podría decirse que es la capa de acceso a los datos de un software o dispositivo. En el Backend se utiliza programas de computación para asegurar que el servidor, la aplicación y la base de datos trabajen en conjunto. Para hacer este trabajo, se utiliza una serie de lenguajes del lado del servidor, como PHP, Ruby, Python y Java.

Figura 4. FrontEnd (cliente) y BackEnd (servidor)
Tomado de Ironhack (2019). Diferencias entre el frontend y backend

2.2. Marco contextual

2.2.1. *Requerimientos funcionales y no funcionales*

Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir (Wikipedia, 2022).

Los requerimientos no funcionales no se refieren directamente a las funciones específicas suministradas por el sistema (características de usuario), sino a las propiedades del sistema: rendimiento, seguridad, disponibilidad. En palabras más sencillas, no hablan de “lo que” hace el sistema, sino de “cómo” lo hace (Medium, 2022).

2.2.2. *Modelo de prototipos*

El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final. Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación; gracias a esta se refinan los requisitos del software que se desarrollará. La interacción ocurre cuando el prototipo se ajusta para satisfacer

las necesidades del cliente. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo (Wikipedia, 2022).

2.2.3. *Diagrama de clases y casos de uso*

Un diagrama de clases nos permitirá representar gráficamente y de manera estática la estructura general de un sistema, mostrando cada una de las clases y sus interacciones representadas en forma de bloques, los cuales son unidos mediante líneas y arcos. Los diagramas de clases son el pilar fundamental del modelado con UML, siendo ampliamente utilizados tanto para análisis como para diseño de sistemas y software en general (Noguera, 2015). Los diagramas de casos de uso permiten saber las acciones o funciones que un sistema lleva a cabo para obtener un resultado.

2.2.4. *Aplicación de una sola página (SPA)*

Este tipo de aplicaciones funciona ejecutando todo su contenido en una sola página, logrando obtener diferentes vistas para cada apartado de la página web. Funciona cargando el contenido HTML, CSS y JavaScript por completo al abrir la web. Al ir pasando de una sección a otra, solo necesita cargar el contenido nuevo de forma dinámica si este lo requiere, pero no hace falta cargar la página por completo. Esto mejora los tiempos de respuesta y agiliza mucho la navegación, favoreciendo así a la experiencia de usuario. (Mora, 2021).

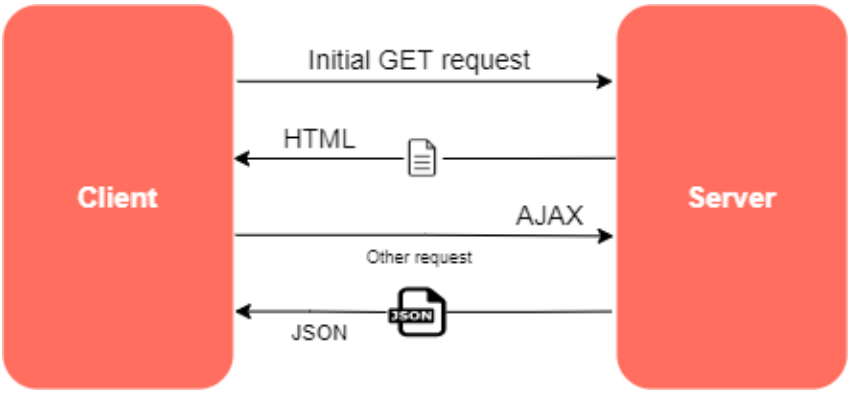

Figura 5. Diagrama de aplicación web SPA
Adaptado de Devopedia (2018). https://devopedia.org/single-page-application

2.2.5. *API REST*

Una API en un conjunto de definiciones y protocolos que permite la comunicación de datos entre aplicaciones. REST, que es la abreviación de REpresentational State Transfer, es un estilo de diseño de arquitectura de software que describe cualquier interfaz entre diferentes sistemas que utilice HTTP para comunicarse. De modo que API REST se refiere a un backend que contiene un conjunto de métodos configurados que hacen ciertas funcionalidades e interactúan con la base de datos y que pueden ser consumidos a través del protocolo HTTP por cualquier tipo de cliente, y cuyas respuestas del servidor se dan en formato JSON o XML.

2.3. Marco tecnológico

2.3.1. *Java*

Es un lenguaje de programación orientado a objetos y el segundo más utilizado en el mundo. Es rápido, seguro y confiable, diseñado para permitir a los desarrolladores una plataforma de continuidad y para reducir dependencias al programar.

2.3.2. *IntelliJ IDEA*

Es un entorno de desarrollo integrado inteligente y sensible al contexto para trabajar con Java y otros lenguajes como JVM como Kotlin, Scala y Groovy en todo tipo de aplicaciones (JETBRAINS, 2022). Una característica de IntelliJ IDEA es que verifica la calidad y validez del código con inspecciones sobre la marcha, ofreciendo ayuda a través de acciones contextuales para resolver algún problema que se presente.

2.3.3. *Spring Boot*

Es una infraestructura que elimina gran parte del trabajo de configurar aplicaciones de Spring, a partir de un conjunto de herramientas. Gracias a esto se pueden crear aplicaciones autocontenidas enfocadas únicamente al desarrollo, es decir, para crear el código, dejando a un lado la configuración de la arquitectura.

2.3.4. *Spring Data JPA*

Es una herramienta basada en Spring y que trabaja en conjunto con JPA (Java Persistence API), la cual reduce el código y mejora la implementación de capas de acceso a datos, permitiendo a su vez un conjunto de funcionalidades muy completo.

2.3.5. *Angular*

Es un framework para aplicaciones web desarrollado en Typescript, de código abierto, mantenido por Google, utilizado para crear y mantener aplicaciones web de una sola página (SPA). Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

2.3.6. *Visual Studio Code*

Es un editor de código fuente ligero muy potente y flexible compatible con varios lenguajes de programación y disponible para sistemas operativos como Windows, Linux y macOs.

2.3.7. *Typescript*

Es un lenguaje de programación libre y de código abierto construido a un nivel superior de JavaScript (JS). Por esta razón este lenguaje tiene características adicionales permitiendo escribir el código de manera más sencilla obteniendo más coherencia y menos errores.

2.3.8. *SQL Server*

Es un sistema de gestión de bases de datos relacional desarrollado como un servidor, el cual da servicio a otras aplicaciones de software que pueden funcionar ya sea en el mismo ordenador o en otro ordenador a través de una red.

2.3.9. *Servidor apache*

Es un software de servidor web HTTP de código abierto desarrollado y mantenido por Apache Software Foundation. Su función principal es servir a los usuarios los necesario para poder visualizar la web, en donde las solicitudes se hacen, generalmente, a través de un navegador.

2.3.10. *GitHub*

Es una plataforma que permite alojar proyectos utilizando el sistema de control de versiones distribuida, es decir, la base del código entero y su historial se encuentran disponibles en la computadora de todo desarrollador, lo cual permite un fácil acceso a las bifurcaciones y fusiones,

es decir, duplicar el código fuente para realizar cambios sin alterar el proyecto y fusionar este código al código fuente principal para hacerlo oficial, respectivamente.

2.4. Marco de arquitectura de software

2.4.1. *Modelo Vista Controlador (MVC)*

Es un estilo de arquitectura de software muy maduro y con gran validez en todo tipo de aplicaciones, este separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos: Modelo, vista y controlador. El Modelo contiene una representación de los datos, La vista compone la información que se envía al cliente y el Controlador es un intermediario entre el Modelo y la Vista encargado de gestionar el flujo de información entre ellos adaptando los datos a las necesidades de cada uno.

3. Requerimientos del proyecto

3.1. Listado de requerimientos

A continuación, se muestran los requerimientos funcionales del proyecto junto con una descripción en cada uno de ellos. Estos fueron definidos al inicio y durante el desarrollo del proyecto.

Tabla 1. Requerimientos funcionales

Id: RF001	**Nombre:** Crear cuenta	**Prioridad:** Alta
Descripción: Los usuarios podrán registrarse ingresando los siguientes datos: nombre, nombre de usuario y contraseña. Estos usuarios serán tipo USER. Tendrán permisos de solicitud y asignación de horas.		
Id: RF002	**Nombre:** Iniciar Sesión	**Prioridad:** Alta
Descripción: Para acceder a los servicios es necesario iniciar sesión con los siguientes datos: nombre de usuario y contraseña.		
Id: RF003	**Nombre:** Cerrar Sesión	**Prioridad:** Alta
Descripción: Los usuarios que hayan iniciado sesión pueden eliminar el token almacenado que contiene la información del usuario para salir de la sesión.		
Id: RF004	**Nombre:** Solicitar servicio	**Prioridad:** Alta
Descripción: Los usuarios (cualquiera) pueden solicitar un servicio de camilleros en donde se requieren los datos: fecha, servicio solicitado, solicitante, destino del servicio, transporte, insumo, acompañante, aislamiento, observaciones, numero de documento del paciente y nombre del paciente.		
Id: RF005	**Nombre:** Asignar camillero	**Prioridad:** Alta
Descripción: El usuario encargado de la asignación de camilleros a los servicios solicitados (ADMIN) podrá realizar esta acción seleccionando el respectivo servicio y asignándole el camillero disponible.		
Id: RF006	**Nombre:** Registrar horas	**Prioridad:** Alta
Descripción: Los usuarios (cualquiera) podrán registrar las horas a medida que el camillero realice el servicio; estas son las horas de ejecución y hora de finalización.		
Id: RF007	**Nombre:** Cancelar servicio	**Prioridad:** Alta
Descripción: Si algún servicio no se pudo realizar por algún motivo este debe reportarse, cambiando el estado del servicio y colocando el motivo por el cual no se realizó dicho servicio, esto solo lo podrá hacer el usuario tipo ADMIN.		
Id: RF008	**Nombre:** Imprimir servicio	**Prioridad:** Alta
Descripción: El usuario (ADMIN) podrá imprimir la información necesaria del servicio para que este se ejecute.		

Id: RF009	**Nombre:** Consultar servicios	**Prioridad:** Alta
Descripción: El usuario (ADMIN) podrá consultar los servicios realizados hasta la fecha.		
Id: RF010	**Nombre:** Ver servicio	**Prioridad:** Alta
Descripción: El usuario (ADMIN) podrá ver la información completa de cada servicio.		
Id: RF011	**Nombre:** Filtrar servicios	**Prioridad:** Media
Descripción: Al consultar los servicios (usuario ADMIN) se podrá aplicar filtros para ver servicios que tengan en común cierto criterio.		
Id: RF012	**Nombre:** Crear camillero	**Prioridad:** Alta
Descripción: El usuario (ADMIN) podrá crear un nuevo camillero.		
Id: RF013	**Nombre:** Modificar camillero	**Prioridad:** Alta
Descripción: El usuario (ADMIN) podrá modificar la información de los camilleros.		
Id: RF014	**Nombre:** Ver camillero	**Prioridad:** Alta
Descripción: El usuario (ADMIN) podrá ver la información de cada camillero.		

