ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UNA APLICACIÓN WEB Y MÓVIL PARA LA GESTIÓN DE LA RESIDENCIA UNIVERSITARIA ILINIZAS UTILIZANDO PRÁCTICAS DEVOPS

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

MUÑOZ GALVÁN GORKY ANDRÉS
gorky.munoz@epn.edu.ec

DIRECTORA: Pamela Catherine Flores Naranjo, Ph.D.
pamela.flores@epn.edu.ec

Quito, septiembre 2021

DECLARACIÓN

Yo, Gorky Andrés Muñoz Galván declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Gorky Andrés Muñoz Galván

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Gorky Andrés Muñoz Galván, bajo mi supervisión.

PAMELA CATHERINE FLORES NARANJO, Ph.D.

DIRECTORA DE PROYECTO

DEDICATORIA

A mis padres, quienes han sido un pilar fundamental en mi desarrollo personal y académico. Nunca me abandonaron ni dejaron que me dé por vencido, siempre creyeron en mi potencial y me ayudaron a llevarlo al máximo. Sin su apoyo, no estaría en el lugar al que he llegado. Amo a mis padres y les dedico esta meta que juntos sudamos para conseguirla.

AGRADECIMIENTOS

A mis padres Jacqueline y Gorky, quienes desde que se enteraron de mi existencia, han puesto todo su esfuerzo en darme la mejor crianza personal, académica y espiritual. Gracias a sus valores, experiencias y enseñanzas me he convertido en una persona capaz de manejar su día a día de manera adecuada. Quiero agradecer el gran esfuerzo que hicieron al dejar a su único hijo, partir a otra ciudad, para cumplir con su sueño; día a día son mi inspiración, mis mejores amigos y consejeros.

A mi abuelito Humberto, a quien considero mi segundo padre. Agradezco que nunca me haya dejado solo, incluso después de su partida, sigue conmigo protegiéndome, bendiciéndome y siendo mi ángel de la guarda.

A mi novia Sofía por apoyarme, amarme y motivarme desde el primer día. Son tantas las enseñanzas que me has dado que mi vida ha tomado un nuevo rumbo desde tu llegada, gracias a ti puedo desenvolverme mucho mejor como ser y persona. A su vez, tu compañía, amor y lealdad han sido un pilar fundamental en mi vida. Agradezco poder estar junto a una compañera tan leal, amorosa y entregada a sus seres amados.

A mis amigos Jamil, Juan Andrés, Nicolás, Daniel, David, José Nicolás, Sofía y Susana quienes fueron una gran compañía mientras cumplía mi trayecto en la universidad. Siempre me supieron apoyar y no permitieron que desista.

A mi querido amigo Juan Pablo, quien en vida velaba por mi bienestar y cumplir con este sueño de ser ingeniero, fue una promesa a su memoria.

A todos mis amigos de la residencia Ilinizas con quienes llegué a tener un vínculo familiar ya que los veía día a día y soportaban mi actitud cuando estaba de malas, gracias por acompañarme en este trayecto.

A mi directora de tesis, Doctora Pamela, le agradezco la guía, constancia y dedicación que brindó a mi proyecto de titulación para que el mismo haya sido finalizado de la mejor manera posible.

A mis compañeros y amigos de la universidad, con quienes aprendí a volver a confiar, estoy muy feliz de haber realizado este camino junto a ustedes.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

RESUMEN

La Residencia Universitaria Ilinizas (RUI) es una residencia en la ciudad de Quito que brinda una variedad de servicios a sus residentes para que se puedan centrar en sus actividades académicas sin distracción. Entre los servicios que brinda la residencia se tiene las comidas, lavandería, limpieza, eventos extracurriculares, retiros espirituales, acción social y atención espiritual con un sacerdote. Sin embargo, toda la información y gestión eran realizadas de forma manual; existían ocasiones en las que la información era modificada según convenga. El registro de comidas era el más vulnerable, a diario tenía modificaciones y no se podía saber quién las hizo. Esto causaba que se desperdicie comida y que aumenten los gastos. Razón por la cual el objetivo del presente trabajo de titulación fue desarrollar un sistema que permita gestionar los servicios de comidas, eventos y residentes, brindando un mayor control a la información y, por ende, evitar gastos dados por modificaciones de esta.

Se desarrolló el proyecto utilizando el marco de trabajo SCRUM, por lo que se realizaron entregas parciales del producto las cuales eran potencialmente desplegables y con objetivos que permitían que el sistema sea puesto en producción. Adicionalmente, se implementaron prácticas DevOps, las cuales permitieron automatizar los procesos de entrega y despliegue, por lo que, el tiempo designado a estas tareas repetitivas fue aprovechado en tiempos de desarrollo.

Se desarrolló las aplicaciones web utilizando el framework Angular. Mientras que la aplicación servidor se desarrolló con el framework Sails.js en el que se utilizó el lenguaje de programación Javascript. La aplicación móvil se desarrolló empleando el framework Flutter para conseguir una aplicación multiplataforma. Para la base de datos se usó un contenedor Docker con la imagen de MySQL. El manejo de control de versiones se lo realizó con GitLab en el cual se definió las configuraciones DevOps. Finalmente, Firebase fue utilizado como hosting de las aplicaciones web clientes y Heroku para la aplicación servidor.

El uso de la metodología ágil para la implementación y de las prácticas DevOps para la automatización del proyecto, permitió tener una alta flexibilidad con el cliente, dando la posibilidad de agregar o modificar funcionalidades en el transcurso del desarrollo. Al finalizar el proyecto, se cumplió todos los requerimientos iniciales y nuevos del sistema. Además, se realizó una encuesta de usabilidad la cual mostró que más del 65% de los usuarios definieron a las funcionalidades del sistema como simples y fáciles de entender. Por lo que se concluyó que se desarrolló el proyecto de manera satisfactoria.

Palabras Clave: Gestión de residentes, Gestión de comidas, Gestión de eventos, Aplicación Web, Aplicación Móvil, Aplicación Multiplataforma.

ABSTRACT

The Residencia Universitaria Ilinizas (RUI) is a residence in the city of Quito that provides a variety of services to its residents, giving them the chance to focus on their academic activities without distraction. Among the services provided by the residence are meals, laundry, cleaning, extracurricular events, spiritual retreats, social action, and spiritual care with a priest. They management and information were made manually; There were occasions when the information was modified as needed. The food registry was the most vulnerable, it had modifications every day and it was not possible to know who made them. This caused food to go to waste and expenses to increase. The objective of this degree project was to develop a system that allows managing food services, events and residents, providing greater control of the information and, therefore, avoiding expenses due to its modifications.

The project was developed and organized using SCRUM framework, so the product had partial deliveries which were potentially deployable and with objectives that allowed the system to be put into production. Additionally, DevOps practices made it possible to automate the delivery and deployment processes, therefore, the time allocated to these repetitive tasks was taken advantage of in development times.

We used Angular framework to develop the web applications. While the server application was developed with the Sails.js framework in which we used Javascript as programming language. The mobile application was developed using the Flutter framework to achieve a cross-platform application. For the database we used a Docker container with a MySQL image. The version control management was done with GitLab and, we use it to configure DevOps practices in each repository. Finally, we used Firebase to host the client web applications and Heroku for the server application.

The implementation of agile methodology and DevOps, allowed us to have a high flexibility with the client, giving us the possibility of adding or modifying functionalities during the development. We met all requirements, initials, and news by the end of the project. Therefore, a usability survey was carried out which showed that more than 65% of users defined the system's functionalities as simple and easy to understand. Therefore, we concluded that the project was developed satisfactorily.

Key words: Resident management, Food management, Event management, Web application, Mobile application, Cross-platform application

CAPÍTULO 1. INTRODUCCIÓN

1.1 ANTECEDENTES

Ilinizas es una Residencia Universitaria para varones en la ciudad de Quito. Fue fundada en el año de 1957, siendo la primera Residencia Universitaria en Quito y en todo el país. Desde entonces ha acogido a cientos de personas procedentes de distintas provincias del país y también extranjeros. Actualmente Ilinizas es administrada por la Corporación de Cultura Ilinizas, la cual es una institución sin fines de lucro orientada a la formación académica, espiritual y humana de jóvenes universitarios. La formación espiritual y humana está confiada a la Prelatura del Opus Dei [1].

1.2 PLANTEAMIENTO DEL PROBLEMA

La Residencia Universitaria Ilinizas brinda varios servicios que permiten a los residentes sentirse en un ambiente familiar y enfocarse en sus estudios. Dentro de los servicios que ofrece Ilinizas se tiene: servicio de lavado y planchado de ropa, alimentación diaria (desayuno, almuerzo, té y cena), eventos extracurriculares como paseos, excursiones y retiros espirituales, acción social y atención espiritual a quien lo desee.

Actualmente Ilinizas realiza la gestión de todos sus servicios de manera física y manual, por lo que, se ha visto en la necesidad de automatizar la gestión de estos, para de esta manera, brindar mayor comodidad a sus residentes y por consecuencia, tener una gestión más eficiente de la residencia. La gestión abarca los procesos de postulación de una persona para entrar a la residencia, el registro de comidas diarias, el registro de prendas de vestir para el servicio de lavandería y la coordinación de eventos extracurriculares.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

- Desarrollar una aplicación web y móvil que permita gestionar la Residencia Universitaria Ilinizas.

1.3.2 OBJETIVOS ESPECÍFICOS

- Recolectar requerimientos de cada proceso en base a entrevistas realizadas con los interesados para poder generar historias de usuario que permitan tener un producto que satisfaga las necesidades de los interesados.
- Diseñar la aplicación web utilizando el patrón de arquitectura modelo vista controlador
- Diseñar *mockups* y un mapa de navegación de la aplicación móvil que permitan definir la mejor arquitectura para la aplicación móvil.

- Implementar una aplicación móvil multiplataforma con el fin de que los residentes puedan hacer uso.
- Evaluar la satisfacción mediante el uso de encuestas de usabilidad a los usuarios finales.

CAPÍTULO 2. MARCO TEÓRICO

2.1. SCRUM

Scrum es un marco de trabajo utilizado en el desarrollo y mantenimiento de productos complejos [2]. Está basado en el proceso de desarrollo de software incremental y está compuesto por el equipo, eventos y artefactos. Cada componente sirve a un propósito específico y es esencial para su correcto uso. El proceso empieza con la recolección de requerimientos los cuales son conocidos como el producto backlog, posterior a esto se debe priorizar estos requerimientos, para dar inicio con el desarrollo. El ciclo de desarrollo está dividido en una serie de iteraciones llamadas sprint, las cuales tienen como objetivo entregar un producto terminado, utilizable y potencialmente desplegable [2].

2.1.1 Equipo Scrum

El equipo Scrum está conformado por el dueño del producto (P*roduct Owner*), el equipo de desarrollo (*Development Team*) y *Scrum Master*. El modelo de equipo Scrum está diseñado con el objetivo de brindar una mayor flexibilidad, creatividad y productividad; razón por la cual los equipos Scrum son autoorganizados y multifuncionales [2].

- **Dueño del Producto (Product Owner)**

El Dueño del Producto (*Product Owner)* es el encargado de maximizar el retorno de la inversión (ROI) identificando las características del producto y traduciéndolas en una lista del producto priorizada. [3] Es la única persona encargada de expresar claramente los elementos de dicha lista, asegurándose que cada elemento sea entendido por el equipo de desarrollo para así optimizar el valor del trabajo desempeñado. El dueño del producto es una sola persona, que representa los deseos e interés de un comité o grupo de interesados. Por lo que, se deben respetar las decisiones tomadas por él, ya que se ven reflejadas en la priorización y contenido de la lista del producto.

- **Equipo de Desarrollo (Development Team)**

El equipo de desarrollo son los encargados de desarrollar el producto y de entregar un incremento del producto terminado, potencialmente pueda ser puesto en producción, al finalizar cada sprint [2]. Los equipos de desarrollo cuentan con un alto nivel de autonomía, lo

que les permite organizar y gestionar de manera adecuada su trabajo. Son autoorganizados, por lo que ellos deciden como llevar a cabo cada elemento en la lista del producto. En Scrum no se definen títulos para cada miembro del equipo de desarrollo, todos cuentan con habilidades que permiten crear un incremento del producto, por lo que, la responsabilidad recae en el equipo de desarrollo como un todo y no de forma individual [4].

El tamaño óptimo del equipo de desarrollo es lo suficientemente pequeño como para permanecer ágil y lo suficientemente grande como para completar una cantidad de trabajo significativa [2]. Por lo que, el tamaño recomendable del equipo es de 3 a 9 personas. Un equipo con menos de 3 personas reduce la interacción y resulta en ganancias de productividad más pequeñas. Mientras que un equipo con más de 9 personas requiere mucha coordinación, haciendo que sea difícil la gestión mediante un proceso empírico.

- **Scrum Master**

El *Scrum Master* es el responsable de ayudar y asegurar que Scrum sea entendido y adoptado de la manera correcta para así, cumplir con el objetivo del negocio [3]. El Scrum Master no es el jefe de los miembros del equipo Scrum, él está al servicio del equipo, ayuda a eliminar impedimentos, protege al equipo de distracciones externas, decidiendo que interacciones pueden ser de ayuda y cuáles no. Además, es el encargado de educar, entrenar y guiar al dueño del producto para encontrar técnicas que permitan gestiona la lista de producto de manera efectiva.

2.1.2 Eventos de Scrum

Se tienen eventos predefinidos los cuales permiten mantener una regularidad y minimizar reuniones no definidas por Scrum que podrían comprometer el tiempo del equipo. Todos los eventos son bloques de tiempo con una duración máxima [3]. Cada evento constituye una oportunidad para la inspección y adaptación de algún aspecto. Si no se cumple uno de los eventos, esto se da como resultado una falta de transparencia al momento de implementar la metodología.

- **Sprint**

Es considerado la pieza clave en Scrum, en el cual el equipo de desarrollo se encarga de crear un incremento terminado del producto, utilizable y potencialmente desplegable [2]. Una vez iniciado el Sprint no se pueden realizar cambios que lleguen a afectar el objetivo del Sprint, pero el alcance si puede ser renegociado.

Ya que en cada Sprint se entrega un producto terminado, pueden ser considerados como un proyecto que brindará lo necesario para que el negocio pueda generar valor del producto.

Están limitados a una duración máxima de un mes calendario, debido a que, si el Sprint llega a ser demasiado grande, el riesgo aumenta y por ende la complejidad, teniendo así un Sprint que podría llegar a fallar.

- **Reunión de Planificación de Sprint (Sprint Planning Meeting)**

El trabajo por realizar en un Sprint es planificado en la *Sprint Planning*, el equipo de desarrollo determinado que es lo que puede ser terminado en el sprint, los elementos seleccionados de la lista del producto para el Sprint son llamados Lista de pendientes del Sprint (*Sprint Backlog)* [2]. A continuación, el equipo Scrum en conjunto define un objeto del sprint (*Sprint Goal*), el cual deberá lograrse durante el sprint y proveerá una guía de por qué se está construyendo el incremento del producto.

- **Objetivo del Sprint (Sprint Goal)**

Es una meta propuesta para el Sprint en curso, la cual el equipo debe tener en mente al momento de realizar la construcción para así satisfacer la implementación de la funcionalidad de manera adecuada.

- **Scrum Diario (Daily Scrum)**

Es una reunión diaria con una duración de 15 minutos con el objetivo de que el equipo de desarrollo esté al tanto de sus avances y actividades para así, crear un plan para las siguientes 24 horas. En esta reunión cada miembro del equipo de desarrollo debe responder a las siguientes preguntas [2]:

- ¿Qué hice ayer que ayudó al equipo a lograr el objetivo del sprint?
- ¿Qué haré hoy para ayudar al equipo a lograr el objetivo del sprint?
- ¿Veo algún impedimento que evite que el equipo o yo logremos el objetivo del sprint?

Esta reunión permite mejorar la comunicación entre el equipo de desarrollo, permite identificar impedimentos que pongan en riesgo la fecha de entrega o el objetivo del sprint y permite mejorar el nivel de conocimiento del equipo de desarrollo.

- **Revisión de Sprint (Sprint Review)**

Al finalizar cada Sprint se lleva a cabo la revisión de Sprint, en el cual se hace una inspección del incremento terminado. Es una reunión informal en la cual el equipo de desarrollo demuestra el trabajo que ha sido terminado y se responden preguntas acerca del incremento. Sin embargo, este no es el único propósito, también la clave de esta reunión es la conversación entre el equipo para identificar qué es lo que fue bien durante el sprint, que

problema ocurrieron y como se resolvió los problemas [3]. En esta reunión se define la fecha de la siguiente Reunión de Planificación de Sprint.

- **Retrospectiva de Sprint (Sprint Retrospective)**

Es una oportunidad para que el Equipo Scrum pueda inspeccionar en cómo se han adaptado a los procesos y ambientes. En este evento se deben discutir las prácticas que están funcionando y cuáles no, para así crear un plan de mejora que sea llevado a cabo en el siguiente sprint [2].