Tabla 2. Requerimientos no funcionales

Id: RNF01	**Nombre:** Interfaz gráfica SPA	**Prioridad:** Alta
Descripción: La interfaz gráfica de usuario será una aplicación de una sola página (SPA) creada con ayuda del framework de Angular.		
Id: RNF02	**Nombre:** Actualizar información en la base de datos	**Prioridad:** Alta
Descripción: La información que se cree y modifique de los servicios y los camilleros se debe ver reflejada de manera inmediata en la base de datos que se use. Al igual que los usuarios creados desde la aplicación.		
Id: RNF03	**Nombre:** Compatibilidad con navegadores	**Prioridad:** Baja
Descripción: Los usuarios podrán acceder al software desde la mayoría de los navegadores recientes (Chrome, Firefox, Edge, etc).		
Id: RNF04	**Nombre:** Tiempos cortos de respuesta	**Prioridad:** Baja
Descripción: Principalmente para peticiones de consultas que se hagan, estas no deben durar más de 8 segundos para obtener una respuesta.		

3.2. Análisis de requerimientos

Se presenta parte del diseño del proyecto en donde se muestra el diagrama de casos de uso, diagrama de actividades y el diagrama de estado para los servicios.

3.2.1. Diagrama de casos de uso

El diagrama de casos de uso muestra el proceso de interacción de los usuarios necesario para llevar a cabo un servicio de camilleros.

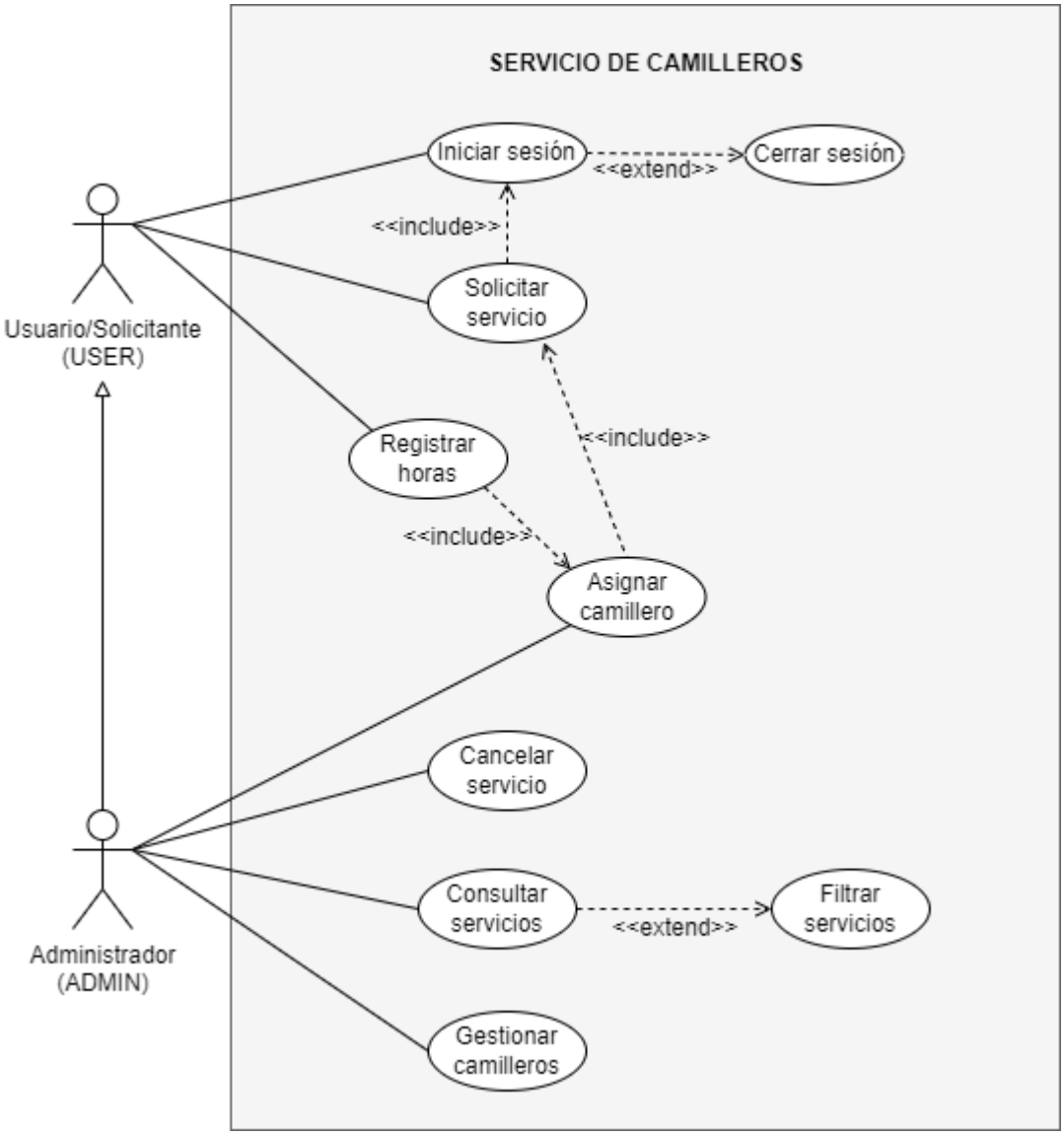

Figura 6. Diagrama de casos de uso

3.2.2. Diagrama de actividades

A continuación, se presentan algunos diagramas de actividades importantes de la aplicación web: registrarse e iniciar sesión, consultar un servicio, solicitar un servicio y asignar un camillero a un servicio.

En el apéndice A se encuentran los demás diagramas de actividades correspondientes a las demás funciones de la aplicación.

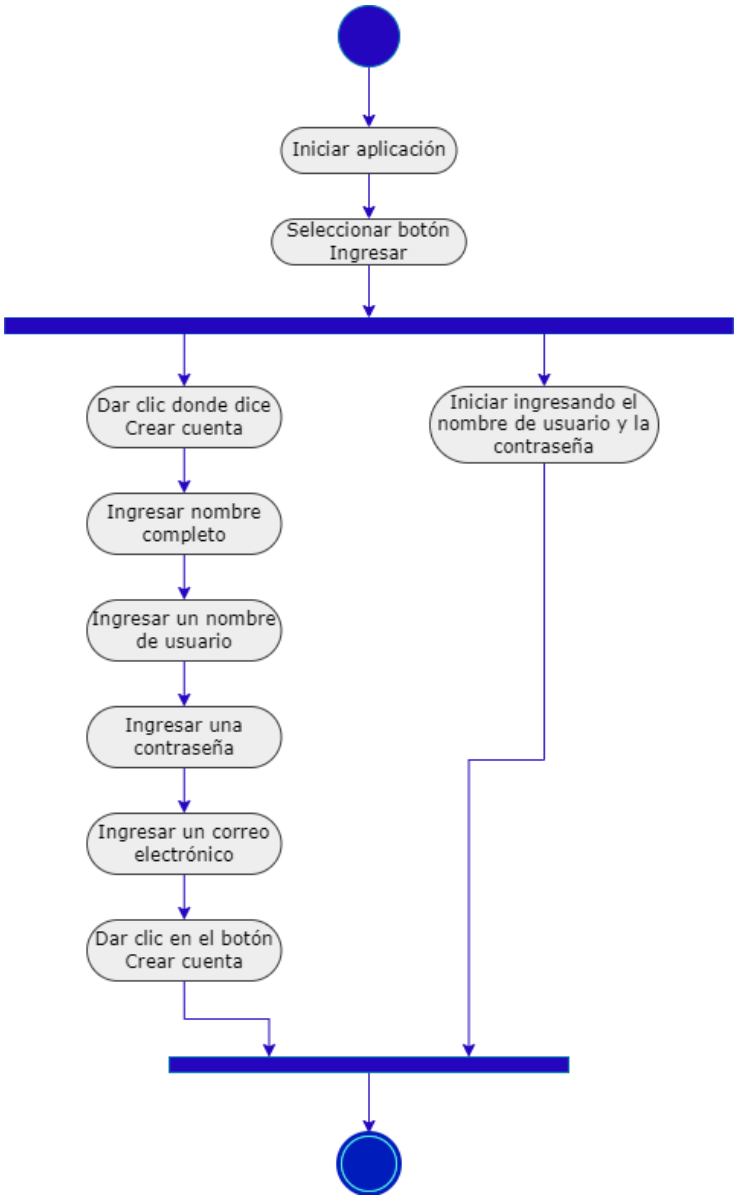

Figura 7. Diagrama de actividades para registrarse e iniciar sesión

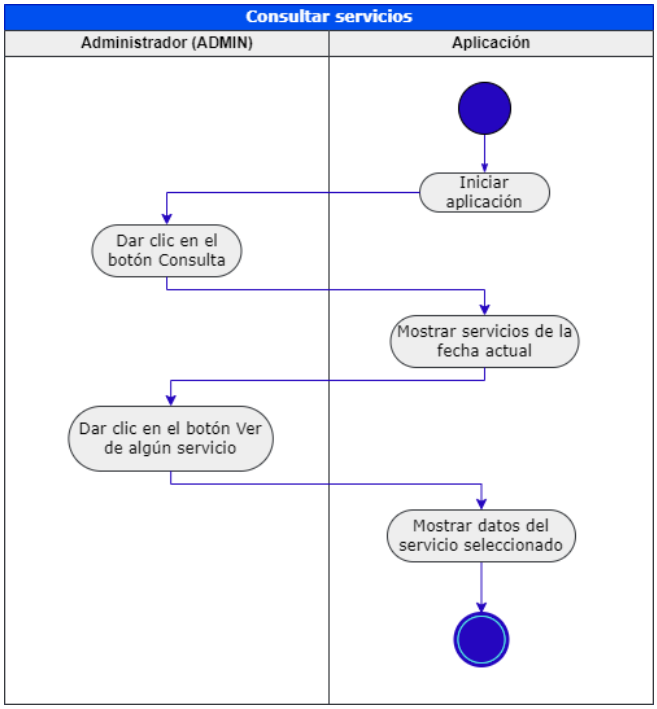

Figura 8. Diagrama de actividades para consultar un servicio

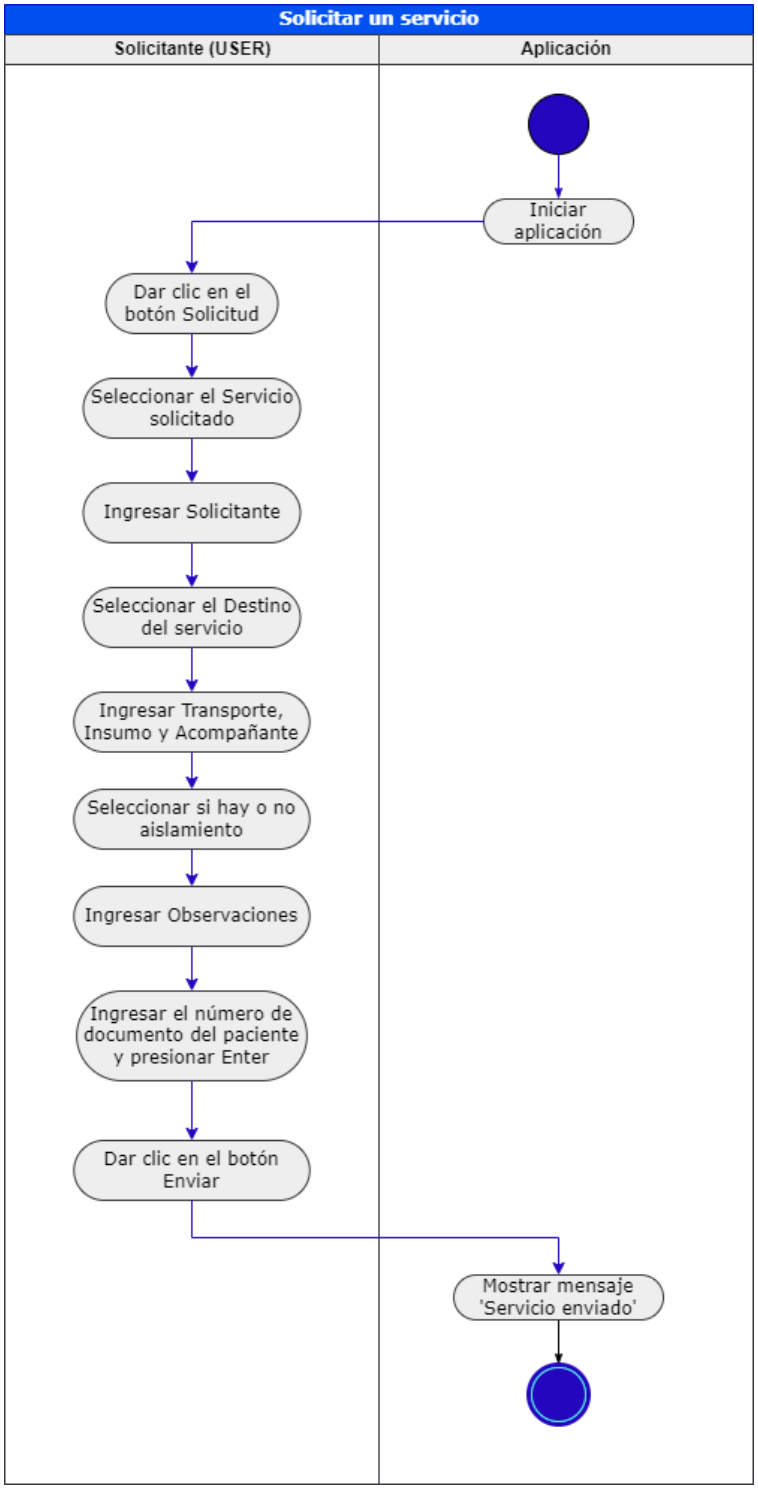

Figura 9. Diagrama de actividades para solicitar un servicio

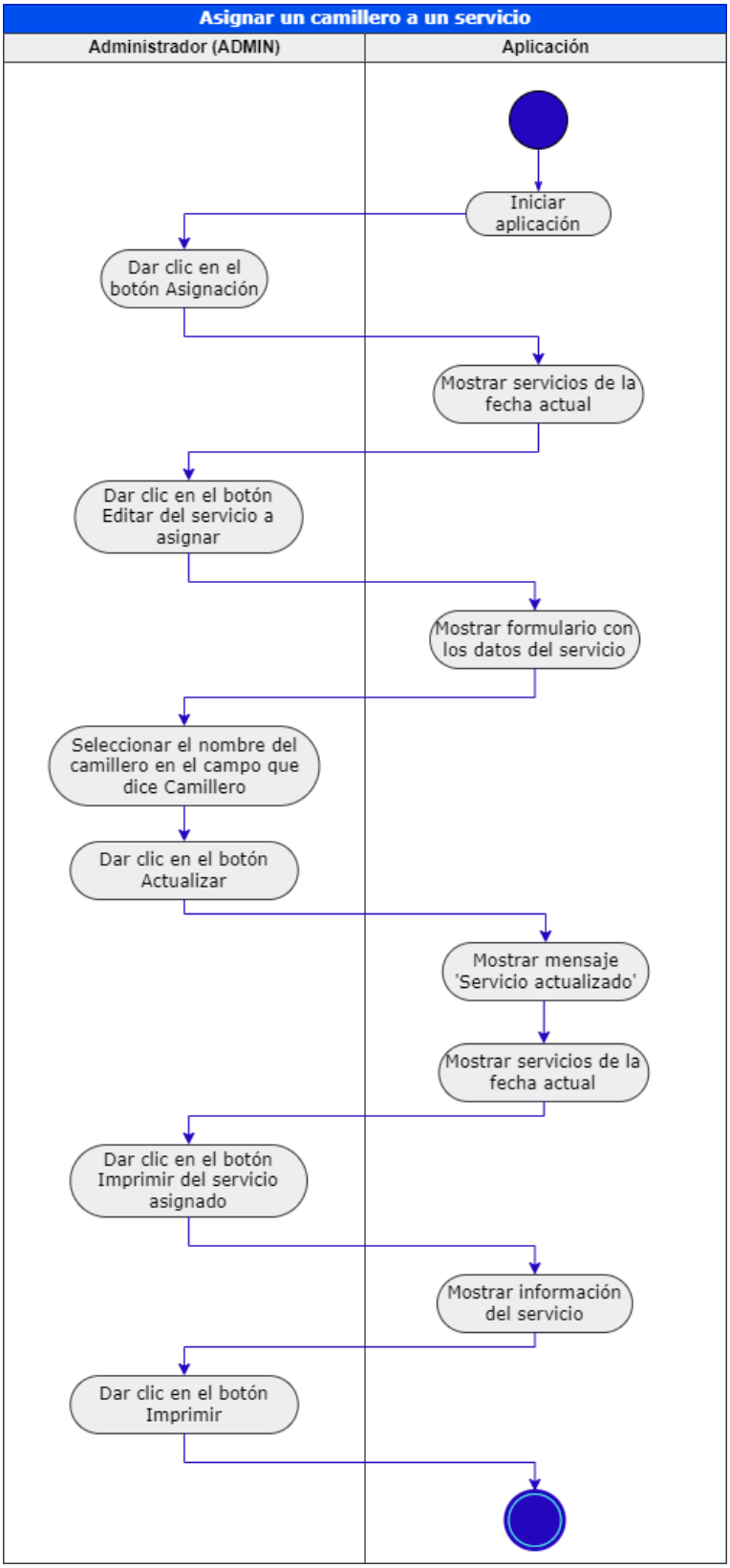

Figura 10. Diagrama de actividades para asignar un camillero a un servicio

3.2.3. Diagrama de estado

Aquí podemos ver el estado en que se puede encontrar el servicio teniendo en cuenta el proceso para realizarlo.

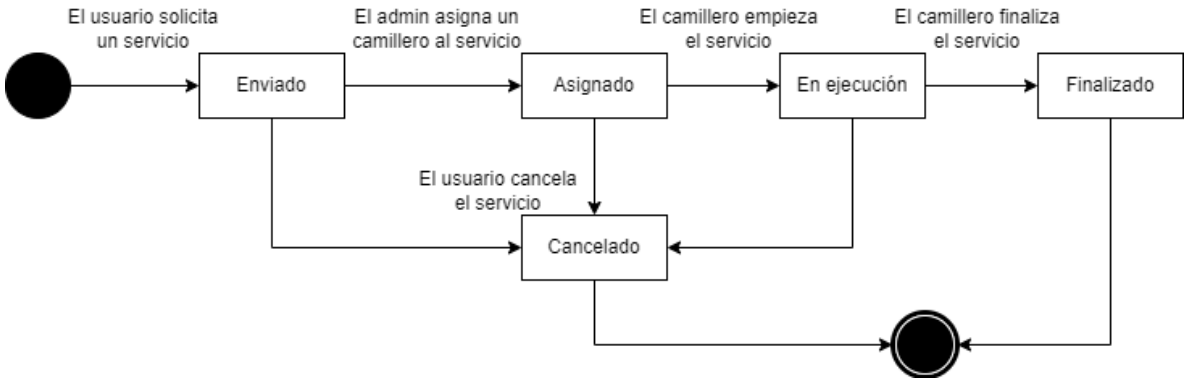

Figura 11. Diagrama de estado de un servicio

4. Diseño de la solución

4.1. Arquitectura

En la figura 11 se presenta el diagrama de arquitectura general diseñado e implementado en el prototipo de software.

En la parte izquierda se encuentra el sistema de gestión de base de datos relacional (RDBMS) producido por Microsoft, este es SQL Server. Utilizando un servidor del HUS fue posible acceder a una base de datos de prueba y trabajar con datos reales.

En el centro, Spring Boot es utilizado para crear la aplicación Spring de manera rápida y así poder desplegar la aplicación de forma más sencilla, esto acelera el trabajo sin tener que realizar manualmente ciertas configuraciones.

Por último, en el lado derecho está la aplicación Angular, se tiene un framework JavaScript de código abierto para el desarrollo front-end. Este se comunica con el servidor a través de protocolos http usando el módulo HttpClient.

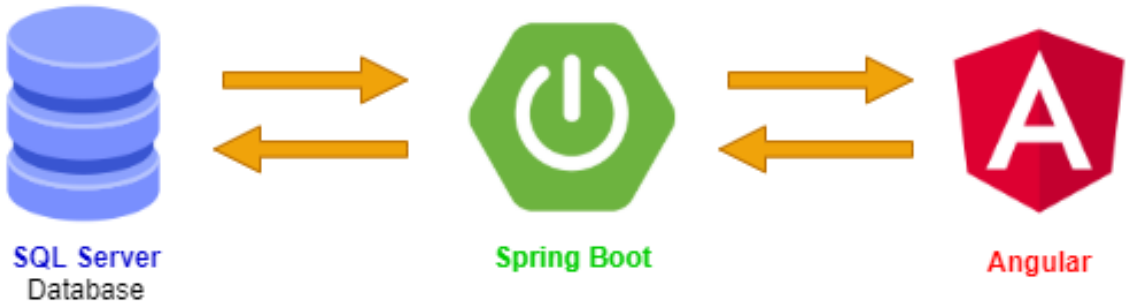

Figura 12. Arquitectura general del proyecto

4.2. Diseño de interfaces

4.2.1. Spring Boot Back-end

Se realizan operaciones CRUD y métodos de búsqueda con JpaRepository de Spring Data JPA. Algunas de ellas se muestran a continuación:

Tabla 3. Métodos implementados en el Back-End para los servicios

Métodos	Urls	Acciones
GET	/servicio/lista	Obtener todos los servicios
GET	/servicio/detail/{id}	Obtener un servicio por id
POST	/servicio/create	Crear un servicio
PUT	/servicio/update	Actualizar un servicio
DELETE	/servicio/delete/{id}	Eliminar un servicio

En la figura se puede observar más a detalle el diseño del componente back-end. Al realizar una solicitud HTTP al servicio REST, Spring Boot exporta REST Apis usando Spring Web MVC e interactúa con la base de datos SQL Server usando Spring Data JPA.