2.1.3 Artefactos de Scrum

- **Lista de Producto (Product Backlog)**

La lista de producto es una lista ordenada y priorizada que contiene todas las características, funcionalidades, mejoras y correcciones del producto. La lista del producto existe mientras el producto exista y evoluciona al igual que lo hace el producto, para así, lograr que el producto sea competitivo y genere valor en el mercado [2].

La lista de producto contiene ítems, los cuales pueden ser expresados de diferentes maneras, tales como historias de usuario, casos de uso o cualquier otra estrategia que el equipo Scrum crea que es útil y entendible [4].

- **Lista de Pendientes del Sprint (Sprint Backlog)**

La lista de pendientes del sprint son los elementos de la lista de producto seleccionados por el equipo de desarrollo para que formen parte del siguiente Sprint. Esta lista de pendientes del Sprint permite validar el progreso del equipo de desarrollo en el transcurso del Sprint. El cumplimiento de esta lista de pendientes permite cumplir el objetivo del Sprint [3].

2.2. DevOps

DevOps es un framework que permite a personas y equipos practicar sus oficios de manera efectiva y duradera. DevOps no es solo una metodología de desarrollo, sino que está relacionada e influenciada por otras metodologías. Estas prácticas pueden abarcar desde el desarrollo de software hasta la automatización de la infraestructura y el lanzamiento continuo. El corazón de DevOps empieza con personas trabajando como equipos, no solo como grupos. [5]

Devops intenta llenar los vacíos que existen entre el equipo de desarrollo y el equipo de operaciones. Por esto, enfatiza la comunicación, colaboración e integración entre los desarrolladores de software y las operaciones de TI. Integración continua (CI), pruebas

continuas (CT) y la entrega continua (CD) son parte importante de la cultura DevOps. [6] Devops intenta llenar los vacíos que existen entre el equipo de desarrollo y el equipo de operaciones. Por esto, enfatiza la comunicación, colaboración e integración entre los desarrolladores de software y las operaciones de TI. Integración continua (I), pruebas continuas (CT) y la entrega continua (CD) son parte importante de la cultura DevOps. [6]

2.2.1 Entrega continua

La entrega continua es el proceso de implementación de la aplicación en cualquier entorno de modo automatizado y proporcionando retroalimentación continua para mejorar la calidad. [6]

2.2.2 Implementación y pruebas continuas

Las pruebas continuas son una parte muy importante del ciclo de vida de la aplicación de principio a fin. Esto envuelve las pruebas funcionales, performance y seguridad. [6]

2.2.3 Monitoreo continuo

El monitorio continuo es una parte fundamental, es deseable tener un seguimiento en casi todas las etapas para tener transparencia sobre todos los procesos. Esto también nos ayuda a resolver y detectar problemas rápidamente. [6]

2.3. Usabilidad

La usabilidad de un producto software se define como la capacidad que tiene el producto para ser entendido, aprendido, usado y que resulte llamativo para el usuario, cuando es usado a través de determinadas condiciones. El grado de usabilidad puede ser definido como una medida empírica y relativa de la usabilidad del producto. La medida empírica no se trata de opiniones o sensaciones, se basa en pruebas de usabilidad las cuales se efectúan en laboratorios o son observadas en el campo de trabajo. Mientras que la forma relativa representa un resultado ni bueno ni malo, si no que se basa en metas planteadas, por ejemplo, que un 80% de usuarios de un grupo seleccionado para estas pruebas, logró realizar con éxito una tarea X sin más ayuda que una guía rápida. [7]

2.4. Lenguajes de programación

Nombre	Descripción
TypeScript	Es un lenguaje de programación y un conjunto de herramientas que permite generar código Javascript. Fue creado y

	mantenido por Microsoft como un proyecto de código abierto [8].
JavaScript	Es un lenguaje de programación que es usado para diseñar y programar el comportamiento de una página web [9].
Dart	Es un lenguaje de programación desarrollado por Google y usado para construir aplicaciones dirigidas a diferentes plataformas como móvil, escritorio, web y backend [10].
CSS	Es un lenguaje de estilos que permite añadir estilos a documentos HTML. Describe como deben ser renderizados los elementos [11].
HTML	Es un lenguaje de marcas de hipertexto, el cual define la estructura y contenido de una página web [12].
YAML	Es un lenguaje de serialización de datos diseñado para ser entendido por personas y cumplir tareas comunes [13].

Tabla 1. Lenguajes de programación

2.5 Herramientas

Nombre	Descripción
Visual Studio Code	Es un editor de código que corre en un computador (Windows, macOS, Linux) el cual contiene varias extensiones para muchos lenguajes de programación [14].
Android Studio	Es el IDE oficial para desarrollar aplicaciones Android, fue desarrollado inicialmente por JetBrains y es mantenido por Google y Jetbrains en conjunto [15].
Docker	Es una plataforma para desarrollar y correr aplicaciones. Permite separar las aplicaciones de la infraestructura para realizar el desarrollo y entregas de forma más rápida [16].

Firebase	Firebase es un servicio backend (BaaS) desarrollado por Google que permite construir, mejorar y crecer aplicaciones móviles y web. [17]
GitLab	Es un proyecto de código abierto que permite a los equipos de desarrollo realizar el control de versiones y permite desarrollar de manera colaborativa [18].
DBeaver	Es una herramienta universal de base de datos, gratis y de código abierto dirigida para desarrolladores y administradores de bases de datos. [19]
Adobe XD	Es una herramienta desarrollada por Adobe para diseñar y crear prototipos de aplicaciones móviles y web. [20]
Heroku	Es una plataforma como servicio de computación que permite construir, ejecutar y gestionar aplicaciones en la nube. [21]
Postman	Es una plataforma colaborativa para el desarrollo de API. [22]

Tabla 2. Herramientas de desarrollo

2.6. Frameworks

Nombre	Descripción
Node.js	Es un entorno de ejecución para Javascript [23] construido con el motor de Javascript V8 creado por Google. [24]
Sails	Es un marco de trabajo web basado en el patrón de diseño modelo-vista-controlador (MVC) que permite la creación de API y aplicaciones web usando node.js. [25]
Angular	Es un marco de trabajo para el desarrollo de aplicaciones web de una sola página (SPA) desarrolladas en Typescript que. Desarrollado y mantenido por Google; y de código abierto. [26]

Flutter	Flutter es un kit de desarrollo de software creado por el Google para el desarrollo de aplicaciones móviles compiladas de forma nativa y aplicaciones web. [27]

Tabla 3. Frameworks

CAPÍTULO 3. PLANIFICACIÓN

En el plan de tesis se definió que el proyecto tendría una duración de 420 horas a lo largo de 5 meses. Estas horas fueron divididas de la siguiente manera:

- 40 horas para realizar el levantamiento de requerimientos con el director de la residencia.
- 30 horas dirigidas al levantamiento y preparación del entorno de desarrollo.
- 230 horas designadas para el desarrollo del proyecto.
- 20 horas para realizar encuestas de usabilidad.

En la implementación del proyecto, se realizó 3 reuniones mediante la plataforma *ZOOM* con el director de la residencia [28], en las cuales se logró obtener los requerimientos necesarios de la aplicación y de igual manera, todos los criterios de aceptación para cada requerimiento, estas entrevistas se encuentran en el apartado de Anexos.

Las horas que se definió para el levantamiento del entorno de desarrollo fueron plenamente aprovechadas y fueron dirigidas en gran parte a investigación de mejores prácticas con las herramientas escogidas para el desarrollo del proyecto.

Las 230 horas de desarrollo fueron divididas en un total de 3 meses. Por lo que, se definió que se tendría 6 *sprints* con una duración de dos semanas cada uno, para así, cumplir con el tiempo establecido en el plan de tesis. Sin embargo, si bien se mantuvieron los tiempos de desarrollo; por temas de viaje y tiempos del dueño del producto y equipo de desarrollo, la duración de la implementación se vio alargada a 5 meses por pausas que se tuvo que tomar.

Finalmente, se tomó unas horas más que las definidas para realizar las encuestas de usabilidad, debido a que en este tiempo no estaba considerado el desarrollo de los casos de prueba.

CAPÍTULO 4. IMPLEMENTACIÓN

En este capítulo se indica los roles, eventos y artefactos definidos para el proyecto, siguiendo lo que la metodología Scrum dicta y tal como se lo ha definido previamente.

Como primer paso en la implementación de la metodología, se definió los roles de los involucrados en el proyecto, tal como se muestra en la TABLA 4.

Roles Scrum	
Scrum Master	Pamela Flores
Dueño del producto (Product Owner)	Antonio Villacis
Equipo de desarrollo (Development Team)	Gorky Muñoz

Tabla 4. Roles Scrum

3.1. Product Backlog

Como se explicó en el capítulo anterior, se mantuvieron varias reuniones con el Dueño del Producto en las cuales, se expresó la necesidad de contar con una página web que permita dar a conocer la residencia, una aplicación móvil para los miembros de la residencia y un sistema que permita gestionar dicha aplicación móvil. Para satisfacer estas necesidades, se definió un conjunto de épicas, las cuales se detallan en la TABLA 5.

Código	Nombre	Descripción
TI01	Página Informativa	Como estudiante universitario, necesito tener información de la residencia Ilinizas a través de diferentes medios para así conocer sus beneficios, ubicación e historia.
TI02	Gestión de proceso de postulación	Como estudiante universitario, necesito registrar una inscripción para ser un postulante a una plaza en la residencia Ilinizas.
TI03	Portal de administración	Como administrador de la residencia Ilinizas, necesito un medio de administración, para gestionar los usuarios y la información de la aplicación móvil.
TI04	Aplicación móvil Ilinizas	Como usuario de Ilinizas, necesito una aplicación para cumplir con los registros de asistencia a las comidas, inscribirme en eventos y modificar mi información personal.
TI05	Gestión de asistencia a comidas	Como miembro de Ilinizas, necesito gestionar mis registros de asistencia a las comidas de cada día para así tener acceso al comedor.

Tabla 5. Épicas

Una vez definidas las épicas, se definió las historias de usuario y se obtuvo la lista del producto, la cual está conformada por la historia de usuario, la complejidad, el grupo de prioridad al que pertenece y el orden en el que serán implementadas las historias. Para la prioridad se usaron los valores mostrados en la TABLA 6.

Prioridad	Valoración
Baja	1
Media	2
Alta	3
Muy alta	4

Tabla 6. Valoración de prioridad

Para clasificar a las historias de acuerdo con su complejidad, se utilizó la técnica de Planning poker, basándose en el entendimiento, esfuerzo y riesgo para cada historia de usuario. Primeramente, se escogió la historia de usuario con menor tamaño (menor complejidad y esfuerzo) para ser usada como guía base - En este caso, la historia de usuario escogida fue la **HI25 – Crear pantalla de Splash**. Por lo que, la complejidad asignada a cada historia es una representación del tamaño de cada historia de usuario en comparación con la guía base.

Para priorizar las tareas definidas, se hizo énfasis los comentarios de los interesados y se analizó las tareas que más retroalimentación necesitan para una correcta entrega [29]. Dado esto, la técnica de priorización fue qué tareas se pueden implementar y adaptar a la residencia de manera gradual, independiente e incremental. Esto con el objetivo de reemplazar poco a poco la gestión manual de la residencia. Por ende, se empezó con las historias que no afectaron a los miembros de la residencia, sino a nuevos. Por lo tanto, la primera priorización fue la creación de la página informativa, para así, llegar a nuevos posibles interesados. A continuación, el proceso de inscripción y entrevista para nuevos miembros. Y finalmente, los procesos de servicios de la residencia y ajustes generales del proyecto.

A continuación, se muestra las historias de usuario agrupadas por épicas, si se desea una mayor descripción en las historias, dirigirse al ANEXO 2.

Épica	**Código**	**Nombre**	**Complejidad**	**Prioridad**
TI01	HI01	Crear sección general	13	4
	HI02	Crear formulario de inscripción	21	4
	HI03	Crear sección servicios	8	2
	HI04	Crear formulario de contacto	5	1

TI02	HI05	Registrar información obligatoria del postulante	13	4
	HI06	Registrar información académica del postulante	5	4
	HI07	Registrar información de intereses del postulante	5	4
	HI08	Registrar información de los titulares del postulante	8	4
	HI09	Notificar por correo electrónico el registro de la postulación	8	2
TI03	HI10	Iniciar sesión en el sistema	8	4
	HI11	Consultar postulante	2	4
	HI12	Crear residente	8	4
	HI13	Consultar residente	2	3
	HI14	Modificar residente	8	3
	HI15	Eliminar residente	5	3
	HI16	Crear administrador	13	3
	HI17	Consultar administrador	5	3
	HI18	Modificar administrador	8	3
	HI19	Asignar tutores a residentes	8	2
	HI20	Eliminar administrador	8	1
	HI21	Crear evento	3	1
	HI22	Consultar evento	3	1
	HI23	Modificar evento	5	1
	HI24	Eliminar evento	5	1
TI04	HI25	Crear pantalla de Splash	1	4
	HI26	Crear pantalla de Inicio	13	4
	HI27	Iniciar Sesión	13	4
	HI28	Barra de navegación dinámica	13	4
	HI29	Sección gestión de registro de comidas	21	4
	HI30	Sección gestión de eventos	13	4
	HI31	Olvidó contraseña	8	3
	HI32	Sección perfil	8	3
	HI33	Cerrar sesión	1	3

	HI34	Editar información personal	3	2
	HI35	Registro de usuario en un evento	5	2
	HI36	Eliminar registro de usuario en un evento	5	2
TI05	HI37	Registrar comida diaria	5	4
	H38	Editar comida diaria	13	4
	H39	Reportería de comidas diarias	13	4

Tabla 7. Product Backlog

3.2. Planificación del proyecto

Para el desarrollo del proyecto se planificaron un total de 6 Sprints, todos con una duración de 2 semanas. En caso de que se agreguen más historias de usuario en el transcurso de los Sprint, serán añadidos en uno de los próximos sprints, de acuerdo con la complejidad y prioridad del nuevo requerimiento o bug.

Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6
HI01	HI10	HI25	HI37	HI21	HI31
HI02	HI11	HI26	HI38	HI22	HI34
HI05	HI12	HI27	HI39	HI23	HI03
HI06	HI13	HI28	HI17	HI24	HI04
HI07	HI14	HI29	HI18	HI35	HI19
HI08	HI15	HI32	HI30	HI36	HI20
HI09	HI16	HI33			

Tabla 8. Planificación de Sprints

3.3. Sprint 0

3.3.1. Objetivo del Sprint

- Configurar el ambiente de desarrollo
- Creación de repositorios para versionado y almacenamiento del proyecto
- Creación de contenedor MySQL en Docker para almacenar la información
- Preparar el ambiente para implementar las prácticas DevOps

3.3.2. Ejecución del Sprint

- **Arquitectura del proyecto**

El proyecto usa una arquitectura Cliente-Servidor [30] en el cual se realiza la comunicación mediante servicios REST y para mantener la comunicación segura se implementó el uso del estándar JWT [31]. A continuación, en la ILUSTRACIÓN 1, se muestra la arquitectura a alto nivel