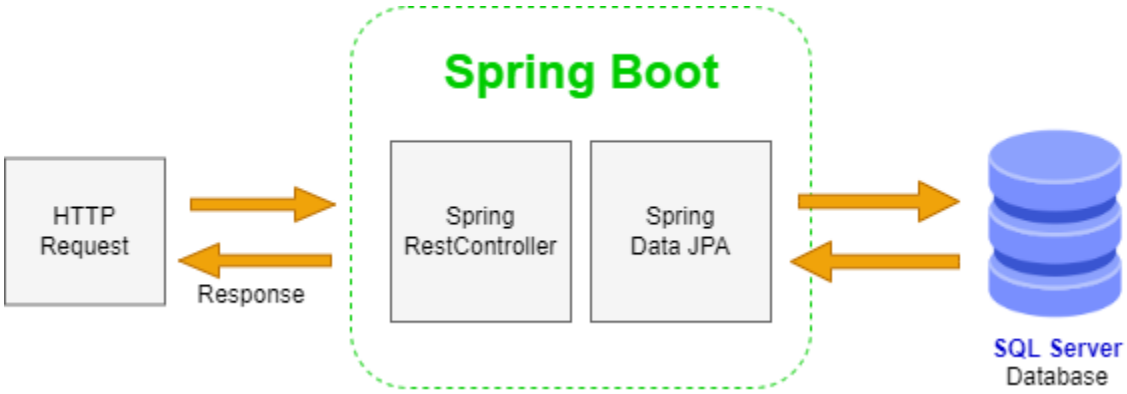

Figura 13. Arquitectura del componente Back-End
Adaptado de Bezkoder (2020). https://www.bezkoder.com/angular-12-spring-boot-mysql/#Project_Structure

4.2.2. Angular 12 Front-end

En la figura se puede ver más a detalle el diseño del componente front-end. El componente de la aplicación (app) es un contenedor con salida de enrutador, es decir, tiene una barra de

navegación que enlaza las rutas a través de routerLink. Los Componentes obtienen y muestran la información, y a partir de formularios se pueden crear nuevos datos y editarlos. Los Componentes llaman a los métodos del Servicio, los cuales usan Angular HttpClient para realizar solicitudes http y recibir respuestas.

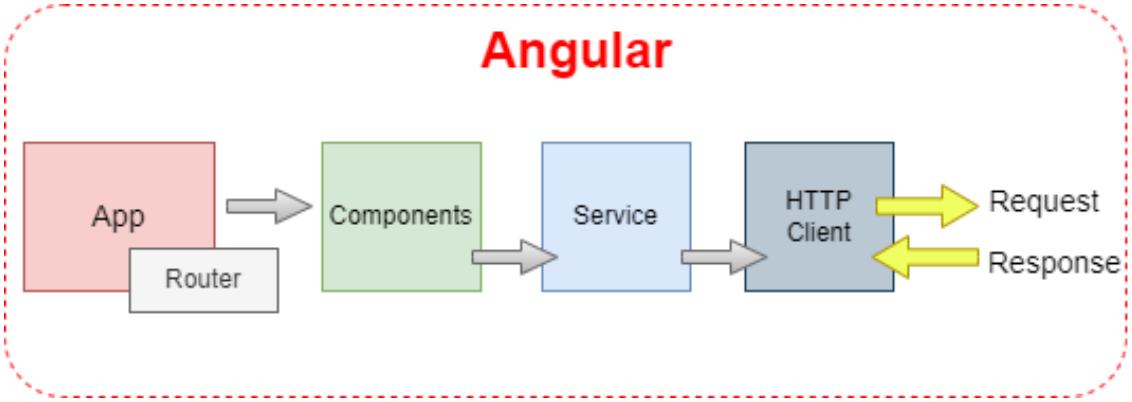

Figura 14. Arquitectura del componente Front-End
Adaptado de Bezkoder (2020). https://www.bezkoder.com/angular-12-spring-boot-mysql/#Project_Structure

Es usado el patrón de arquitectura Modelo-Vista-Controlador (MVC), permitiendo clasificar la información, la lógica del sistema y la interfaz que se le presenta al usuario. Con esto es posible gestionar las entradas y salidas del sistema, encargar a uno o varios modelos la búsqueda de la información y tener una interfaz que muestra los resultados al usuario final. Así pues, este patrón permite modificar cada uno de los componentes (controlador, modelo y vista) sin necesidad de afectar a los demás. En la figura se muestra la arquitectura mencionada anteriormente.

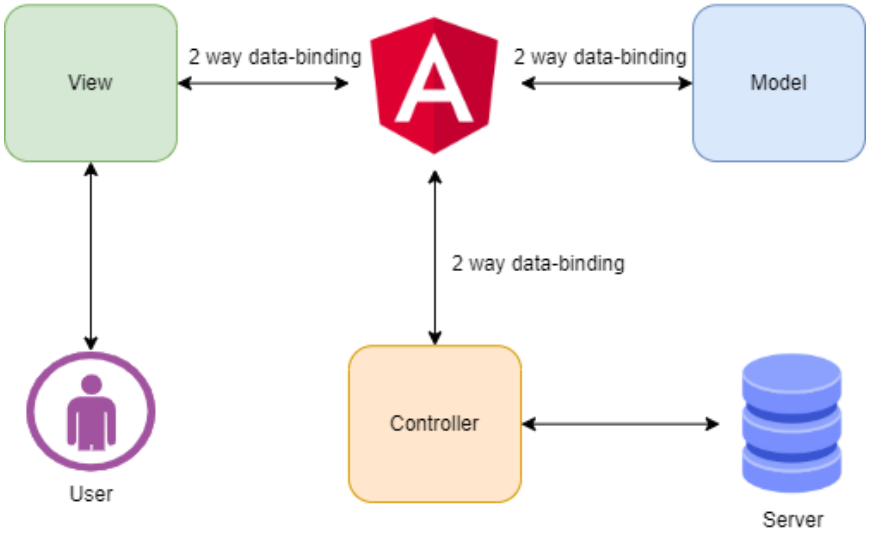

Figura 15. Arquitectura MVC de la aplicación web
Adaptado de Maheshwari (2020). https://nmgtechnologies.com/blog/web-application-development-with-angularjs.html

Finalmente, esta aplicación web es de una única página, es decir, Single Page Application (SPA). Esta funciona cargando los estilos y formatos por completo al abrir la web, de esta manera solo se necesita cargar el contenido nuevo de forma dinámica para ir de una sección a otra, sin necesidad de cargar la página por completo. “Esto mejora los tiempos de respuesta y agiliza mucho la navegación, favoreciendo así a la experiencia de usuario” (Mora, 2021). En la figura podemos observar la arquitectura para la aplicación SPA.

4.3. Diseño de datos

A continuación, se presentan las tablas usadas para el desarrollo de la aplicación. Como se mencionó anteriormente, se usó un servidor del HUS en SQL Server, el cual contenía una base de datos de prueba con datos reales.

Las tablas Genpacien y Genareser ya hacían parte de la base de datos de prueba desde un inicio. Estas se usaron para extraer la información de los pacientes y de las áreas del hospital, respectivamente.

Fue necesario crear tablas para llevar a cabo el proceso de desarrollo; las tablas Servicio y Camillero fueron creadas con el fin de guardar la información de servicios y camilleros pertenecientes al HUS.

Las tablas Rol, Usuario y usuario_rol se crearon para que los usuarios del hospital se registraran y posteriormente lograran iniciar sesión para realizar las actividades de solicitud, asignación y consulta de los servicios. Con estas últimas tablas fue posible establecer permisos dependiendo del tipo de Rol de cada usuario:

- USER: solicitar un servicio
- ADMIN: solicitar, asignar y consultar servicios. Gestionar camilleros.
- SUPERADMIN: solicitar, asignar y consultar servicios. Gestionar camilleros. Eliminar servicios y camilleros.

Tabla 4. Tablas de la base de datos requeridas para el desarrollo del proyecto

TABLAS	DESCRIPCIÓN	ATRIBUTOS	
Genpacien	Contiene la información de los pacientes del HUS.	OID (PK, int) PACNUMDOC (nvarchar(15)) PACPRINOM (nvarchar(30)) PACSEGNOM (nvarchar(30)) PACPRIAPE (nvarchar(30)) PACSEGAPE (nvarchar(30))	
Genareser	Aquí se guardan las áreas existentes del hospital, las cuales pueden realizar la solicitud de un servicio.	OID (PK, int) GASCODIGO (nvarchar(10)) GASNOMBRE (nvarchar(120))	
Servicio	Contiene la información de los servicios de camilleros solicitados por las áreas del HUS.	id (PK, bigint) fecha (date) servicioSolicitado (FK, int) solicitante (nvarchar) destinoServicio (FK, int) transporte (nvarchar) insumo (nvarchar) familiar (nvarchar)	idCamillero (FK, int) horaEnvio (time(7)) horaAsignacion (time(7)) horaEjecucion (time(7)) horaFinalizacion (time(7)) minutosTiempoTotal (int) cancelado (bit)

		aislamiento (bit) observaciones (nvarchar) oidGenpacien (FK, int)	motivoCancelado (nvarchar) horaCancelacion (time(7))
Camillero	Entidad encargada de realizar los servicios solicitados, en esta tabla se guarda la información de los camilleros.	idCamillero (PK, int) estadoCamillero(bit) emailCamillero(nvarchar) nombreCamillero (nvarchar)	
Rol	Aquí se guardan los tipos de roles que existen para iniciar sesión y acorde a eso de establecen los permisos.	id (PK, int) rolNombre (varchar)	
Usuario	Esta tabla contiene los usuarios que pueden iniciar sesión en la aplicación web.	id (PK, int) email (nvarchar) nombre (nvarchar) nombreUsuario (nvarchar) password (nvarchar)	
usuario_rol	Esta tabla guarda la información de relación de los usuarios con el o los roles que tienen.	usuario_id (PK, FK, int) rol_id (PK, FK, int)	

Para una mejor comprensión sobre las tablas y sus relaciones se presenta el diagrama de base de datos del proyecto.

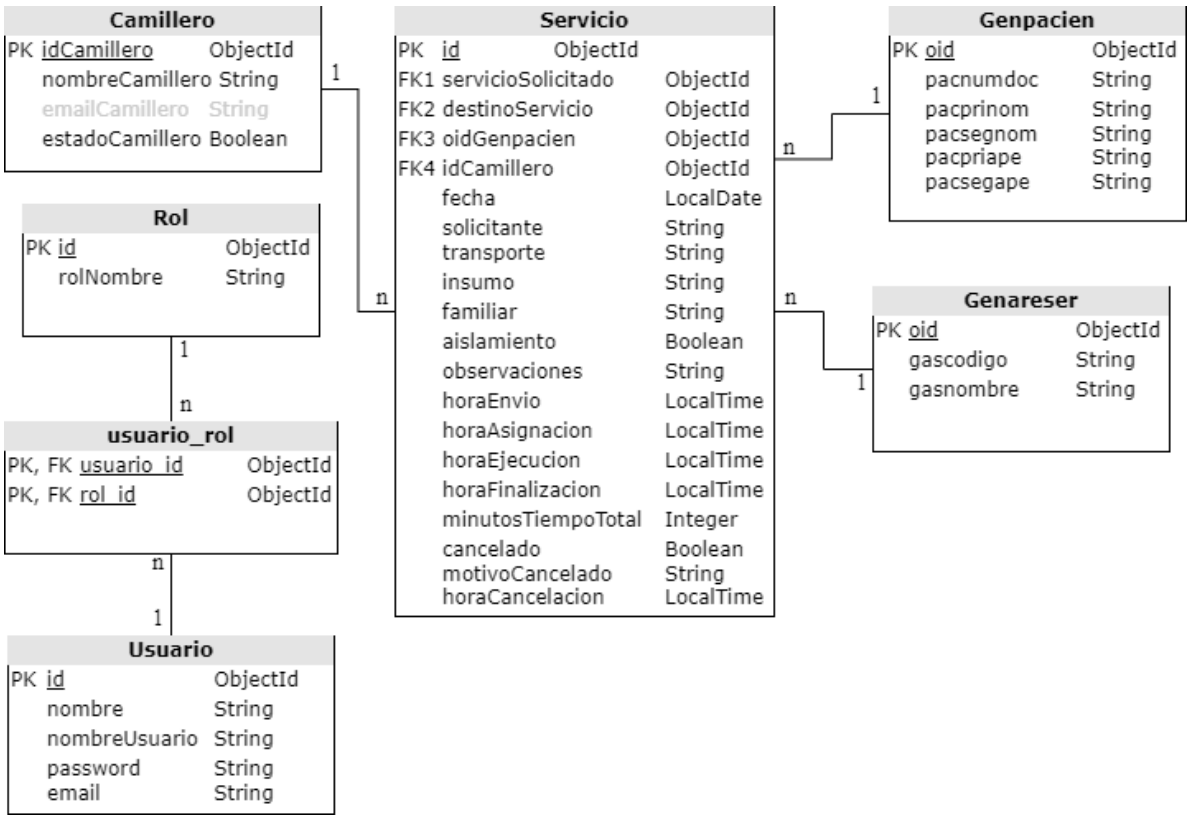

Figura 16. Diagrama de base de datos (ER)

5. Implementación

5.1. Tecnologías

Las tecnologías usadas en el proyecto fueron las siguientes:

5.1.1. Base de datos

- SQL Server Management Studio: esta aplicación fue usada para trabajar con la Base de Datos. Se usó un servidor para acceder a una base de datos de prueba del HUS.

5.1.2. Tecnologías Back-End

- Maven: v4.0.0
- Spring Boot: v2.5.6
- Java Development Kit (JDK): 14.0.2
- Java (JRE): 1.8.0_311
- IntelliJ IDEA: editor de texto usado para el BackEnd.
- Postman: como cliente http (client library).

5.1.3. Tecnologías Front-end

- Angular 12
- Angular HttpClient
- Bootstrap 5
- Visual Studio Code: versión 1.62.1 (user setup). Este editor de texto fue usado para construir el FrontEnd.
- Node JS: versión 14.16.0. El BackEnd no fue construido con Node sino con Spring Boot, pero la razón de haber usado Node JS es porque se necesitaba el gestor de paquetes npm para instalar Angular CLI (npm install -g @angular/cli).

5.1.4. Trazabilidad

- GitHub: se crearon dos repositorios, uno para el componente Back y otro para el componente Front y así poder llevar un control más ordenado; ServicioCamillerosBack y ServicioCamillerosFront respectivamente.

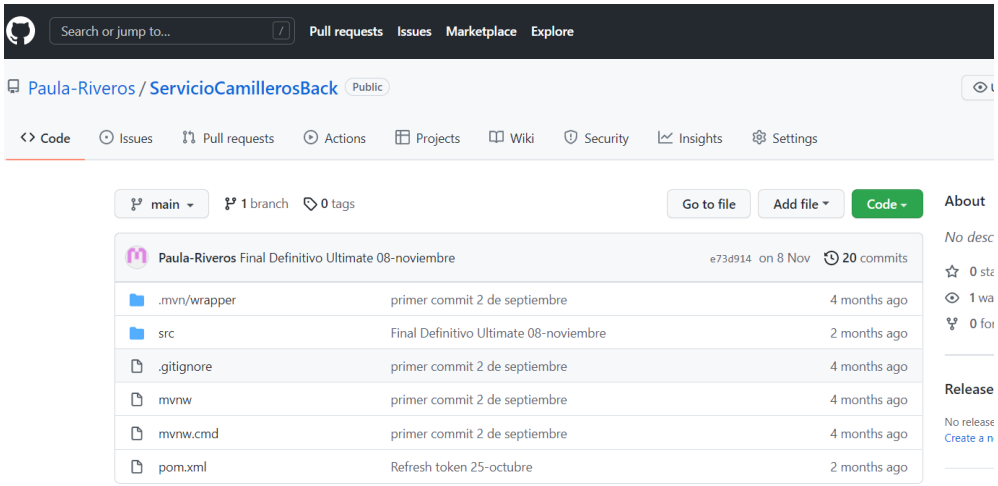

Figura 17. Control de versiones Back-End en GitHub

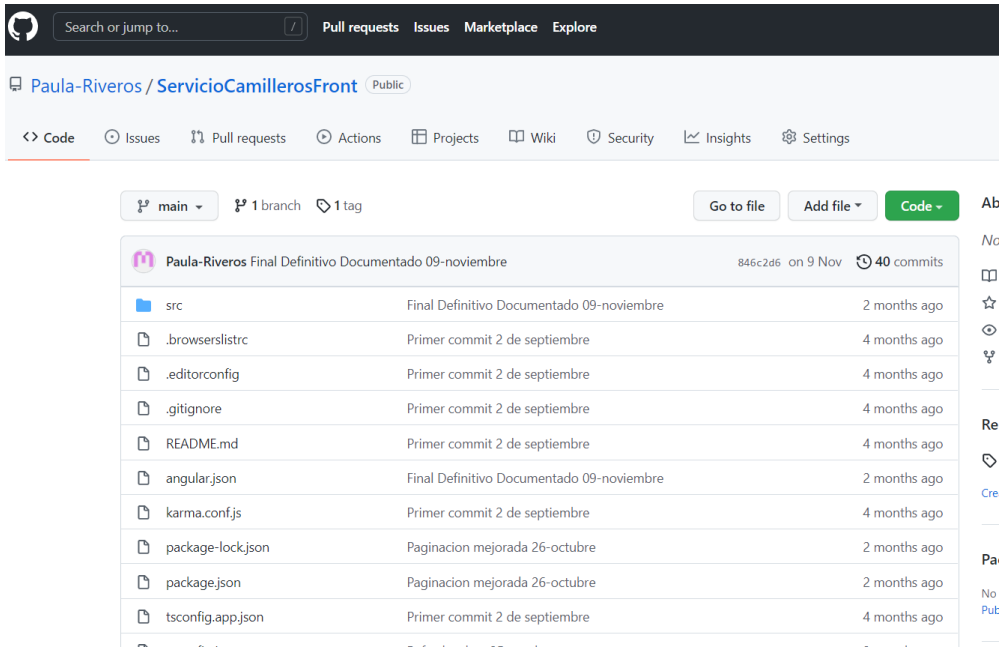

Figura 18. Control de versiones Front-End en GitHub

- Trello: esta se usó con el fin de organizar el proceso de la implementación del proyecto, y así saber lo que ya se había realizado y lo que faltaba por hacer.

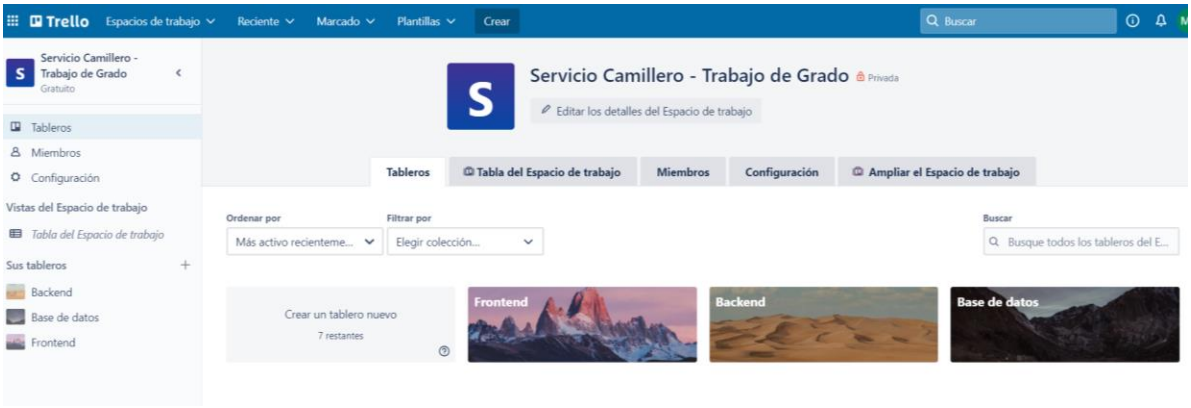

Figura 19. Control de tareas en Trello

5.2. Prototipos

La implementación se llevó a cabo según el cronograma establecido inicialmente y con ayuda de las tareas asignadas en la plataforma Trello. De esta manera se realizaron dos prototipos con el fin de dar cumplimientos a los requerimientos del proyecto.

5.2.1. Primer prototipo

Como lo muestra la tabla 5, en este primer prototipo se implementó la base de datos de prueba del HUS. Se desarrolló el componente back y el componente front del proyecto, en donde se agregaron los requerimientos necesarios para realizar un servicio de camilleros sin mayor detalle, es decir, solicitar, asignar y consultar servicios y gestionar camilleros. Adicionalmente, se implementó el registro e inicio de sesión de usuarios para la gestión de las actividades.

Tabla 5. Requerimientos funcionales desarrollados en el primer prototipo.

Requerimientos funcionales cumplidos
Base de datos de prueba del HUS
Solicitar servicio
Asignar camillero
Consultar servicios (Ver servicio)

Gestionar camilleros (Crear, modificar y ver camillero)
Registrarse, iniciar sesión y cerrar sesión

Se muestran las vistas de usuario generadas por el componente front-end para este primer prototipo.