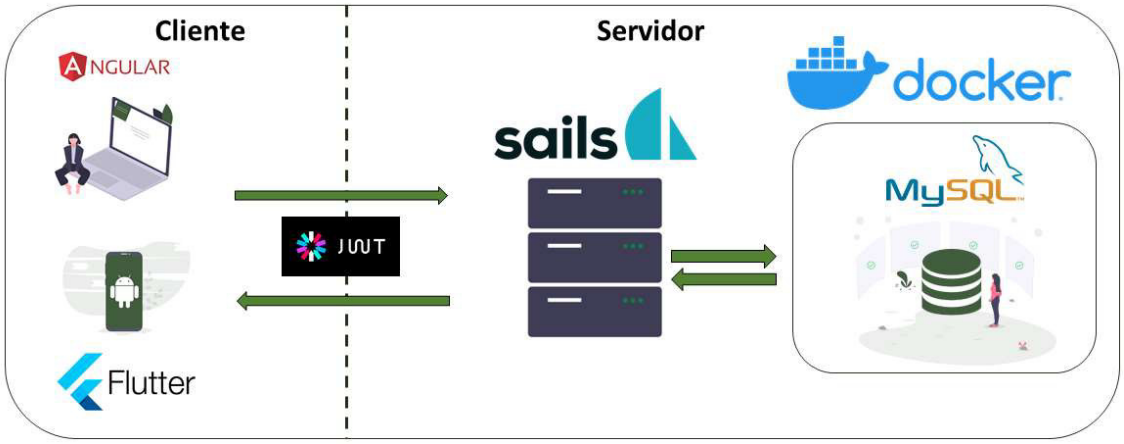

Ilustración 1. Arquitectura de alto nivel

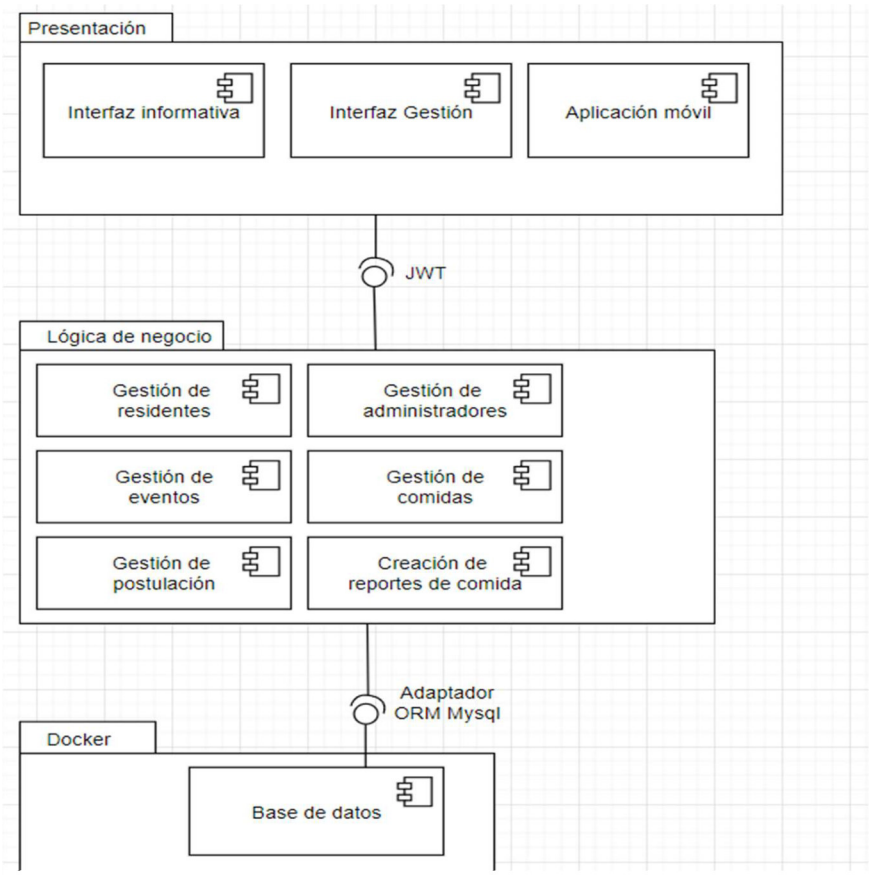

Ilustración 2. Diagrama de componentes

Las aplicaciones Web, tanto el frontend como el backend, fueron desarrolladas haciendo uso del patrón de arquitectura MVC (Modelo Vista Controlador) [30], mientras que la aplicación móvil se desarrolló haciendo uso del patrón BLoC (Bussiness Logic Component) [32]. A continuación, se muestra un diagrama de alto nivel de la arquitectura de la aplicación móvil.

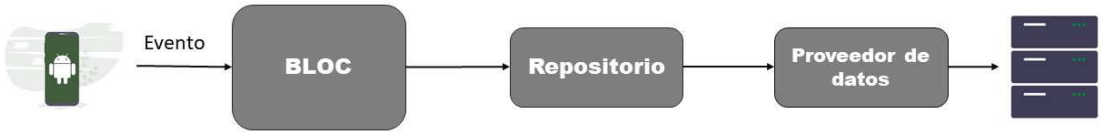

Ilustración 3. Diagrama de arquitectura de aplicación móvil

- **Control de versionamiento y repositorios en GitLab**

Para el control de versiones del proyecto se creó cuatro repositorios para cada una de las plataformas del proyecto. Tal como se lo indica en la TABLA 9. Esto permite que cada plataforma sea independiente de las demás, llegando a tener una mayor flexibilidad y mantenibilidad al momento de versionar el código. De igual manera, GitLab permite tener solo un archivo de configuraciones DevOps por repositorio, razón por la cual se enfatizó en la separación de los proyectos por repositorio.

Nombre de Repositorio	Plataforma
Ilinizas-admin	Aplicación web de gestión del sistema
Ilinizas-inf	Aplicación web informativa de la residencia
Ilinizas-app	Aplicación móvil de la residencia
Ilinizas-backend	Backend del sistema

Tabla 9. Repositorios del proyecto

La estregia que se escogió para el manejo de ramas dentro de cada repositorio fue **GitFlow** [33] debido a que se necesitaron ramas específicas para el despliegue en las cuales se aplicaron configuraciones específicas, por lo que, esta estrategia fue la que más se ajustó a las necesidades del equipo de desarrollo.

- **Herramientas de trabajo**

Se hizo uso de un contenedor MySQL en Docker como base de datos. Mientras que, para el desarrollo de las aplicaciones web se utilizó el framework Angular y para el desarrollo del backend se empleó el framework Sails.js; para ambos casos el editor de código seleccionado fue Visual Studio Code. Como host de estas aplicaciones web se utilizó Firebase y para el despliegue del backend se aprovechó las facilidades brindadas por Heroku. Finalmente, para

el desarrollo de la aplicación móvil se escogió el framework Flutter y se utilizó Android Studio como entorno de desarrollo.

- **Creación contenedor MySQL en Docker**

Para crear el contenedor MySQL se utilizó Docker Compose en Windows, utilizar Compose permitió definir una política de reinicio, el puerto a exponer y el volumen en el cual la información debe persistir.

```yaml
version: '3.3'
services:
  db:
    image: mysql:5.7
    restart: 'always'
    environment:
      MYSQL_DATABASE: 'db'
      MYSQL_USER: 'user'
      MYSQL_PASSWORD: '******'
      MYSQL_ROOT_PASSWORD: '******'
    ports:
      - '3306:3306'
    expose:
      - '3306'
    volumes:
      - D:\dockerData\mysql:/var/lib/mysql
```

Ilustración 4. Configuración de contenedor Docker con imagen MySQL

- **Configuración CI/CD**

Mediante GitLab se puede hacer uso de las configuraciones de integración, entrega, despliegue y monitoreos continuos. Sin embargo, en el proyecto se ha decidió implementar solo las configuraciones para la entrega y despliegue continuo, debido a que el proyecto no comprendió el desarrollo de pruebas unitarias en gran parte. Para configurar estas tareas en el entorno de GitLab, se necesitó crear un archivo YAML en cada repositorio a automatizar, en el caso de este proyecto los proyectos que se automatizó fueron

1. Aplicación web de gestión
2. Aplicación web informativa
3. Aplicación móvil

- **Modelado Base de Datos**

A continuación, se muestra el diagrama de base de datos realizado para la ejecución del proyecto. Adicionalmente, el diagrama se encuentra en el ANEXO 3.

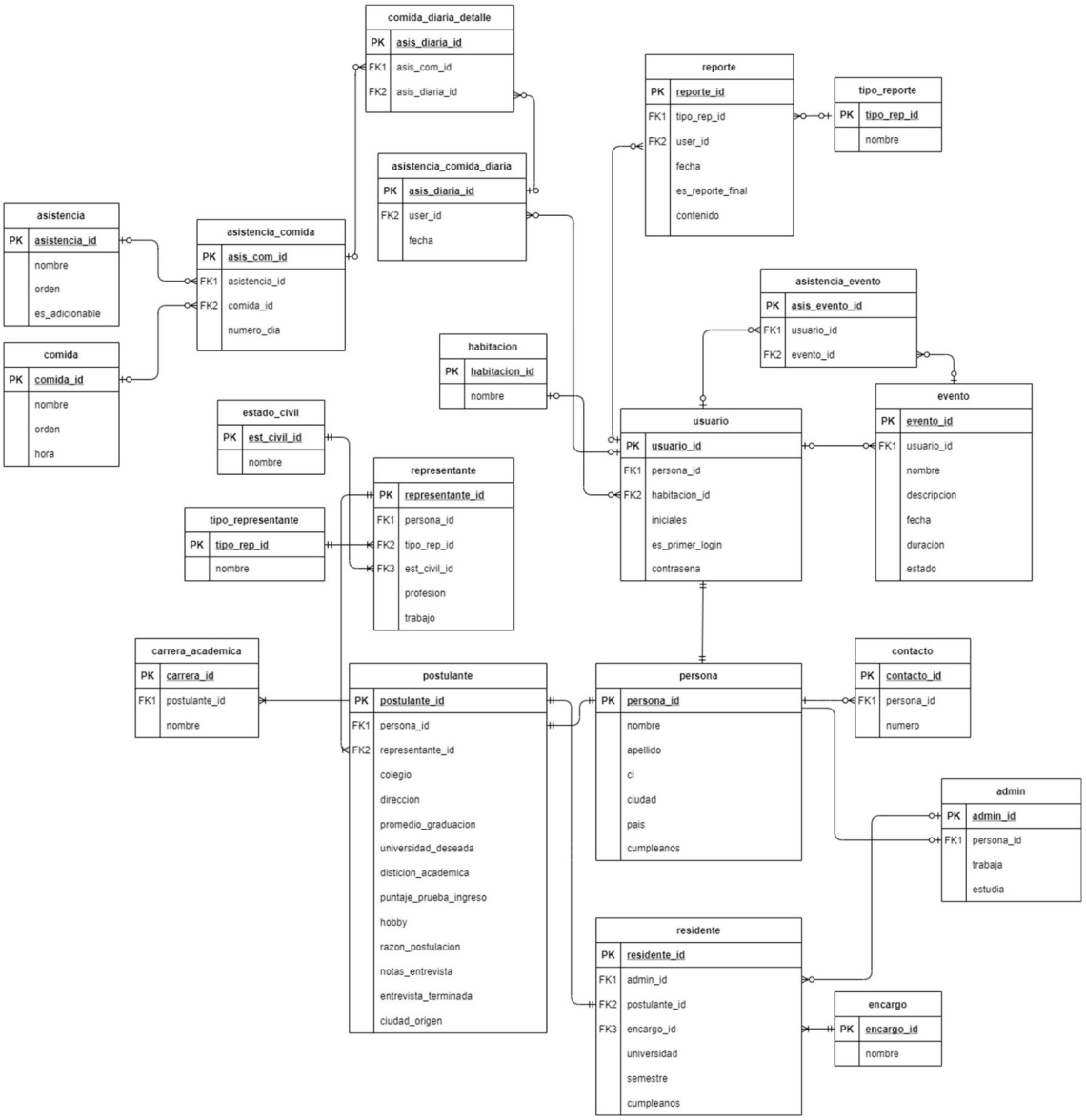

Ilustración 5. Diagrama de base de datos

3.3.3. Revisión del Sprint

Al terminar este Sprint se logró tener el ambiente de trabajo listo para empezar con el desarrollo. Las configuraciones que permiten el despliegue continuo van a permitir tener una mayor fluidez en el transcurso de los siguientes Sprints, sin embargo, se definió que para

cada repositorio se va a partir con una rama master en la cual se tendrá el código que está puesto en producción y una rama desarrollo en la cual se va a continuar el desarrollo.

3.4. Sprint 1

En este Sprint se realizaron las historias que permitieron tener un medio que le permita a la residencia darse a conocer. De igual manera, ya se pueden recibir postulaciones en el sistema. Logrando así dar inicio al proceso de inscripción.

3.4.1. Objetivo del Sprint

Desarrollar una página web en la cual se pueda registrar una inscripción a la residencia Ilinizas.

3.4.2. Sprint Backlog

Código	Nombre
HI01	Crear sección general
HI02	Crear formulario de inscripción
HI03	Crear sección servicios
HI05	Registrar información obligatoria del postulante
HI06	Registrar información académica del postulante
HI07	Registrar información de intereses del postulante
H08	Registrar información de los titulares del postulante
HI09	Notificar por correo electrónico el registro de la postulación

Tabla 10. Sprint Backlog del Sprint 1

3.4.3. Ejecución del Sprint

Se diseñó la interfaz de la aplicación informativa, de manera que el usuario tenga que realizar la cantidad mínima de pasos para completar su inscripción. Para esto, se implementó un formulario divido en pasos, tal como se lo indica en el ANEXO 4, para solicitar la información del postulante. Este formulario está compuesto por los pasos:

1. Información requerida: se solicita la cédula, nombre, email y fecha de nacimiento.

Ilustración 6. Información obligatoria en el formulario de inscripción

2. Información adicional: se solicita información de la ciudad de nacimiento, colegio de graduación, intereses y hobbies.

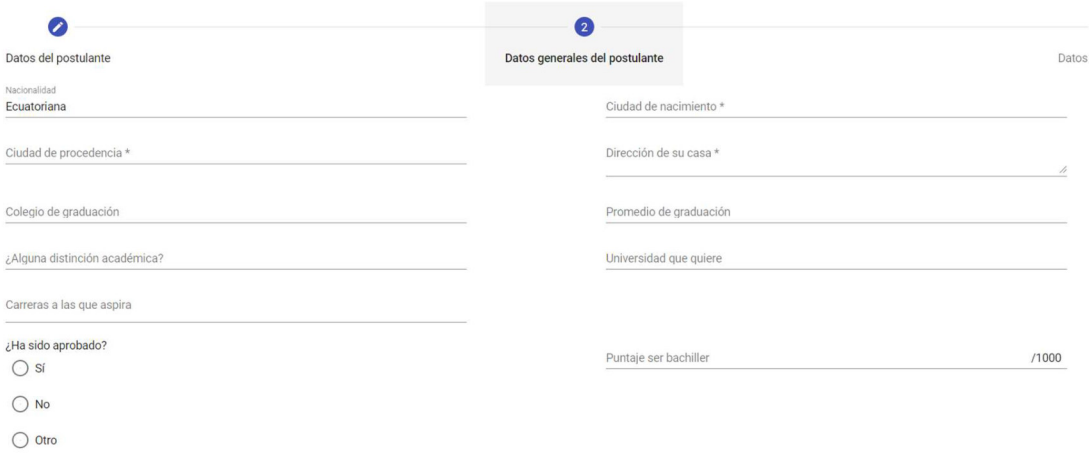

Ilustración 7. Información adicional en el formulario de inscripción

3. Información de representantes: se solicita información de los representantes, los tipos de representantes son configurables y se pueden añadir o quitar tipos de representantes, por lo que se crean controles dinámicos de acuerdo con la selección.

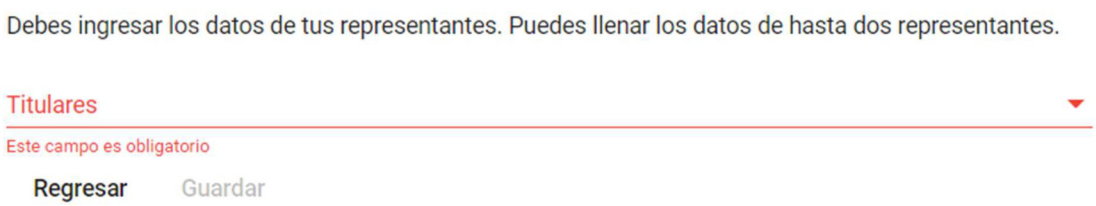

Ilustración 8. Información de representantes vacío en el formulario de inscripción

Para registrar la información de los representantes es obligatorio ingresar la información de cada uno.

Ilustración 9. FI: información de varios representantes

Una vez registrada la postulación, se procede a notificar al postulante el éxito del registro y se notifica al director de la aplicación que existe un nuevo registro de inscripción.

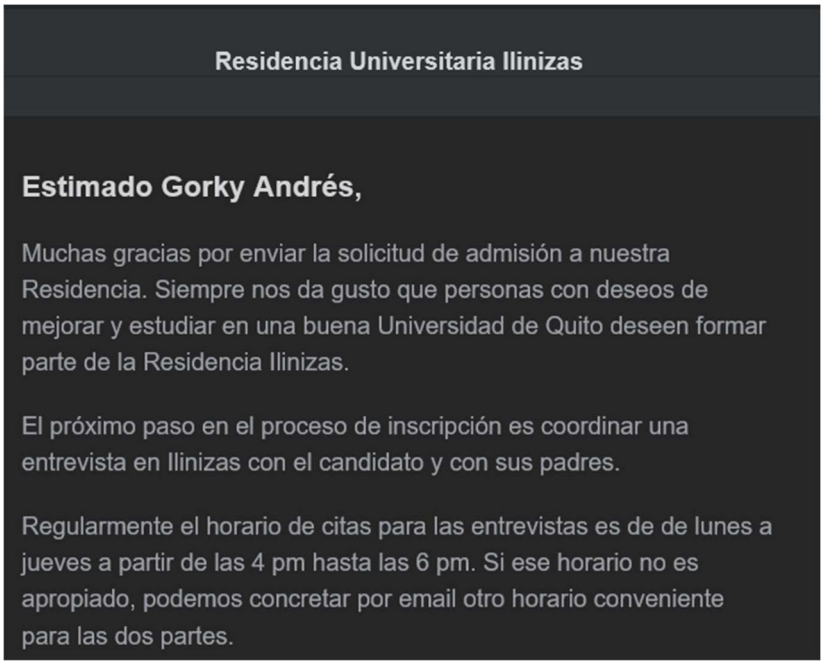

Ilustración 10. Email de inscripción satisfactoria enviado al postulante

Ilustración 11. Email de inscripción satisfactoria enviado al director

3.4.4. Daily Scrum

Se realizaron 6 reuniones, en las cuales el objetivo principal fue mostrar al cliente los diseños realizados para la aplicación web. Esto permitió realizar cambios de manera rápida y, por ende, tener la aprobación inmediata de los mismos para iniciar con la maquetación y desarrollo. Además, sirvieron para eliminar bloqueantes referentes a la información requerida de un postulante al realizar una inscripción.

3.4.5. Revisión del Sprint

En la revisión del Sprint se presentó el formulario de inscripción terminado, pero de forma local, con lo cual, se verificó que se cumplió el objetivo del Sprint. Sin embargo, el dueño del producto realizó sugerencias referentes al formulario, por lo cual, se realizó cambios en el Product Backlog para que se adapte a estos nuevos requerimientos.

3.4.6. Adaptación del Product Backlog

Al realizar pruebas del formulario de inscripción, se evidenció que cierta información debe estar parametrizada y no se debe permitir que el usuario sea quien la ingrese.

Épica	Código	Nombre	Complejidad	Prioridad
TI02	NI01	Parametrizar el tipo de representante	5	4
	NI02	Listar universidades de Quito	2	2
	NI03	Registrar varios números telefónicos del representante	3	2

Tabla 11. Nuevos requerimientos Sprint 1

3.4.7. Retrospectiva del Sprint

Se decidió manejar las retrospectivas con la técnica del velero, por lo que, el resultado de la reunión fue:

- Fortalezas:
  - Rápida comunicación entre el dueño del producto y el equipo de desarrollo
  - No se tiene dependencia de otros procesos
- Objetivos
  - Se cumplió el objetivo planteado para el sprint
- Debilidades
  - El ambiente que se tiene para probar es solo local
- Amenazas
  - Tiempo disponible para desarrollo

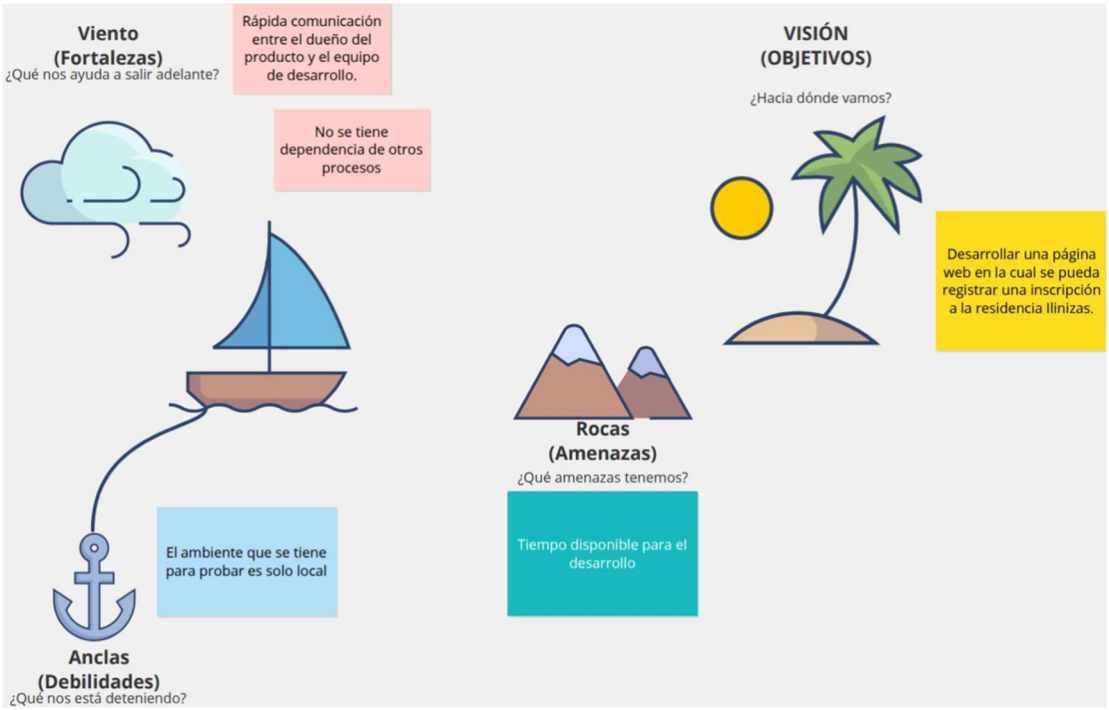

Ilustración 12. Retrospectiva Sprint 1

3.5. Sprint 2

En este Sprint se realizaron las historias añadidas en el Sprint anterior y las requeridas para continuar con el proceso de inscripción. Además, para que se pueda probar el formulario y se visualice la información, se realizó el despliegue de la aplicación backend y las aplicaciones web informativa y de gestión. Por lo que en el desarrollo de este sprint se detallará el proceso de entrega y despliegue continuo para las aplicaciones web.

3.5.1. Objetivo del Sprint

Visualizar la información de un postulante para analizar su posible aceptación en la residencia.

3.5.2. Sprint Backlog

Código	Nombre
NI01	Parametrizar el tipo de representante
NI02	Listar universidades de Quito
NI03	Registrar varios números telefónicos del representante
HI10	Iniciar sesión en el sistema
HI11	Consultar postulante

HI12	Crear residente
HI13	Consultar residente
HI14	Modificar residente
HI15	Eliminar residente
HI16	Crear administrador

Tabla 12. Sprint Backlog del Sprint 2

3.5.3. Ejecución del Sprint

Se realizó un catálogo de tipos de representantes, para así parametrizar los representantes que puede registrar el postulante.

Ilustración 13. Catálogo de tipos de representante

Y de igual manera, se creó una tabla para los números telefónicos. De esta forma, se puede registrar varios números telefónicos no solo para los representantes, si no, para todos los usuarios del sistema.

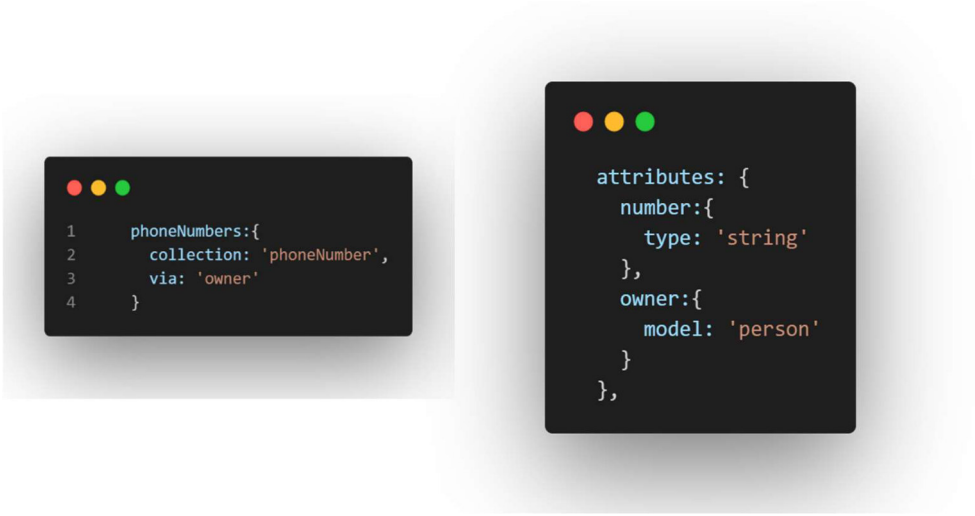

Ilustración 14. Relación para múltiples números de contacto

- **Iniciar Sesión en el sistema**

En este Sprint se comenzó con el desarrollo del portal de administración, por lo que, se realizó el diseño de la interfaz del sistema de gestión como se lo muestra en el ANEXO 5.

El servicio de Iniciar sesión fue diseñado para ser usado por el portal de administración y la aplicación móvil. Sin embargo, tiene un comportamiento diferente dependiendo de la plataforma. En el portal de administración solo permite el ingreso de usuarios que sean de tipo administrador y que estén activos en el sistema, tal como se lo indica en la siguiente ilustración.

```
    if (!person.user[0].state) {
      return exits.noAccess({
        err:
'This user has no permission to access this site.',
        type: 'user-disabled',
      });
    }
    if (channel === 'WEB' && !person.admin[0]) {
      return exits.noAccess({
        err:
'This user has no permission to access this site.',
        type: 'no-access',
      });
    }
```

Ilustración 15. Validaciones de login para portal de gestión

Las credenciales requeridas para realizar el inicio de sesión son el email y la contraseña, como se lo muestra en la siguiente ilustración

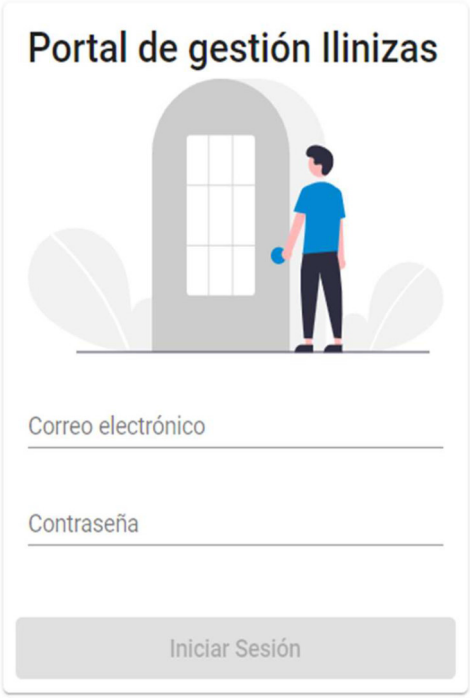

Ilustración 16. Pantalla de inicio de sesión en el portal de gestión

Para mantener la sesión iniciada en los dispositivos, el servicio devuelve un JWT el cual contiene la información del usuario que inició sesión y la duración del token. A continuación, se procede a guardar este token en el navegador y se mantiene la sesión iniciada hasta que el token caduque o el usuario cierre sesión. Dicho token tiene una duración de 2 días, sin embargo, esta duración es configurable y puede ser modificada sin problema en caso de ser requerido.

```
  friendlyName: 'Generate jwt',
  description: 'Generate a valid token',

  inputs: {
    payload: {
      type: 'ref',
      description: 'payload to the JWT.',
      required: true,
    },
    expirationTime: {
      type: 'string',
    },
    id: {
      type: 'number',
    },
  },

  fn: async function (inputs, exits) {
    const { expirationTime, id } = inputs;
    const idString = (id ? id : Date.now()).toString();
    const signedJWT = jwt.sign(inputs.payload, sails.config.custom.secret, {
      expiresIn: expirationTime
        ? expirationTime
        : sails.config.custom.expirationTime,
      jwtid: idString,
    });
    return exits.success(signedJWT);
  },
};
```

Ilustración 17. Generación de JWT

La llave secreta para firmar el JWT es almacenada como una variable de entorno para que así, no quede expuesta en el repositorio del proyecto y solo el equipo de desarrollo tenga acceso a la misma.

```
  custom: {
    secret: process.env.SECRET_JWT,
    expirationTime: '24 days',
  },
```

Ilustración 18. Configuración para firma y duración de JWT

Una vez realizado el inicio de sesión, se estableció que en cada petición que se realiza desde el sistema de gestión, se envía el token que está almacenado en el navegador, para esto, se implementó un interceptor que añade el token en caso de existir y si no, la petición es enviada sin el token, tal como se lo muestra en la siguiente ilustración.

```
intercept(
  request: HttpRequest<any>,
  next: HttpHandler
): Observable<HttpEvent<any>> {
  this.spinnerOverlayService.show();
  const token = this.authenticationService.userTokenValue;
  if (token) {
    const decoded = window.atob(token);
    request = request.clone({
      setHeaders: {
        authorization: `Bearer ${decoded}`,
      },
    });
  }
  return next
    .handle(request)
```

Ilustración 19. Interceptor para añadir JWT

Se realizó esto con el propósito de tener una comunicación segura. Para validar esta comunicación segura, se verifica el token en el backend y en caso de que sea inválido o esté caducado, el servidor no permite que la petición avance y retorna un error indicando el problema. El portal de gestión identifica el error y procede a cerrar la sesión del usuario.

```
if (req.header('authorization')) {
  var token = req.header('authorization').split('Bearer ')[1];
  if (!token) {
    return res.status(401).json({
      error: {
        name: 'noToken',
        message: 'Cant find the token in the authorization header',
      },
    });
  }
  return jwToken.verify(token, sails.config.custom.secret, (error, payload) => {
    if (error) {
      return res.status(401).json(error);
    }
    req.body = { ...req.body, payload };
    next();
  });
}
return res.status(401).json({
  error: {
    name: 'noAuthorizationHeader',
    message: 'The authorization header is not present in the request',
  },
});
```

Ilustración 20. Validación de JWT

El login se consideró como una de las funcionalidades más importante y utilizadas, se realizó pruebas unitarias para validar la correcta ejecución de dicho proceso. A continuación, se muestra una de las pruebas unitarias que conformaron el lote de pruebas para el inicio de sesión en el portal de gestión.

```
it('should return Disabled Message when the type is user-disabled', () => {
  expect(component.getErrorLoginMessage({ type: 'user-disabled' })).toBe(
    'El usuario ingresado se encuentra dado de baja.'
  );
});

it('should return Not Found Message when the type is not-found', () => {
  expect(component.getErrorLoginMessage({ type: 'not-found' })).toBe(
    'El usuario ingresado no existe'
  );
});

it('should return Incorrect credentials Message when the type is user-disabled', () => {
  expect(component.getErrorLoginMessage({ type: 'credentials' })).toBe(
    'La constraseña o email ingresados son incorrectos.'
  );
});

it('should return Disabled Message when the type is user-disabled', () => {
  expect(component.getErrorLoginMessage({ type: 'no access' })).toBe(
    'Lo sentimos, no tienes acceso para este sitio.'
  );
});
```

Ilustración 21. Pruebas unitarias inicio de sesión

- **CRUD RESIDENTES**

Dentro del portal de gestión se puede obtener información de los postulantes y de los residentes en la misma pantalla como se lo muestra en la siguiente ilustración

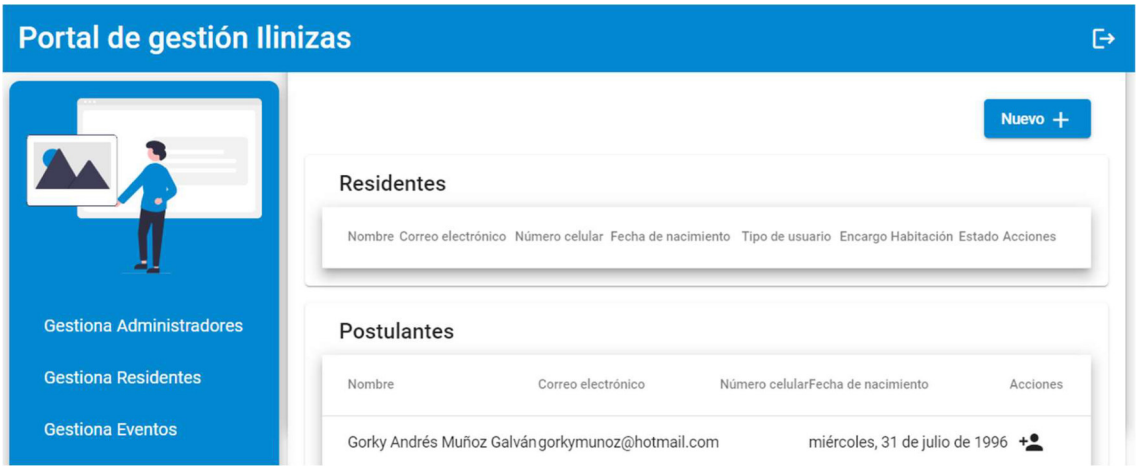

Ilustración 22. Pantalla de gestión de residentes

Se realizó esto debido a que, el modelo de la base de datos permite que un postulante ya forme parte del sistema, por lo que, para crear un residente, se debe indicar que el postulante ahora será un residente como se lo muestra en la ILUSTRACIÓN 22; y finalmente, se agrega información adicional requerida del nuevo residente.

```
createResident: async (req, res) => {
  const { resident, user } = req.body;
  try {
    await Resident.create(resident);
    const residentType = await UserType.findOne({ name: 'residente' });
    user.type = residentType.id;
    await User.create(user);
    return res.ok('Resident Created');
  } catch (er) {
    return res.serverError({ er });
  }
```

Ilustración 23. Creación de residente

Un residente tiene una habitación y una tarea asignada dentro de la residencia, por lo que, estos son los únicos datos que un administrador puede modificar del residente. En caso de que se quiera modificar la información personal del residente, esto se lo hace dentro de la aplicación móvil, requerimiento que será abordado en la historia de usuario HI33.

- **Despliegue de backend en Heroku**

Para que las aplicaciones web tengan acceso a los servicios desarrollados, es necesario primero desplegar a la web el backend de la aplicación. Primero se realizó los cambios necesarios en las configuraciones de la aplicación para el ambiente productivo, y se siguió la guía brindada por Heroku y se logró tener la aplicación web desplegada.