Figura 20. Vista de usuario de la página de inicio al ingresar

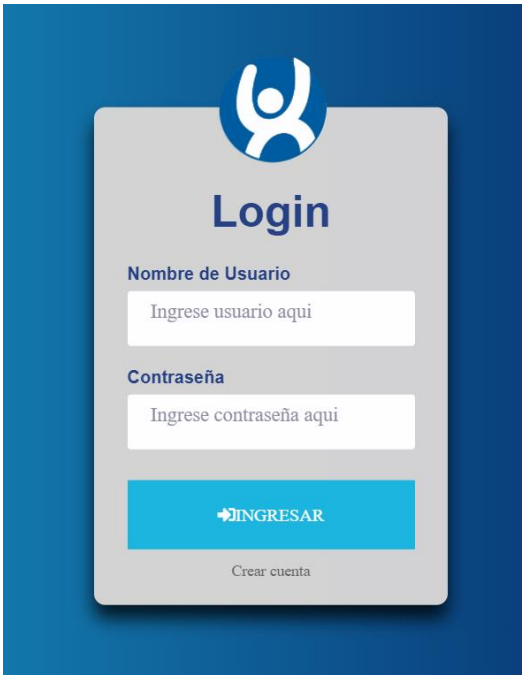

Figura 21. Vista de usuario de iniciar sesión

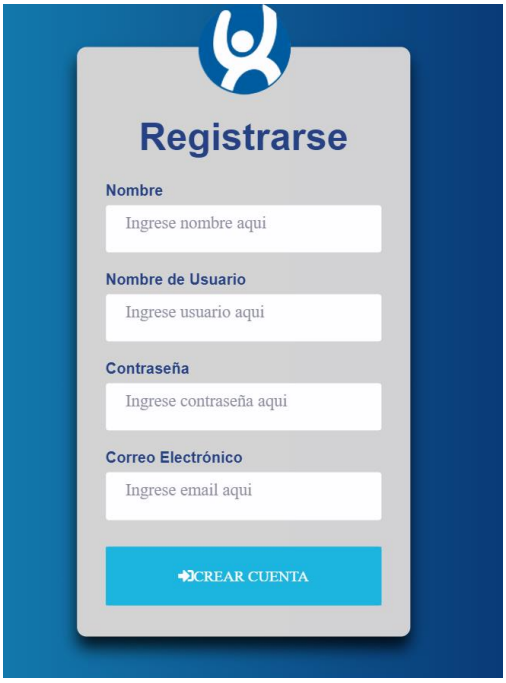

Figura 22. Vista de usuario de registrarse

5.2.2. Segundo prototipo

En este segundo prototipo y resultado final del proyecto se implementó el registro de las horas, el poder cancelar un servicio y se agregaron filtros para la consulta de los servicios. Se ajustaron algunos detalles en la interfaz de usuario y en algunos requerimientos del proyecto. Se modificaron algunos permisos de acuerdo con el tipo de usuario y se agregó un tipo de usuario al cual se le asignaron todos los permisos. Finalmente, se creó una base de datos de respaldo con el fin de llevar un proceso constante y aprovechar el tiempo en su totalidad, ya que en algunas ocasiones no se fue posible acceder a la base de datos de prueba del HUS debido a problemas técnicos con la base de datos y esto detenía el proceso de desarrollo.

Tabla 6. Requerimientos funcionales desarrollados en el segundo prototipo.

Requerimientos funcionales cumplidos
Registrar horas
Cancelar servicio
Imprimir servicio
Filtrar servicios
Permisos a cada tipo de usuario
Base de datos de respaldo

A continuación, se presentan las vistas de usuario creadas para el desarrollo de los requerimientos para el segundo prototipo.

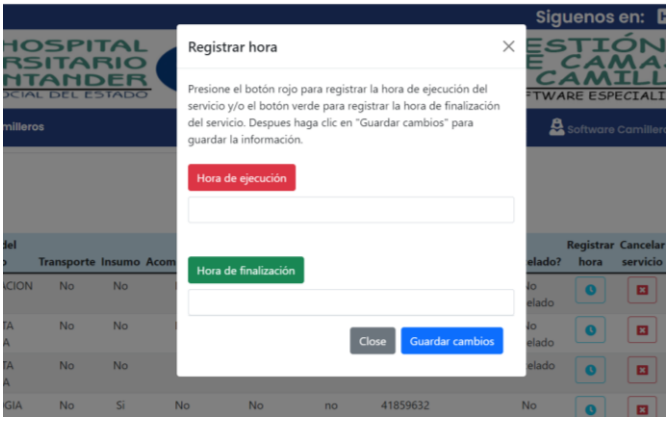

Figura 23. Vista de usuario de registrar las horas

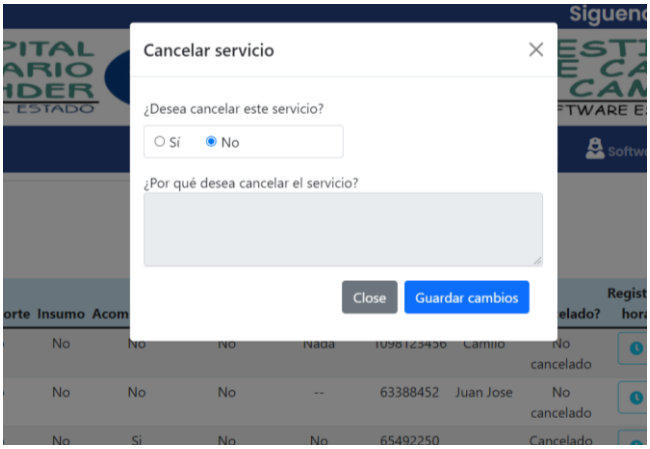

Figura 24. Vista de usuario de cancelar un servicio

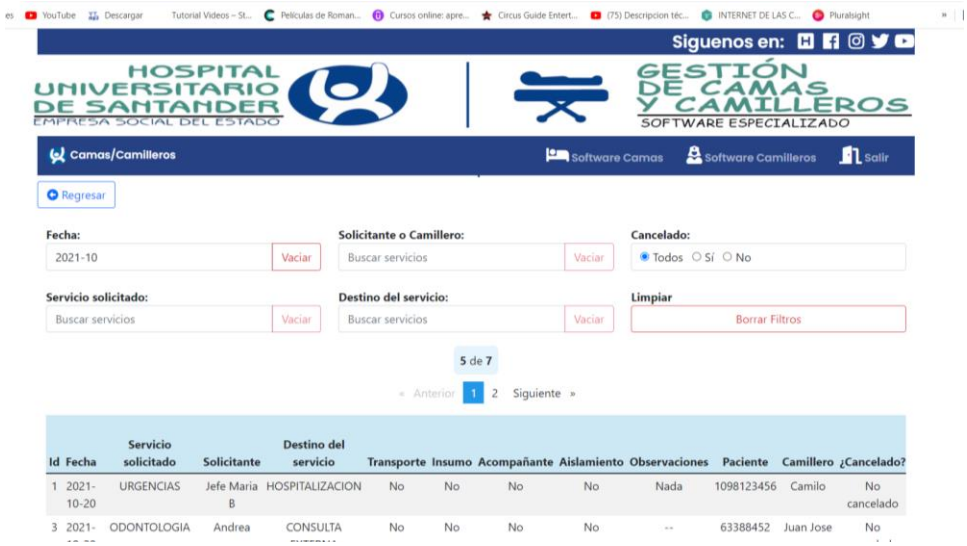

Figura 25. Vista de usuario de filtrar los servicios

6. Validación de la solución

Se diseñó un plan de pruebas para cada diagrama de actividades y se realizaron pruebas en tiempo real con usuarios del Hospital Universitario de Santander.

6.1. Pruebas de software

Tabla 7. Pruebas para registrarse (crear cuenta)

#	Usuario	Software	Datos	Revisar
1	Seleccionar el botón “Ingresar”	-	-	-
2	-	Mostrar formulario de Login	-	-
3	Dar clic donde dice “Crear cuenta”	-	-	-
4	-	Mostrar formulario de Registrarse	-	-
5	Llenar el formulario	-	Nombre: Paula Nombre de usuario: mariap Contraseña: mariap Email: mariap@hotmail.com	El email debe ser valido
6	Crear cuenta (botón crear cuenta)	-	-	-
7	-	Registrar el usuario	-	-
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Tabla 8. Pruebas para ingresar (iniciar sesión)

#	Usuario	Software	Datos	Revisar
1	Seleccionar el botón “Ingresar”	-	-	-
2	-	Mostrar formulario de Login	-	-
3	Llenar formulario	-	Nombre de usuario: mariap Contraseña: mariap	-

4	Ingresar (botón ingresar)	-	-	-
5	-	Mostrar página principal al inicio de sesión	-	-
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Tabla 9. Pruebas para cerrar sesión

#	Usuario	Software	Datos	Revisar
1	Seleccionar el botón “Salir”	-	-	-
2	-	Destruir token de autenticación	-	-
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Tabla 10. Pruebas para solicitar un servicio

#	Usuario	Software	Datos	Revisar
1	Seleccionar el botón “Solicitud”	-	-	-
2	-	Mostrar formulario	-	-
3	Llenar el formulario para pedir un servicio de camilleros	-	Fecha: 28/10/2021 Servicio solicitado: Laboratorio clínico Solicitante: Laura Destino: Urgencia adultos Transporte: No Insumo: No Acompañante: No Aislamiento: No Observaciones: --	-
4	Ingresar el número de documento del paciente y presionar Enter	-	Doc. Paciente: 28419493	No dejar ingresar un paciente que no exista

5	-	Mostrar nombre del paciente en el respectivo campo	Ana Francisca Mejía	-
6	Enviar servicio (Enviar)	-	-	-
7	-	Registrar el servicio	-	-
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Tabla 11. Pruebas para asignar un camillero a un servicio

#	Usuario	Software	Datos	Revisar
1	Seleccionar el botón “Asignación”	-	-	-
2	-	Mostrar servicios de la fecha actual	-	Se muestran los servicios de la fecha actual (esta se puede cambiar)
3	Seleccionar el botón “Editar” del respectivo servicio	-	-	-
4	-	Mostrar formulario diligenciado con los datos del servicio escogido	-	Todos los campos están bloqueados, solo está activo el del camillero.
5	Escoger al camillero	-	Jesús	Solo se muestran los camilleros activos
6	Actualizar servicio (Actualizar)	-	-	-
7	-	Actualizar datos del servicio	-	-
8	-	Mostrar los servicios de la fecha actual	-	-
9	Seleccionar el botón “Imprimir” del servicio asignado	-	-	-
10	-	Mostrar información del servicio	-	-

11	Imprimir servicio (Imprimir)	-	-	-
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Tabla 12. Pruebas para consultar la información de un servicio

#	Usuario	Software	Datos	Revisar
1	Seleccionar el botón “Consulta”	-	-	-
2	-	Mostrar servicios de la fecha actual	-	Se muestran los servicios de la fecha actual (esta se puede cambiar)
3	Seleccionar el botón “Ver” de algún servicio	-	-	-
4	-	Mostrar datos del servicio seleccionado	-	-
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Tabla 13. Pruebas para cancelar un servicio

#	Usuario	Software	Datos	Revisar
1	Seleccionar el botón “Asignación”	-	-	-
2	-	Mostrar servicios de la fecha actual	-	Se muestran los servicios de la fecha actual (esta se puede cambiar)
3	Seleccionar el botón “Cancelar” de un servicio	-	-	-
4	-	Mostrar ventana emergente de Cancelar servicio	-	-
5	Seleccionar “Si” para cancelar el servicio	-	-	-

6	Ingresar el motivo por el que se va a cancelar el servicio	-	Ya se realizó la solicitud por otro medio	-
7	Actualizar servicio (Guardar cambios)	-	-	-
8	-	Servicio cancelado	-	-
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Tabla 14. Pruebas para registrar la hora de ejecución y de finalización de un servicio

#	Usuario	Software	Datos	Revisar
1	Seleccionar el botón “Asignación”	-	-	-
2	-	Mostrar servicios de la fecha actual	-	Se muestran los servicios de la fecha actual (esta se puede cambiar)
3	Seleccionar el botón “Registrar hora” de un servicio	-	-	-
4	-	Mostrar ventana emergente de Registrar horas	-	-
5	Seleccionar el botón “Hora de ejecución”	-	09:45:40	-
6	Registrar hora (Guardar cambios)	-	-	-
7	-	Hora guardada	-	-
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Tabla 15. Pruebas para filtrar los servicios

#	Usuario	Software	Datos	Revisar
1	Seleccionar el botón “Consulta”	-	-	-

2	-	Mostrar servicios de la fecha actual	-	Se muestran los servicios de la fecha actual (esta se puede cambiar)
3	Ingresar algún dato en el/los filtro(s) adecuado(s)	-	-	-
4	-	Mostrar servicios de acuerdo con los datos ingresados en los filtros	-	-
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Tabla 16. Pruebas para agregar un nuevo camillero

#	Usuario	Software	Datos	Revisar
1	Seleccionar el botón “Camilleros”	-	-	-
2	-	Mostrar los camilleros que han hecho algún servicio	-	-
3	Seleccionar el botón “Nuevo camillero”	-	-	-
4	-	Mostrar formulario	-	-
5	Llenar el formulario para agregar un camillero	-	Número de documento: 123456 Nombre: Juan Pablo	Solo se permiten números en el campo “Número de documento”
6	Crear camillero (Crear)	-	-	-
7	-	Guardar camillero	-	-
8	-	Mostrar los camilleros	-	-
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

Tabla 17. Pruebas para editar la información de un camillero

#	Usuario	Software	Datos	Revisar
1	Seleccionar el botón “Camilleros”	-	-	-
2	-	Mostrar los camilleros que han hecho algún servicio	-	-
3	Seleccionar el botón “Editar” de algún camillero	-	-	-
4	-	Mostrar formulario	-	-
5	Modificar los campos pertinentes	-	Estado: Inactivo	-
6	Actualizar camillero (Actualizar)	-	-	-
7	-	Actualizar camillero	-	-
8	-	Mostrar los camilleros	-	-
#	**Errores – Descripción**		**Corregido**	**Fecha de Corrección**

6.2. Pruebas con usuarios

Con el fin de tener una mayor certeza del funcionamiento de la aplicación se realizó una prueba piloto en el Hospital Universitario de Santander (HUS) junto con la jefe de enfermería encargada del servicio de camilleros. Esta prueba no se realizó en un ambiente de producción ya que se ponía en riesgo la información y se podían retrasar las actividades en el caso en que ocurriera algún fallo. Sin embargo, el proyecto se desplegó en un servidor del HUS y desde allí se realizó dicha prueba con datos reales, pero usando la misma base de datos de prueba que se usó durante el desarrollo del proyecto, obteniendo resultados favorables en cuando a la ejecución de los servicios,

permitiendo registrar la información requerida y cumpliendo de manera completa los servicios de camilleros ejecutados con la aplicación.

7. Conclusiones

Se logró desarrollar la aplicación de manera exitosa, cumpliendo con los objetivos planteados y satisfaciendo las necesidades de los interesados.

Durante el desarrollo del proyecto se aplicaron los conocimientos adquiridos durante la carrera, aplicando la metodología necesaria y dedicando el tiempo necesario para cada una de sus fases; análisis, diseño, codificación y pruebas.

Por medio del plan de pruebas fue posible detectar errores tempranamente, evitando así una crisis del software y dando como resultado final un prototipo funcional en los términos que se establecieron en los requerimientos.

La trazabilidad tuvo un rol importante permitiendo mantener el control en cuanto al desarrollo de la aplicación.

Esta idea de solución a la problemática inicial es un punto de partida muy importante para la evolución de la ejecución de los procesos de los servicios de camilleros en el Hospital Universitario de Santander, los cuales, actualmente, se llevan a cabo de manera manual.

Este proyecto hace un gran aporte de una herramienta tecnológica en la web, la cual permitirá la trazabilidad de los camilleros, así como proveerá información de utilidad para el análisis de los servicios de camilleros con los datos recolectados, logrando una mejora a la calidad del servicio de camilleros prestados en el HUS.

8. Trabajo futuro

Para futuras versiones o ajustes se puede tener en cuenta la implementación del cambio de contraseña a partir del correo electrónico del usuario.

Implementar un aplicativo móvil en el cual los camilleros puedan dar por finalizado un servicio y revisar el o los siguientes servicios que se les asignen, de esta manera se ahorra el tiempo que demora el camillero en regresar a la central de camilleros para recibir un nuevo servicio.

Poner en producción la aplicación, a la cual se pueda acceder mediante la página oficial del HUS.

Referencias Bibliográficas

Codmind. 2021. ¿Qué es Spring Boot?. [online] Available at: <https://blog.codmind.com/que-es-spring-boot/> [Accessed 6 December 2021].

bezkoder. (2021, mayo 15). Angular 12 + Spring Boot + MySQL example: Build a CRUD app. BezKoder. https://www.bezkoder.com/angular-12-spring-boot-mysql/

Social network for programmers and developers. (s. f.). Morioh.com. Recuperado 16 de diciembre de 2021, de https://morioh.com/p/feefdac0620f

Maheshwari, N. (2017, agosto 29). Build faster web applications with AngularJs framework [2020]. Nmgtechnologies.Com. https://nmgtechnologies.com/blog/web-application-development-with-angularjs.html

Mora, S. L. (2021, junio 8). ¿Qué son las Single-Page Application (SPA)? El desarrollo elegido por Gmail y LinkedIn. DIGITAL55. https://www.digital55.com/desarrollo-tecnologia/que-son-single-page-application-spa-desarrollo-elegido-por-gmail-linkedin/

(S. f.). Com.br. Recuperado 4 de enero de 2022, de https://mv.com.br/es/blog/los-10-pasos-para-una-gestion-hospitalaria-de-exito--

Gestión Hospitalaria. (2016, diciembre 3). Organización para la Excelencia de la Salud – OES. https://oes.org.co/gestion-hospitalaria/

de OpenClassrooms, M. (2017, septiembre 11). OpenClassrooms ES. OpenClassrooms ES. https://blog.openclassrooms.com/es/2017/09/11/que-es-el-desarrollo-web/

Modelo integral de gestión hospitalaria en México. (s. f.). Elhospital.com. Recuperado 4 de enero de 2022, de https://www.elhospital.com/temas/Debemos-fortalecer-el-modelo-integral-de-gestion-hospitalaria,-Presidente-de-la-Asociacion-Mexicana-de-Hospitales+133532

admin. (s. f.). ¿Qué es la arquitectura de software? Jucaripo. Recuperado 4 de enero de 2022, de https://jucaripo.com/que-es-la-arquitectura-de-software/

García, M. (s. f.). MVC (Modelo-Vista-Controlador): ¿qué es y para qué sirve? Codingornot.com. Recuperado 4 de enero de 2022, de https://codingornot.com/mvc-modelo-vista-controlador-que-es-y-para-que-sirve

Bibliografía

Blokdyk, G. (2018). IBM docs: Complete self-assessment guide. Createspace Independent Publishing Platform.

Noguera, B. (2015, enero 29). ¿Qué es un diagrama de clases? Culturación. https://culturacion.com/que-es-un-diagrama-de-clases/

Wikipedia contributors. (s. f.-a). Modelo de prototipos. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Modelo_de_prototipos&oldid=139152201

Wikipedia contributors. (s. f.-b). Requisito funcional. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Requisito_funcional&oldid=137963774

(S. f.-a). Medium.com. Recuperado 11 de enero de 2022, de https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a

(S. f.-b). Edu.ar. Recuperado 11 de enero de 2022, de https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

REST vs WebSocket. ¿Hay diferencias? (s. f.). Chiyanasimoes.com. Recuperado 12 de enero de 2022, de https://www.chiyanasimoes.com/blog/rest-vs-websocket-hay-diferencias

Velasco, R. (2020, marzo 13). Visual Studio Code: el editor de código de Microsoft que querrás instalar. SoftZone. https://www.softzone.es/programas/utilidades/visual-studio-code/

Chacón, J. L. (2021, octubre 25). TypeScript: qué es, diferencias con JavaScript y por qué aprenderlo. Profile Software Services. https://profile.es/blog/que-es-typescript-vs-javascript/

Gacelaweb. (2020, octubre 26). Qué es node.js y para qué sirve. Gacelaweb. https://www.gacelaweb.com/que-es-nodejs-y-para-que-sirve/

¿Qué es Apache y para qué sirve? (2019, enero 8). Ayuda | dinahosting. https://dinahosting.com/ayuda/que-es-apache-y-para-que-sirve/

Funcionalidades - IntelliJ IDEA. (s. f.). JetBrains. Recuperado 14 de enero de 2022, de https://www.jetbrains.com/es-es/idea/features/