Ilustración 24. Aplicación backend desplegada

Finalmente se definió las variables de entorno configuradas para el proyecto. Esto permite que no se tenga que realizar nuevos despliegues en caso de que alguna de estas variables cambie, por lo que, solo se realizarán nuevos despliegues en caso de que se añadan nuevas funcionalidades.

- **Despliegue de aplicaciones web**

Para realizar el despliegue de las aplicaciones web se utilizó el servicio de hosting brindando por Firebase. Por lo que primero se realizó la creación del proyecto desde la consola de Firebase, al cual se lo denominó *tesis-ilinizas*

Ilustración 25. Proyecto en firebase

A continuación, se activó el servicio de *Hosting* y se instaló el servicio de Firebase CLI en el computador con los pasos detallados en la guía de Firebase. Una vez finalizados los pasos, se realizó un despliegue manual, con el cual se obtuvo el primer despliegue de la aplicación web

Ilustración 26. Despliegue manual

Una vez comprobado el correcto despliegue de la aplicación de forma manual, se procedió a configurar el despliegue continuo. Primero se definió en la raíz del repositorio un archivo llamado *.gitlab-ci.yaml* en el cual se definen todas las configuraciones necesarias para el entorno DevOps en el repositorio en GitLab. Primero se define el entorno necesario para la ejecución de las tareas, en este caso se requiere ***node*** y a continuación, se procede a definir las etapas a ejecutarse:

```
image: node:12.16.1

stages:
  - install
  - build
  - deploy
```

Ilustración 27. Etapas CI de aplicaciones web

En la primera etapa (*install)* se instalan todas las dependencias necesarias del proyecto y se las guarda por una hora en cache para que las siguientes etapas puedan utilizar estas dependencias. Finalmente, se ha definido que esta etapa se ejecute solo en la rama master, la cual se ha definido que será la única que tendrá la versión que se pondrá a producción.

```
install:
  stage: install
  variables:
    BUILD_CONFIG: "production"
  script:
    - npm install
  artifacts:
    expire_in: 1h
    paths:
      - node_modules/
  cache:
    paths:
      - node_modules/
  only:
    - master
```

Ilustración 28. Etapa install

La siguiente etapa (*build*) es la encargada de compilar y generar la aplicación que será desplegada a producción, de igual manera solo es ejecutada en la rama master.

```
build:
  stage: build
  variables:
    BUILD_CONFIG: "production"
  dependencies:
    - install
  script:
    - npm run build:ci
  artifacts:
    expire_in: 1h
    paths:
      - dist/
  only:
    - master
```

Ilustración 29. Etapa build

La última etapa es la encargada de desplegar la aplicación en Firebase. Para eso, instala el CLI de firebase y despliega la aplicación en el proyecto creado previamente en firebase.

```
deploy:
  stage: deploy
  dependencies:
    - build
  before_script:
    - npm i -g firebase-tools
  script:
    - firebase deploy -m "Pipeline $CI_PIPELINE_ID, build $CI_BUILD_ID"
    --non-interactive --token $FIREBASE_TOKEN
  environment:
    name: production
  only:
    - master
```

Ilustración 30. Etapa deploy

GitLab permite observar el estado de las etapas, en la cual se visualiza si se han completado o si se ha tenido algún error en una de las etapas, como se muestra en la ILUSTRACIÓN 30.

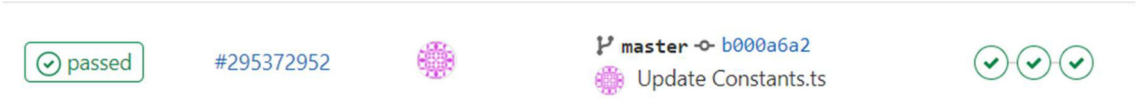

Ilustración 31. Visualizador de GitLab

Si no se presentaron errores al momento de ejecutar las etapas, la nueva versión de la aplicación se encuentra desplegada y al revisar la consola de firebase se obtiene el resultado de la ILUSTRACIÓN 31.

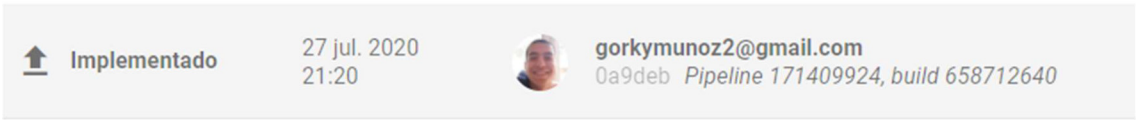

Ilustración 32. Despliegue continuo

3.5.4. Daily Scrum

En este Sprint se tuvo 8 reuniones durante la duración del sprint, en las cuales se validó los diseños del portal de administración y se discutió acerca de las diferentes opciones para el hosting de las aplicaciones.

3.5.5. Revisión del Sprint

Durante la revisión del sprint, se validó que el dueño del producto tuvo acceso al sitio web y se logró acceder al portal de gestión. Se utilizó el postulante registrado en el sprint anterior para darle continuación al proceso de inscripción con los mismos datos, con lo cual se validó la creación y consulta de un nuevo residente. Adicionalmente el dueño del producto indicó que se deben añadir información adicional para el residente, por lo cual se realizó un ajuste del producto backlog.

3.5.6. Adaptación del Product Backlog

El dueño del producto indicó que un residente tiene asignado un cargo en la residencia, por lo cual se decidió añadir un CRUD para los cargos que puede tener un residente.

Épica	Código	Nombre	Complejidad	Prioridad
TI02	NI04	Crear encargo	5	3
	NI05	Consultar encargo	2	3

	NI06	Modificar encargo	2	1
	NI07	Eliminar encargo	2	1

Tabla 13. Nuevos requerimientos Sprint 2

3.5.7. Retrospectiva del Sprint

Se continuó con la técnica del barco, con la cual se identificó los siguientes puntos:

- Fortalezas
  - Automatización del proceso de despliegue
- Objetivo
  - Se cumple el objetivo del sprint y se permite el acceso a cualquier usuario al formulario de inscripción
- Debilidades
  - El tamaño del equipo de trabajo
  - El tiempo que se le asigna al desarrollo del proyecto
- Amenazas
  - Constantes cambios que ponen en riesgo la duración de los sprints.

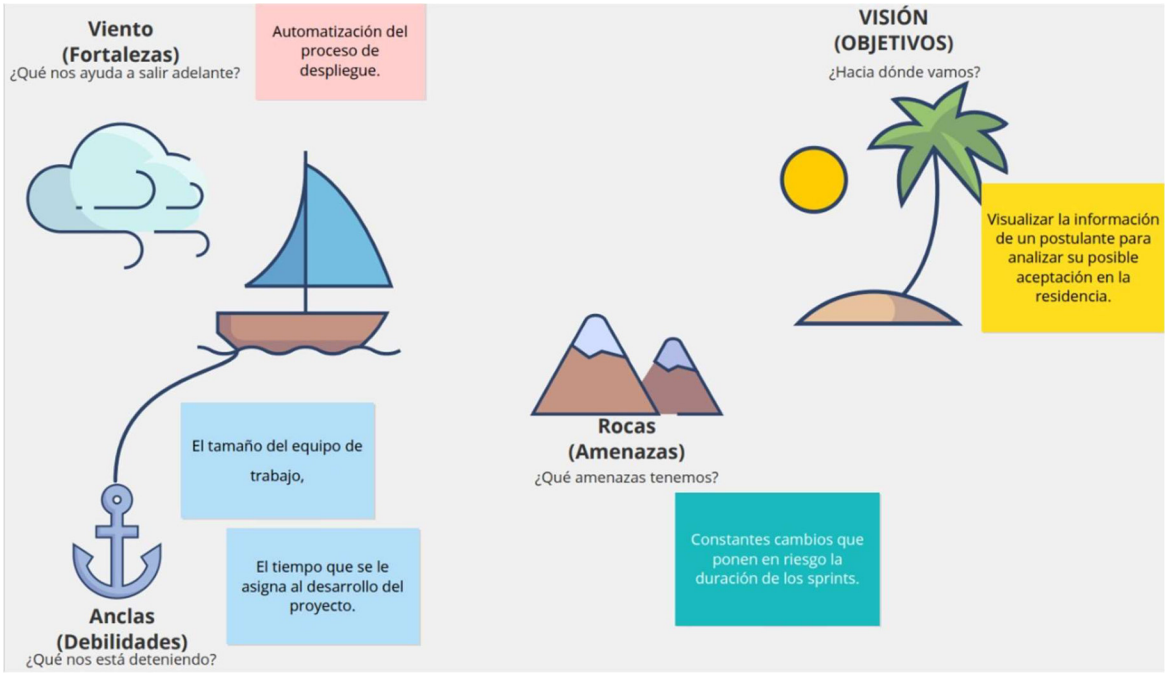

Ilustración 33. Retrospectiva Sprint 2

3.6. Sprint 3

3.6.1. Objetivo del Sprint

Iniciar sesión en la aplicación móvil como usuario registrado de Ilinizas.

3.6.2. Sprint Backlog

Código	Nombre
HI25	Crear pantalla de *Splash*
HI26	Crear pantalla de Inicio
HI27	Iniciar Sesión
HI28	Barra de navegación dinámica
HI29	Sección gestión de registro de comidas
HI32	Sección perfil
HI33	Cerrar sesión
NI04	Crear encargo
NI05	Consultar encargo
NI06	Modificar encargo
NI07	Eliminar encargo

Tabla 14. Sprint Backlog del Sprint 3

3.6.3. Ejecución del Sprint

En este Sprint se comenzó el desarrollo de la aplicación móvil. Por lo que se inició realizando los diseños y un mapa de navegación, los cuales se encuentran en el ANEXO 6.

Para la aplicación móvil se reutilizó el servicio de login creado para el portal administrativo. En caso de tener un inicio de sesión exitoso, el token de autenticación es almacenado en las preferencias del dispositivo y es utilizado para validar la pantalla inicial a mostrar. En caso de que exista un token almacenado en el dispositivo, se navega a la pantalla principal y si no existe un token almacenado se navega a la pantalla de bienvenida.

```
checkInitialRoute(BuildContext context) async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  String loggedUser = prefs.getString('user');
  if (loggedUser == null) {
    Navigator.pushReplacementNamed(context, WelcomeScreen.WELCOME_ROUTE);
  } else {
    Navigator.pushReplacementNamed(
        context, NavigationScreenProvider.NAVIGATION_ROUTE);
  }
}
```

Ilustración 34. Ruta inicial aplicación móvil

En el transcurso del desarrollo de esta aplicación no se encontró una librería certificada para la validación de un token JWT; razón por la cual, la validación la realiza el backend. Por lo que, se envía el token en cada petición que se realiza al servidor. La sesión permanecerá abierta hasta que el usuario decida cerrarla de forma manual o hasta que el servidor notifique en que el token expiró.

La capa de datos está conformada por peticiones HTTP la cual fue realizada con una base, en la cual se especifican los métodos GET, POST y PUT realizados desde la aplicación, tal como se indica en la ILUSTRACIÓN 34.

```
Future<dynamic> post({String url, var bodyContent}) async {
  var responseJson;
  try {
    final response = await (bodyContent != null
        ? http.post('$_baseUrl$url', body: bodyContent)
        : http.post('$_baseUrl$url'));
    responseJson = _returnResponse(response);
  } on SocketException {
    throw FetchDataException(kNoInternetConnectionError);
  }
  return responseJson;
}
```

Ilustración 35. Función genérica para una petición POST

Una vez recibida la respuesta del servicio, se utiliza el código de estado de la petición para decidir si mapear el resultado o decidir cómo se debe manejar la excepción, tal como se lo muestra en la siguiente ilustración.

```
dynamic _returnResponse(http.Response response) {
  String body = response.body.toString();
  print('${response.statusCode}::${response.body}');
  switch (response.statusCode) {
    case 200:
    case 202:
      return json.decode(body);
    case 400:
      throw BadRequestException(body);
    case 401:
      throw UnauthorisedException(body);
    case 403:
      throw DisabledException(kDisabledUserError);
    case 404:
      throw NotFoundException(kNotFoundUserError);
    case 409:
      throw CredentialsException(kCredentialsError);
    case 500:
    default:
      var errorBody = json.decode(body);
      String reasonError = errorBody['description'] ??
          'Error occurred while Communication with Server with StatusCode : ${response.statusCode}';
      throw FetchDataException(reasonError);
  }
}
```

Ilustración 36. Manejador de respuestas HTTP

Se definió varias excepciones personalizadas para tratar de una mejor manera el resultado y brindar al usuario una mejor explicación del resultado obtenido. Las excepciones que se maneja para las peticiones se las muestra en la ILUSTRACIÓN 36.

```
class AppException implements Exception {
  final _message;
  final _prefix;

  AppException([this._message, this._prefix]);
  String toString() {
    return "$_message";
  }
}

class FetchDataException extends AppException {
  FetchDataException([String message])
      : super(message, "Error During Communication: ");
}
class BadRequestException extends AppException {
  BadRequestException([message]) : super(message, "Invalid Request: ");
}
class UnauthorisedException extends AppException {
  UnauthorisedException([message]) : super(message, "Unauthorised: ");
}
class CredentialsException extends AppException {
  CredentialsException([message]) : super(message, "Credentials: ");
}
class DisabledException extends AppException {
  DisabledException([message]) : super(message, "Disabled: ");
}
class NotFoundException extends AppException {
  NotFoundException([message]) : super(message, "Not Found: ");
}
class InvalidInputException extends AppException {
  InvalidInputException([String message]) : super(message, "Invalid Input: ");
`
```

Ilustración 37. Excepciones personalizadas

Para manejar el estado de cada petición y lo que se va a mostrar al usuario se definieron tres estados:

- Cargando: se inició la petición
- Completado: se completó la petición de forma correcta
- Error: se completó la petición con un error

Una vez finalizada la petición, se procede a mapear la información obtenida en caso de que la petición sea exitosa, caso contrario se informa a través de un mensaje explicativo el error, tal como se muestra en la siguiente ilustración

```
class ApiResponse<T> {
  Status status;
  T data;
  String message;

  ApiResponse();

  ApiResponse.loading(this.message) : status = Status.LOADING;

  ApiResponse.completed(this.data) : status = Status.COMPLETED;

  ApiResponse.completedWithMessage(this.data, this.message) : status = Status.COMPLETED;

  ApiResponse.error(this.message) : status = Status.ERROR;

  @override
  String toString() {
    return "Status : $status \n Message : $message \n Data : $data";
  }
}
```

Ilustración 38. Manejo de estado de peticiones

Una vez iniciada la sesión, las opciones disponibles en la barra de navegación son definidas de acuerdo con el tipo de usuario. Para satisfacer esta necesidad, se utilizó el patrón de diseño *Factory,* como se muestra en la ILUSTRACIÓN 38.

```
abstract class MenuUser {
  factory MenuUser(UserTypeEnum type) {
    switch (type) {
      case UserTypeEnum.Administrador:
        return AdministratorMenuOption();
      case UserTypeEnum.Director:
        return DirectorMenuOption();
      case UserTypeEnum.Residente:
        return ResidentMenuOption();
      default:
        return null;
    }
  }

  List<BottomNavigationBarItem> menuItems();

  List<Widget> menuScreens();
}
```

Ilustración 39. Opciones de Usuario

Para cerrar sesión se elimina de las preferencias el token de autenticación y se borra la pila de navegación de la aplicación. Para así, enviar al usuario a la pantalla de Inicio y que no pueda regresar a las opciones que requieren de una sesión activa.

```
void logOut(BuildContext context) async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  await prefs.remove('user');
  await prefs.remove('token');
  Navigator.pushReplacementNamed(context, WelcomeScreen.WELCOME_ROUTE);
}
```

Ilustración 40. Cerrar Sesión

Para que la aplicación pueda ser revisada por el dueño del producto, se subió la aplicación a la *Play Store* de Google. Por lo cual, en este sprint se inició con el despliegue continuo en la aplicación móvil. Cabe recalcar que la publicación en tiendas solo se lo hizo para la aplicación Android debido a que no se cuenta con una cuenta de desarrollador Apple.

Para esto se utilizó la herramienta *fastlane* en Windows, por lo que se utilizó la guía brindada en la documentación para la instalación y configuración necesaria.

Una vez finalizada la configuración inicial, se creó dos tareas que permiten la creación y publicación de la aplicación en tiendas. La primera tarea es la encargada de generar un archivo *Android App Bundle* (aab) el cual va a ser el archivo por subir en la tienda de Google.

```
platform :android do
  desc "Build android aab"
  lane :build_android do
    Dir.chdir "../.." do
      sh("flutter", "build", "appbundle")
    end
  end
```

Ilustración 41. Creación de aab

A continuación, se procede a realizar a subir en la tienda de Google el archivo generado en el paso anterior, el cual es el objetivo de la siguiente tarea que permite publicar en cualquier tipo de escenario (producción, alfa, beta, interna).

```
desc "Deploy (Upload to play store)"
lane :deploy_android do |options|
  upload_to_play_store(
    track: options[:production] ? 'production' : options[:internal] ? 'internal' : options[:alpha] ? 'alpha' : 'beta',
    aab: '../build/app/outputs/bundle/release/app-release.aab',
    release_status: 'draft',
    mapping: '../build/app/outputs/mapping/release/mapping.txt',
    skip_upload_images: true,
    skip_upload_screenshots: true,
  )
end
```

Ilustración 42. Publicación en Play Store

Una vez definidas las tareas en *fastlane* se procedió a definir las tareas de despliegue continuo en GitLab. Se crearon un total de 3 etapas: *clean, build, deploy.* Además, se definieron variables necesarias para el correcto funcionamiento de fastlane y se realiza la instalación de las dependencias necesarias para ejecutar *fastlane.*

```
stages:
  - clean
  - build
  - deploy

variables:
  LC_ALL: "en_US.UTF-8"
  LANG: "en_US.UTF-8"

before_script:
  - gem install bundler
  - cd android
  - bundle install
  - cd ..
```

Ilustración 43. Configuración inicial CI Android

En la primera etapa se obtiene las dependencias de la aplicación y se limpia los archivos generados que no se va a utilizar. Se define que esta etapa puede ser interrumpida para que, en caso de que exista un nuevo *commit* mientras se esté ejecutando la tarea, se cancele y proceda a ejecutarla con los nuevos cambios subidos. Finalmente, se definió que esta tarea

solo debe ejecutarse en las ramas de *master* e *intern*, en las cuales se tiene el código de las aplicaciones que se subió a producción.

```
clean:
    stage: clean
    tags:
        - app-ilinizas
    script:
        - flutter packages get
        - flutter clean
    interruptible: true
    only: ['master', 'intern']
```

Ilustración 44. Etapa clean CI Android

La siguiente etapa que se definió es la encarga de compilar la aplicación y crear el archivo por subir en tiendas. La información de la llave utilizada para firmar la aplicación no se encuentra en el repositorio por motivos de seguridad, por lo que, en esta etapa se crea el archivo de manera temporal para que al compilar la aplicación se pueda obtener la información de la llave. Una vez generado el archivo por subir se indica que se lo debe mantener en el ambiente. De igual manera, esta etapa solo se aplica para las ramas *master* e *intern*.

```yaml
android:build:
    stage: build
    dependencies:
        - clean
    tags:
        - app-ilinizas
    script:
        - cd android
        - echo "$LOCAL_PROPERTIES_BASE_64" | base64 --decode > local.properties
        - echo "KEY_PROPERTIES_BASE_64" | base64 --decode > key.properties
        - echo "$PLAY_STORE_UPLOAD_KEY" | base64 --decode > key.jks
        - bundle exec fastlane build_android
        - rm -f key.properties
        - rm -f key.jks
        - cd ..
    artifacts:
        paths:
            - build/app/outputs/bundle/release/app-release.aab
            - build/app/outputs/mapping/release/mapping.txt
        expire_in: 2 days
    interruptible: true
    only: ['master', 'intern']
```

Ilustración 45. Etapa de construcción de aab desplegable

Finalmente, se tiene la etapa encargada de subir a la Play Store los archivos generados en la etapa anterior, sin embargo, como se tiene varios escenarios, se creó dos tareas para esta etapa, una se encarga de publicar de manera interna (solo para testers) y producción.

En la ILUSTRACIÓN 45 se muestra la etapa para desplegar la aplicación de manera interna. Esta etapa solo se ejecuta en la rama *intern*.

```yaml
internal:android:deploy:
    stage: deploy
    dependencies:
        - android:build
    tags:
        - app-ilinizas
    script:
        - cd android
        - bundle exec fastlane deploy_android internal:true
    only:
        - intern
```

Ilustración 46. Etapa de despliegue en pruebas internas Android

Finalmente, la ILUSTRACIÓN 46 muestra la configuración para el despliegue a producción, se definió que esta etapa solo debe ejecutarse en la rama *master* y solo va a ser ejecutada de forma manual, para que nunca se haga una actualización sin una correcta revisión.

```
prod:android:deploy:
    stage: deploy
    dependencies:
        - android:build
    tags:
        - app-ilinizas
    script:
        - cd android
        - bundle exec fastlane deploy_android production:true
    when: manual
    only:
        - master
```

Ilustración 47. Etapa de despliegue a producción Android

Una vez configuradas las etapas de despliegue para la aplicación móvil, se procedió a realizar la primera publicación de la aplicación móvil en tiendas, sin embargo, estas publicaciones fueron en el área de **pruebas internas,** para que solo la lista de testers y el dueño del producto, puedan tener acceso a la misma. Para cumplir este proceso, bastó con realizar un commit en la rama **intern** y el proceso DevOps configurado para la aplicación móvil empezó de manera automática y cuando finalizó se obtuvo el siguiente resultado

Ilustración 48. Despliegue de aplicación con DevOps

Finalmente, en la **Play Console** se indica que una versión de la aplicación ha sido subida como versión de prueba interna, como lo indica la siguiente ilustración.

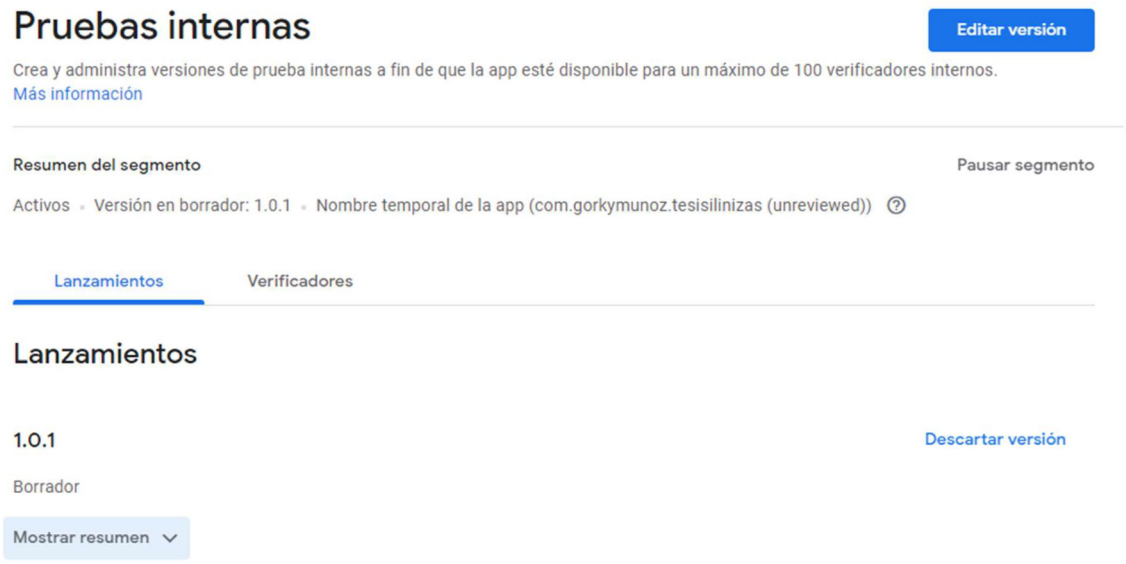

Ilustración 49. Aplicación móvil publicada en pruebas internas

3.6.4. Daily Scrum

En este sprint se tuvo 8 reuniones con el dueño del producto, en las cuales se discutió un cambio muy fuerte al sprint en curso, ya que, se solicitó eliminar un tipo de usuario. Sin embargo, al haber implementado la barra de navegación utilizando el patrón de diseño *Factory* el cambio no implicó un alto riesgo al cumplimiento del sprint.

3.6.5. Revisión del Sprint

En la revisión del sprint se verificó que un usuario registrado ya puede acceder a la aplicación móvil, sin embargo, no se tenía completado el diseño de la interfaz gráfica debido a que se solicitó un cambio en el transcurso del sprint. Esto será completado en el siguiente sprint ya que no se ha solicitado la creación de nuevas tareas. De igual manera, se validó que la aplicación está disponible en la tienda de Google para el dueño del producto. Esto debido a que la aplicación fue desplegada al estado de pruebas internas.

3.6.6. Retrospectiva del Sprint

Para este sprint se identificaron los siguientes puntos:

- Fortalezas:
    - El equipo de desarrollo brindó la facilidad de que el dueño del producto tenga en su dispositivo la aplicación instalada.
- Objetivo:

- El equipo de desarrollo ha adquirido los conocimientos básicos para realizar el despliegue continuo de aplicaciones web y Android.
- El equipo de desarrollo cumplió con la entrega de las funcionalidades base

- Debilidades:
  - El equipo de desarrollo no cuenta con una Mac para compilar la aplicación para dispositivos iOS
- Amenazas:
  - El equipo no posee una cuenta de desarrollador Apple

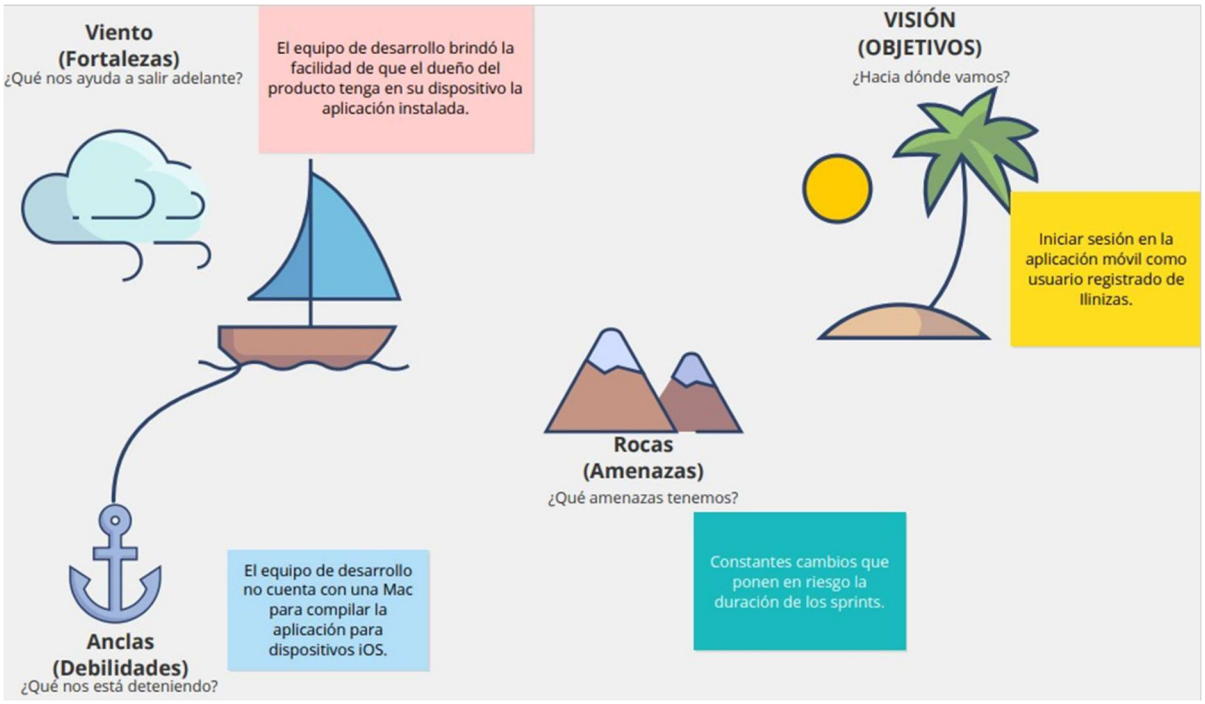

Ilustración 50. Retrospectiva Sprint 3

3.7. Sprint 4

3.7.1. Objetivo del Sprint

Registrar la asistencia a las comidas de la residencia mediante la aplicación y obtener un reporte diario de la cantidad de asistentes.

3.7.2. Sprint Backlog

Código	Nombre
HI37	Registrar comida diaria
HI38	Editar comida diaria

HI39	Reportería de comidas diarias
HI17	Consultar administrador
HI18	Modificar administrador
HI30	Sección gestión de eventos

Tabla 15. Sprint Backlog del Sprint 4

3.7.3. Ejecución del Sprint

El desarrollo de este sprint fue el más complejo, debido a que la asistencia a comidas diarias es el proceso más complejo de la residencia.

Primeramente, se completó el diseño de la pantalla de inicio de sesión en la aplicación como se lo muestra en la siguiente ilustración, la cual quedó como deuda del sprint anterior.

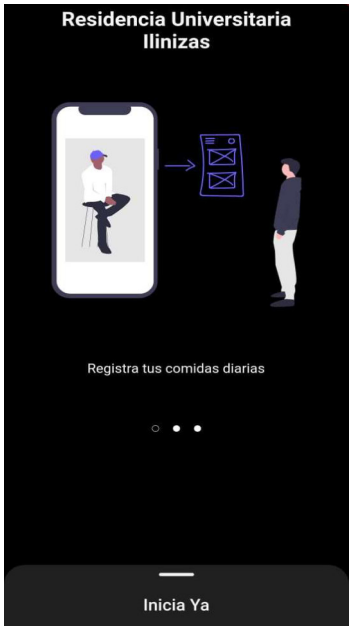

Ilustración 52. Pantalla de bienvenida

Ilustración 51. Pantalla de inicio de sesión

Se realizó una configuración de las diferentes comidas que brinda la residencia y de los tipos de asistencia que existen para cada comida. De esta manera, se permite que la configuración sea dinámica, ya que en época de vacaciones se quitan algunos tipos de asistencia debido al bajo aforo. Finalmente, se definió a la configuración de la asistencia a las comidas como diaria e independiente, tal como se lo muestra en la siguiente ilustración.

Ilustración 53. Pantalla de registro de asistencia a comidas

Es obligatorio escoger la asistencia a todas las comidas existentes, para así poder guardar el registro, tal como lo muestra en la siguiente ilustración.

Ilustración 54. Selección obligatoria de asistencia a comidas

Se permitió que el usuario pueda modificar un registro de asistencia a las comidas, sin embargo, se definieron dos consideraciones para la modificación.

1. En caso de que la fecha del registro por modificar sea la fecha actual, se definió una hora límite configurable para realizar la modificación. Sin embargo,

Para permitir al usuario registrar su asistencia en varios días, se ha definido como límite 8 días, es decir, puede registrar su asistencia desde el día actual hasta una semana después. En la ILUSTRACIÓN 52 el día actual es el 15 de junio, por lo que, podrá registrar su asistencia hasta el 22 de junio, tal como se lo muestra en la siguiente ilustración.

Ilustración 55. Fecha máxima de registro de asistencia

Una vez realizado el registro de la asistencia, se obtiene un resumen con la información registrada, como se lo muestra en la siguiente ilustración.

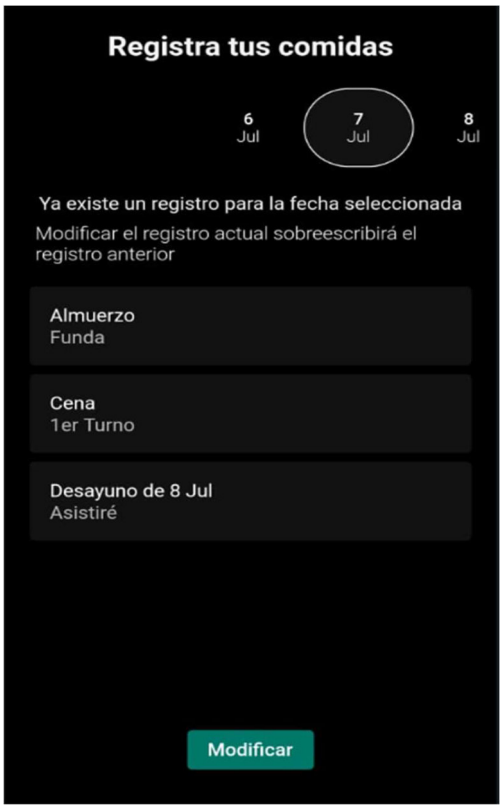

Ilustración 56. Resumen de registro de asistencia

Para modificar un registro de asistencia, se debe modificar el registro entero. Al igual que un nuevo registro, sin embargo, existe una condición, en caso de que haya pasado la hora definida por el administrador, el usuario no podrá modificar su registro de almuerzo en caso de que haya escogido la opción **funda,** sin embargo, el resto de las opciones si podrá modificar; tal como se lo muestra en la siguiente ilustración.