Apéndices

Apéndice A. *Diagramas de actividades complementarios*

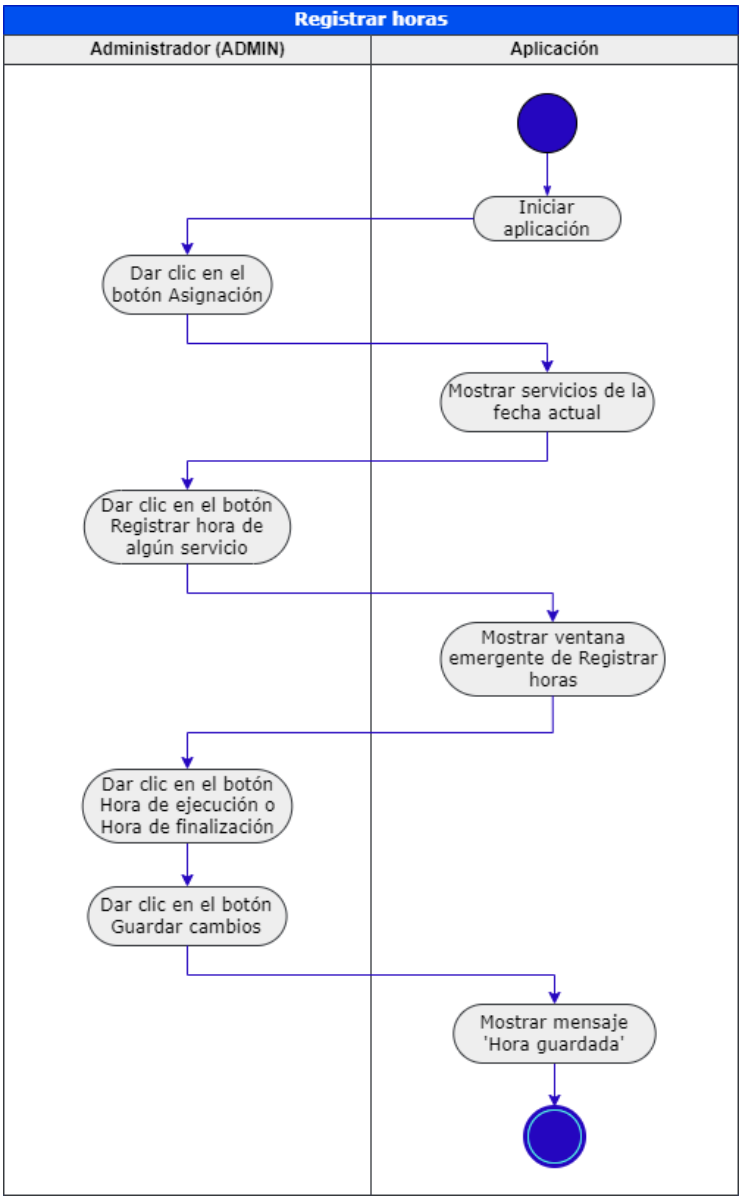

Figura 26. Diagrama de actividades de registrar horas

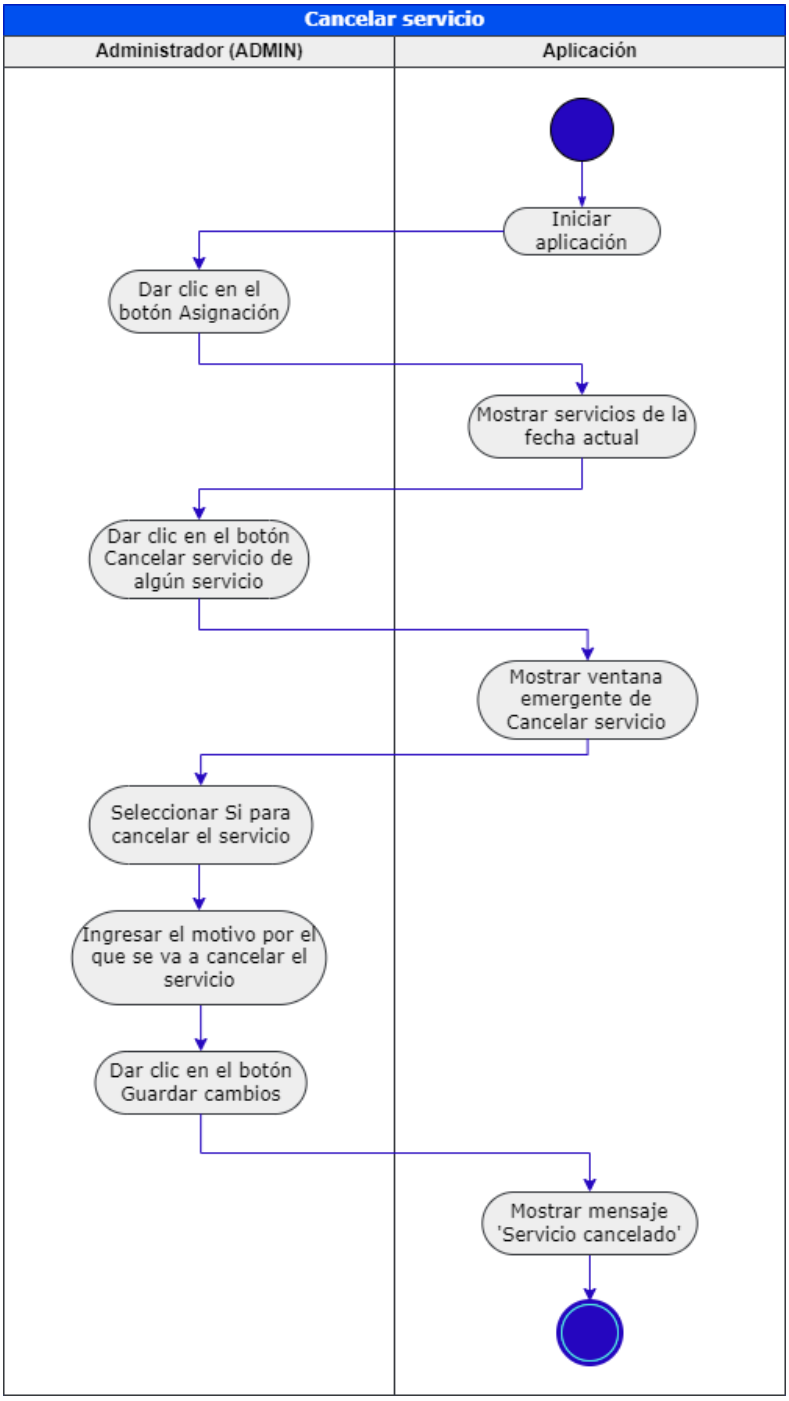

Figura 27. Diagrama de actividades de cancelar un servicio

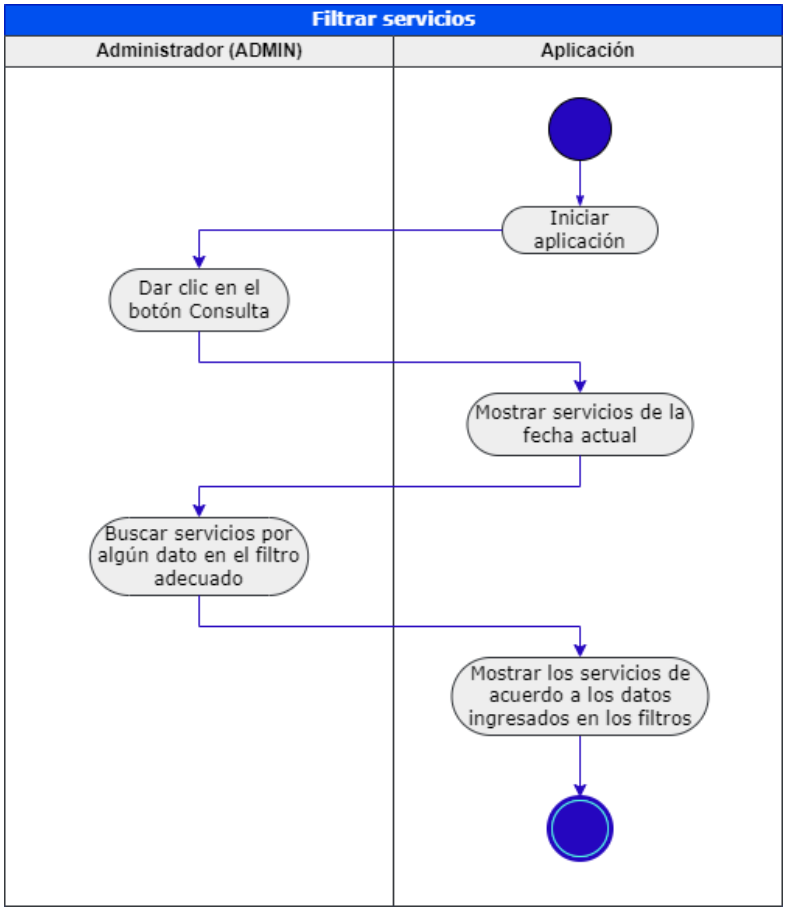

Figura 28. Diagrama de actividades de filtrar servicios

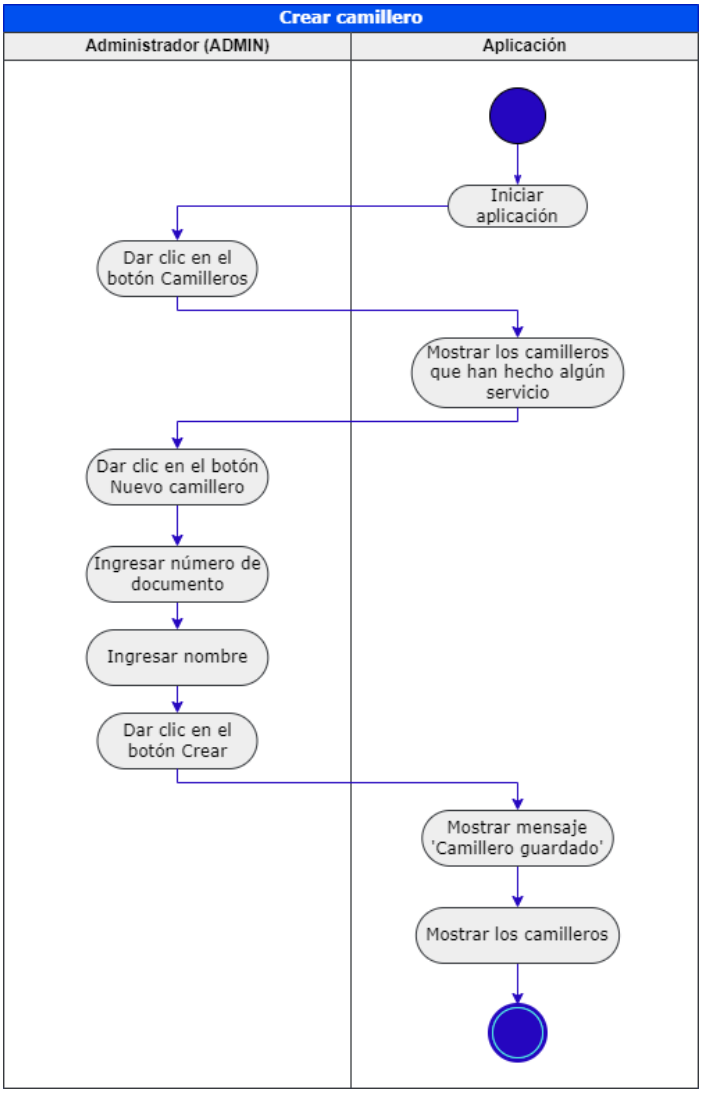

Figura 29. Diagrama de actividades de crear un camillero

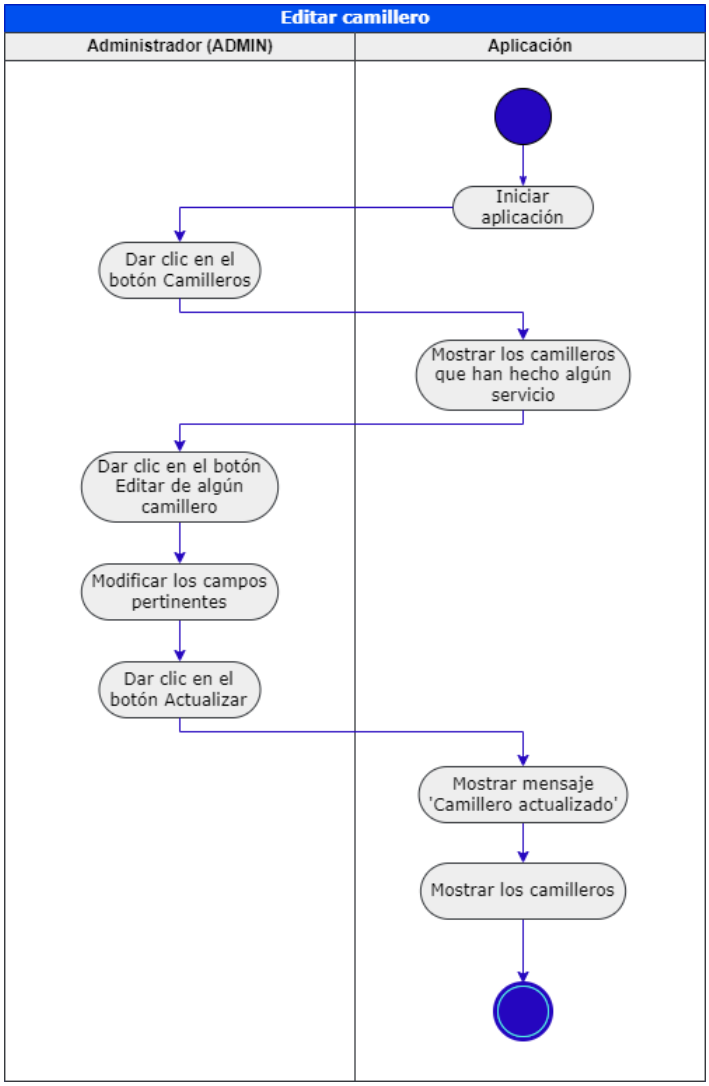

Figura 30. Diagrama de actividades de editar un camillero