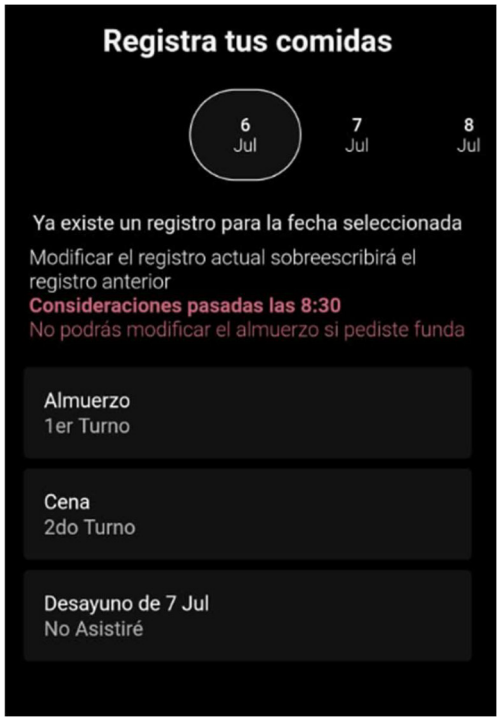

Ilustración 57. Resumen de asistencia pasada la hora configurable

Para obtener el reporte de asistencias se debe escoger la fecha del reporte que se desea generar, el sistema permite consultar reportes pasados y el actual, pero restringe la consulta de reportes posteriores a 7 días.

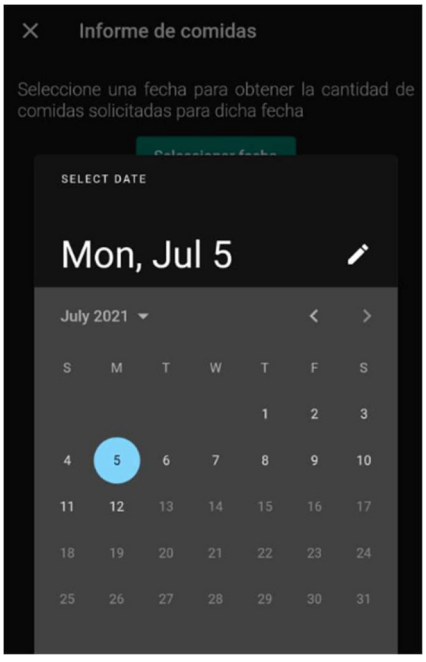

Ilustración 58. Selección de fecha de reporte de asistencia.

El reporte indica la cantidad de personas que asistirán al tipo de comida y los separa por tipo de asistencia para así, tener más claro quiénes son las personas que pueden ingresar al comedor, a continuación, se muestra un ejemplo de un reporte generado.

Ilustración 59. Reporte de asistencia a comida

Finalmente, el código fue desplegado a la rama *intern* para que automáticamente se despliegue una nueva versión de la aplicación a la tienda de Google y así el dueño del producto tenga la aplicación actualizada para la revisión.

3.7.4. Daily Scrum

En este sprint se realizaron 10 reuniones diarias debido a la complejidad de la asistencia a las comidas diarias. Se realizaron varias validaciones con el dueño del producto en estas reuniones y se logró tener una mayor claridad de las reglas y configuraciones necesarias.

3.7.5 Revisión del Sprint

Se realizó una revisión general de la funcionalidad de asistencia a comidas diarias, en las cuales se evidenció su correcto funcionamiento, sin embargo, los tipos de comida y asistencia se mostraban de forma aleatoria, por lo que se solicitó que se ordenen los registros, lo cual será tratado como nuevo requerimiento en los siguientes sprints.

3.7.6. Adaptación del Product Backlog

El dueño del producto indicó que se necesita registrar notas de los postulantes en las entrevistas, por lo que será tratado en el siguiente sprint con las tareas correspondientes.

Épica	Código	Nombre	Complejidad	Prioridad
TI04	NI08	Crear notas de entrevista	8	4
	NI09	Consultar notas de entrevista	3	4
	NI10	Ordenar los tipos de comida	3	2
	NI11	Ordenar los tipos de asistencia	3	2

Tabla 16. Nuevos requerimientos sprint 4

3.7.7. Retrospectiva del Sprint

Para este sprint se identificaron los siguientes puntos:

- Fortalezas:
  - La rápida comunicación entre el dueño del producto y el equipo de desarrollo.
  - Se tiene mayor conocimiento de los beneficios del negocio.
- Objetivo:
  - Se cumplió con un producto mínimo viable para el registro de la asistencia a las diferentes comidas. Permitiendo que se reemplace el previo modelo manual.

- Debilidades:
  - Tiempo limitado del equipo de desarrollo para reuniones con el dueño del producto.
- Amenazas:
  - Debido a complejidad de la funcionalidad se requiere una previa explicación al usuario.

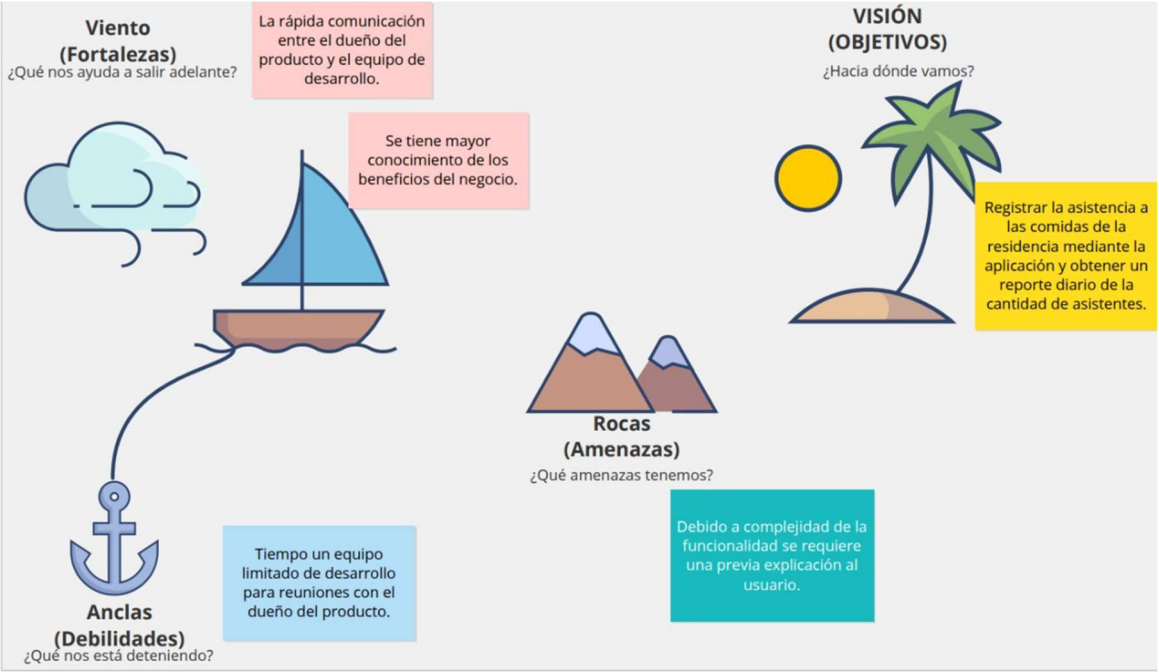

Ilustración 60. Retrospectiva Spring 4

3.8. Sprint 5

3.8.1. Objetivo del Sprint

Registrar la asistencia a eventos de la residencia mediante la aplicación.

3.8.2. Sprint Backlog

Código	Nombre
HI21	Crear evento
HI22	Consultar evento
HI23	Modificar evento
HI24	Eliminar evento
HI35	Registro de usuario en un evento
HI36	Eliminar registro de usuario en un evento
NI08	Crear notas de entrevista
NI09	Consultar notas de entrevista
NI10	Ordenar los tipos de comida
NI11	Ordenar los tipos de asistencia

Tabla 13. Sprint Backlog del Sprint 5

3.8.3. Ejecución del Sprint

- **CRUD Eventos**

En el portal de gestión se realiza todo el CRUD de eventos en una sola pantalla, como se lo muestra en la siguiente ilustración.

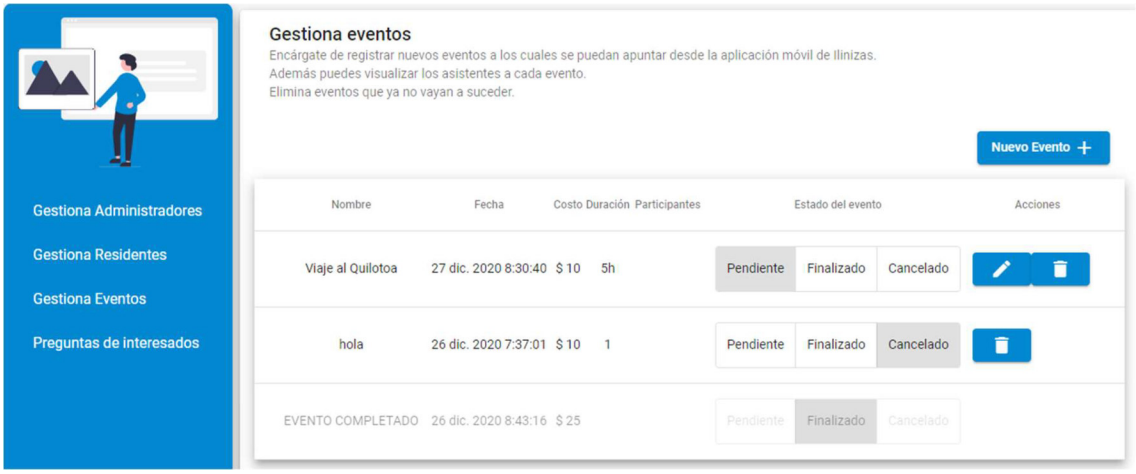

Ilustración 61. Pantalla de gestión de eventos

Para crear un evento, se debe seleccionar en el botón **Nuevo Evento,** el cual despliega una pequeña ventana en la cual se debe ingresar la información requerida para un evento.

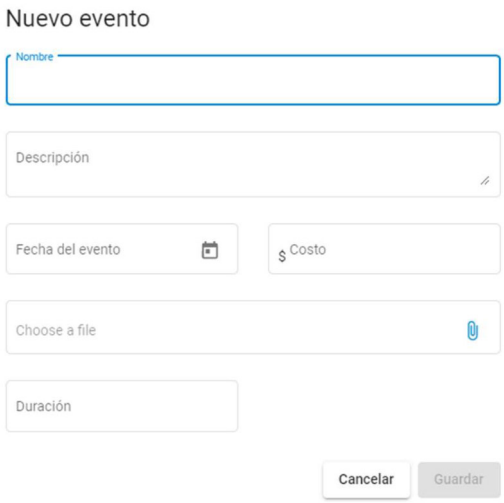

Ilustración 62. Nuevo evento

Se definió tres estados para un evento, los cuales permiten definir que eventos deben ser mostrados en la aplicación móvil para registrarse. Un evento va a ser creado con el estado de **PENDIENTE**, tal como se lo muestra en la siguiente ilustración del modelo.

```
status: {
  type: 'string',
  isIn: ['PENDING', 'CANCELLED', 'COMPLETED'],
  defaultsTo: 'PENDING',
  required: true
},
```

Ilustración 63. Estados de un evento

Una vez creado el evento, pasa a formar parte de la lista de eventos creados. En esta lista se muestra la información ingresada en la creación del evento y se permite modificar el estado del evento.

Un evento con estado **PENDIENTE** será listado en la aplicación móvil y los usuarios de la aplicación podrán registrarse en él. Los eventos con estado **FINALIZADO** y **CANCELADO** solo serán mostrados en el portal de gestión, con una importante diferencia, un evento **FINALIZADO** puede volver a estar como **PENDIENTE** o puede ser eliminado; mientras que un evento **CANCELADO** no podrá ser modificado ni eliminado y permanecerá en el registro, tal como se lo muestra en la siguiente ilustración.

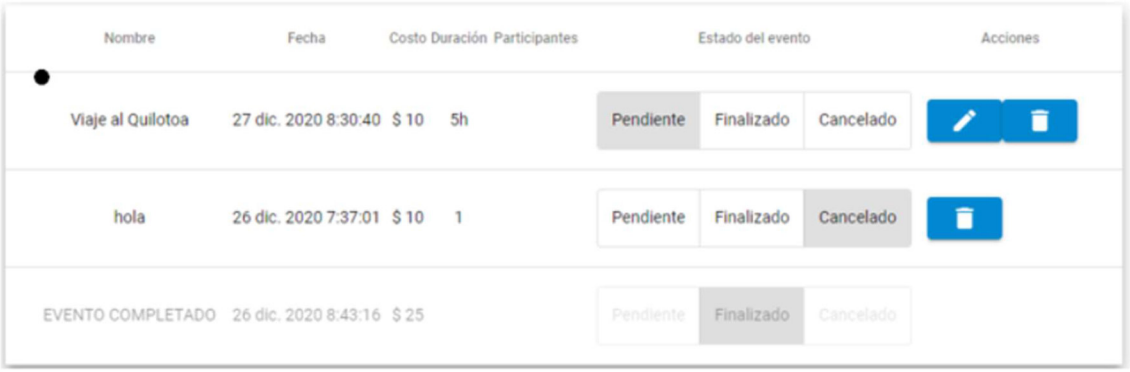

Ilustración 64. Listado de eventos

La modificación de un evento se la puede realizar de dos formas.

1. En caso de modificar solo el estado se lo debe realizar desde la lista de eventos, seleccionando el nuevo estado para el evento.
2. En caso de modificar información del evento se debe seleccionar el botón de edición (lápiz)

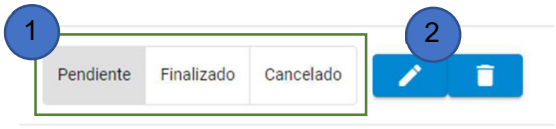

Ilustración 65. Tipos de modificación de evento

En caso de seleccionar el botón de edición, se mostrará una pequeña ventana la cual indica cómo se visualiza el evento en la aplicación y permitirá modificar la información del evento, como se lo muestra en la siguiente ilustración.

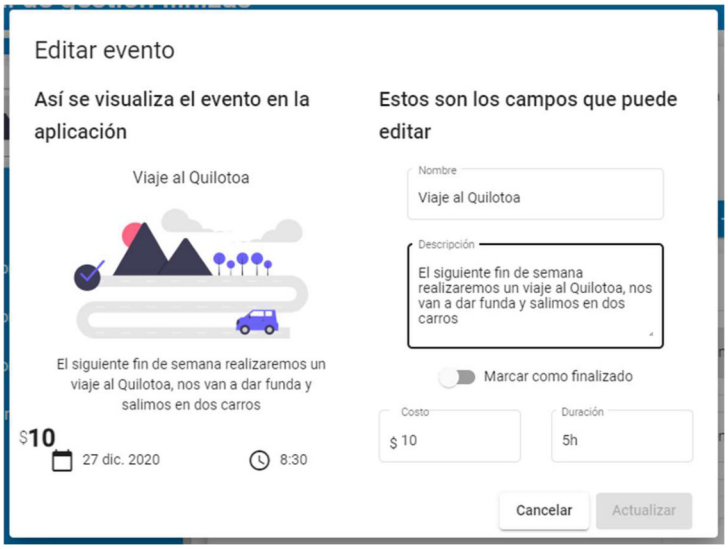

Ilustración 66. Modificar informació de evento

Dentro de la aplicación móvil, se tiene listados los eventos **PENDIENTES,** como se muestra en la siguiente ilustración.

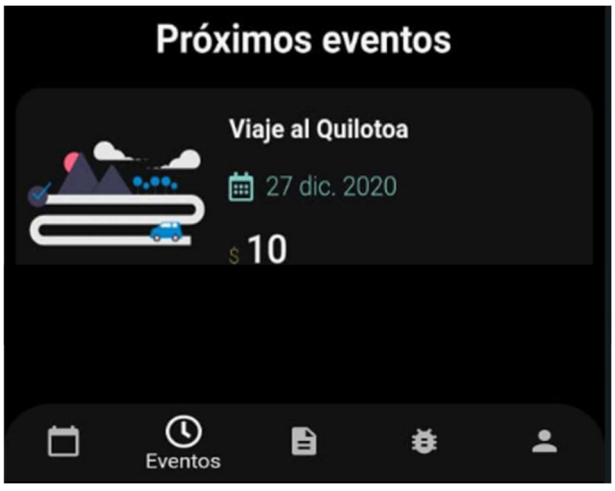

Ilustración 67. Sección de eventos

Para inscribirse en un evento, el usuario debe seleccionar un evento, y a continuación se le mostrará el detalle, en el cual podrá visualizar más información sobre el evento y le dará la opción de registrarse, en la siguiente ilustración se muestra esta pantalla de detalle.

Ilustración 68. Detalle de evento

Un usuario tiene la capacidad de registrarse en un evento, así como eliminar su registro en el mismo. El servicio que se creó para esta acción se basa en un parámetro recibido en la petición, el cual indica la acción por tomar; como se lo muestra en la siguiente ilustración.

```
fn: async function (inputs, exits) {
  const { userId, eventId, action } = inputs;
  if (action === 'ADD') {
    await Event.addToCollection(eventId, 'interested', userId);
  } else if (action === 'REMOVE') {
    await User.removeFromCollection(userId, 'events', eventId);
  }
  return exits.success({ id: eventId });
},
```

Ilustración 69. Servicio de registro a evento

- **Notas de entrevistas**

En la aplicación móvil se creó una sección para los postulantes de la residencia, la cual solo es visible para los administradores; en esta sección se lista el estado de las entrevistas de los postulantes, como lo indica la siguiente ilustración.

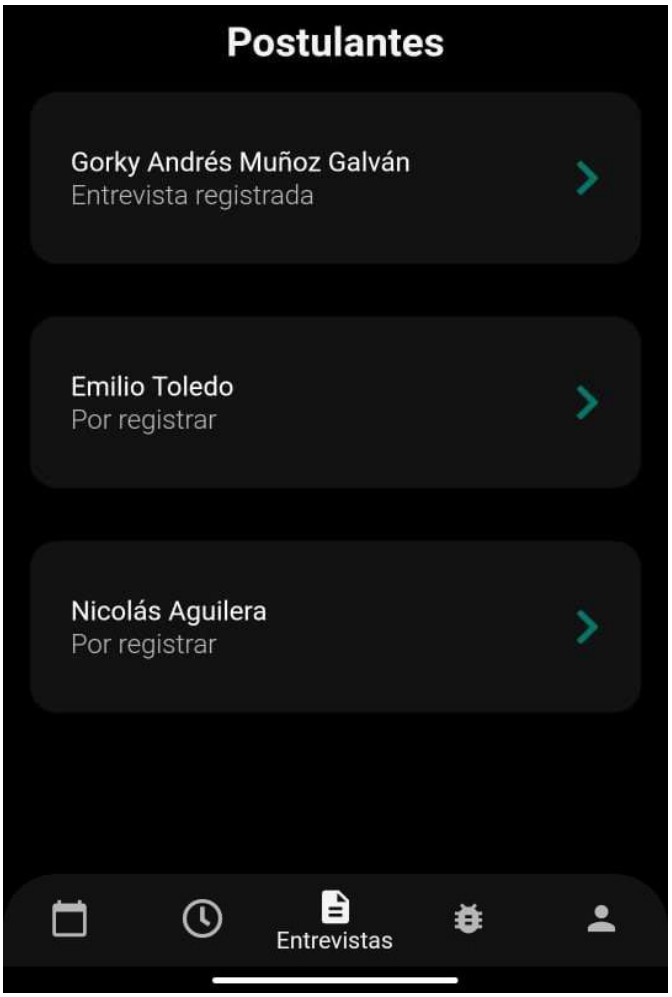

Ilustración 70. Sección de postulantes

En caso de que no existan registros de entrevista para el postulante, la aplicación permitirá ingresar información relevante de la entrevista, mediante un editor de texto enriquecido y con una longitud máxima de 2500 caracteres, como se lo indica en la siguiente ilustración.

Ilustración 71. Registro de notas de entrevista

Mientras que, si ya existe un registro de la entrevista para el postulante, se mostrará la información registrada.

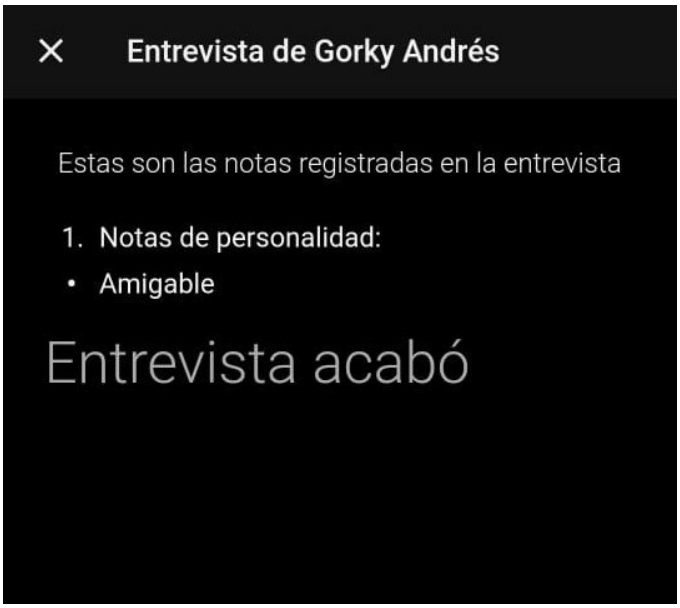

Ilustración 72. Notas de entrevista registradas

- **Ordenar tipos de comida**

Para ordenar los tipos de comida se agregó en el modelo una propiedad que indica la posición para cada tipo de comida y finalmente, cuando se agrupa la configuración de comidas, se ordena por el tipo de comida.

```
dates.forEach((date) => {
  const day = date.getUTCDay() > 0 ? date.getUTCDay() : 7;
  const foodGroupByDaySorted = groupedByDay[day]
  .sort(
    (a,b) =>
      a.food.order - b.food.order
  );
  configuration.push({ date, configuration: foodGroupByDaySorted });
});
```

Ilustración 73. Ordenamiento de comidas

```
order: {
  type: 'number',
  columnName: 'position_order'
},
```

Ilustración 74. Atributo de orden en el modelo de comida

3.8.4. Daily Scrum

En el transcurso de este sprint se tuvo 6 reuniones diarias, debido a que no se tenía dudas por resolver, más que puntos específicos acerca de los eventos. En estas reuniones se presionó la aprobación del editor de texto enriquecido, mostrando posibles resultados finales. De igual manera, en estas reuniones se definió el orden que debería tener cada comida y asistencia, ya que estaban siendo un bloqueante y riesgo para completar el sprint.

3.8.5. Revisión del Sprint

Con la nueva versión desplegada de la aplicación, se realizó la presentación del sprint, en la cual se validó que se cumplió el objetivo y de igual manera, se cumplió con los nuevos requerimientos añadidos.

3.8.6. Adaptación del sprint

Se validó que un criterio de aceptación de la consulta de eventos estaba presentando un error, por lo que será solventado en el siguiente sprint.

Épica	Código	Nombre	Complejidad	Prioridad
TI04	BUG01	No se listan los interesados de un evento	8	2

Tabla 16. Adaptación del sprint 5

3.8.7. Retrospectiva del Sprint

Para este sprint se identificaron los siguientes puntos:

- Fortalezas:
  - Aprobación rápida de diseños por parte del dueño de producto
  - Comprensión total de los servicios cubiertos en la aplicación móvil
- Objetivo: La aplicación móvil ha logrado reemplazar los servicios manuales de la residencia
- Debilidades: Poca disponibilidad de tiempo de los interesados en el producto
- Amenazas: Administradores mayores que no cuentan con un Smartphone

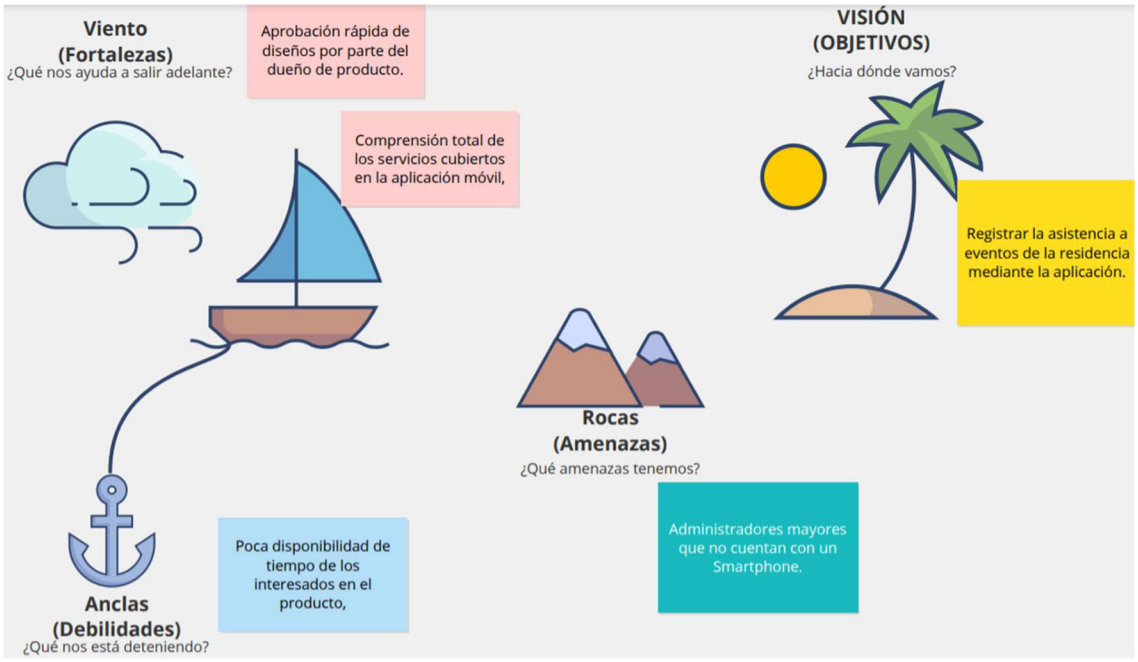

Ilustración 75. Retrospectiva de Sprint 5

3.9. Sprint 6

3.9.1. Objetivo del Sprint

Permitir que el usuario modifique su información y se comunique con los administradores de la residencia.

3.9.2. Sprint Backlog

Código	Nombre
HI31	Olvidó contraseña
HI34	Editar información personal
HI03	Crear sección servicios
HI04	Crear formulario de contacto
HI19	Asignar tutores a residentes
HI20	Eliminar administrador
BUG01	No Se listan los interesados de un evento

Tabla 17. Sprint Backlog del Sprint 6

3.9.3. Ejecución del Sprint

Para cuando el usuario necesita recuperar su contraseña porque la olvidó, se hace uso de la aplicación móvil y de la página web informativa. Primero debe realizar una solicitud de cambio de contraseña en la aplicación móvil, como se lo muestra en la siguiente ilustración

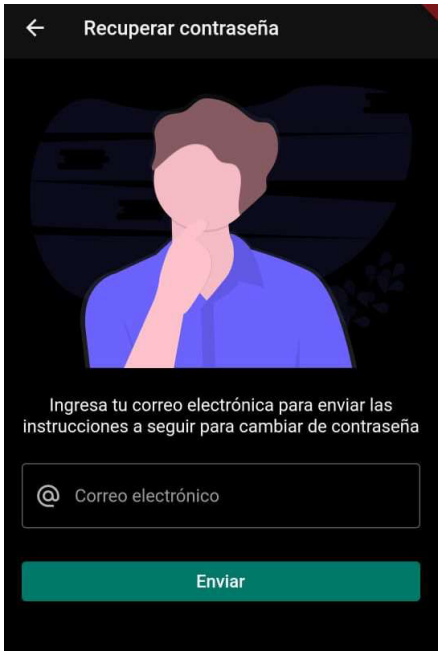

Ilustración 76. Pantalla de recuperar contraseña

Una vez realizada la solicitud de recuperar contraseña el sistema envía un correo electrónico al usuario el cual le redirige a la página web informativa en donde podrá ingresar su nueva contraseña. Para que el usuario no tenga que volver a ingresar su correo electrónico, se envía un JWT que contiene la información del correo e información adicional que permitirán validar que el token no ha sido modificado ni que la información de este ha sido comprometida,

además, para asegurar la solicitud, se le da una duración de una hora al token y se valida con un registro en base de datos su ID.

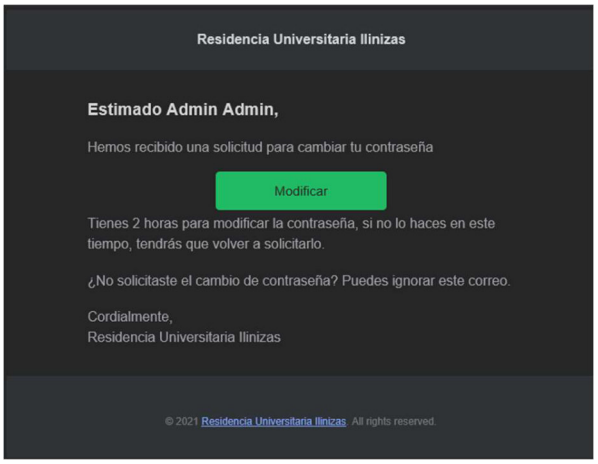

Ilustración 77. Correo de modificación de contraseña

Ilustración 78. Pantalla de cambiar contraseña

- **Modificar datos personales**

En la sección perfil, el usuario tiene la posibilidad de modificar sus datos, de los cuales no puede modificar su nombre y su email, tal como se muestra en la siguiente ilustración:

Ilustración 79. Pantalla de datos personales

- **Sección Servicios**

Dentro de la página informativa se creó la sección de servicios, la cual muestra información de los servicios que brinda la residencia. Esta información es configurable y, por ende, el director puede modificarla a su gusto. Añadiendo o eliminando servicios. En la siguiente ilustración se muestra la sección de servicios.

Ilustración 80. Servicios de la residencia

Finalmente, se añadió una sección en la página informativa para contactarse con los administradores de la residencia. Esto permitirá tener un contacto directo con el interesado sin que haya un compromiso de por medio, como lo es con el registro de inscripción. En la siguiente ilustración se muestra visualiza esta sección.

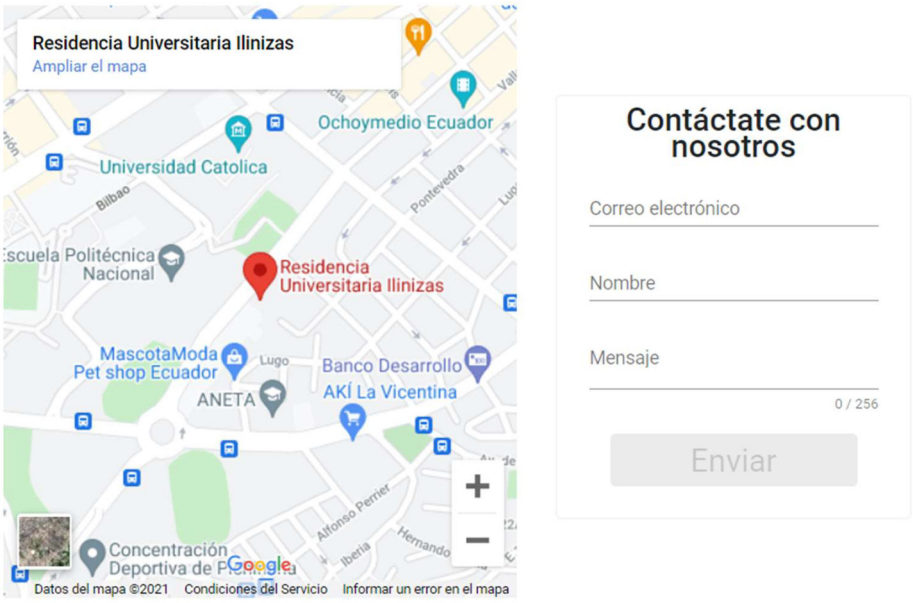

Ilustración 81. Sección contáctate con nosotros

Estas preguntas pueden ser visualizadas en el portal de gestión, como se lo muestra en la siguiente ilustración:

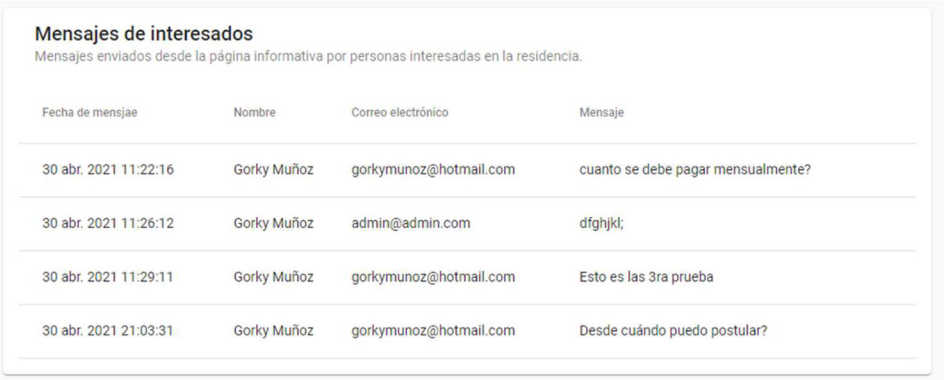

Ilustración 82. Mensajes de interesados

3.9.4. Daily Scrum

En este sprint se tuvo 4 reuniones debido a que la complejidad y prioridad de las tareas no eran altas. Se discutió el dueño del producto acerca del manual de usuario.

3.9.5. Revisión del Sprint

En el último sprint se realizó una demostración completa de la aplicación. Desde el registro de un postulante, hasta su cambio de contraseña. Por lo que se validó que, al culminar este sprint, se tiene un producto altamente viable para desplegar en producción.

3.9.6. Retrospectiva del Sprint

- Fortalezas: El equipo maduró mucho debido a que en este sprint el tiempo no estuvo ajustado.
- Objetivo: Se obtuvo un producto potencialmente desplegable
- Debilidades: Se necesita una inducción a residentes y administradores.
- Amenazas: El tiempo y grupo de pruebas es reducido.

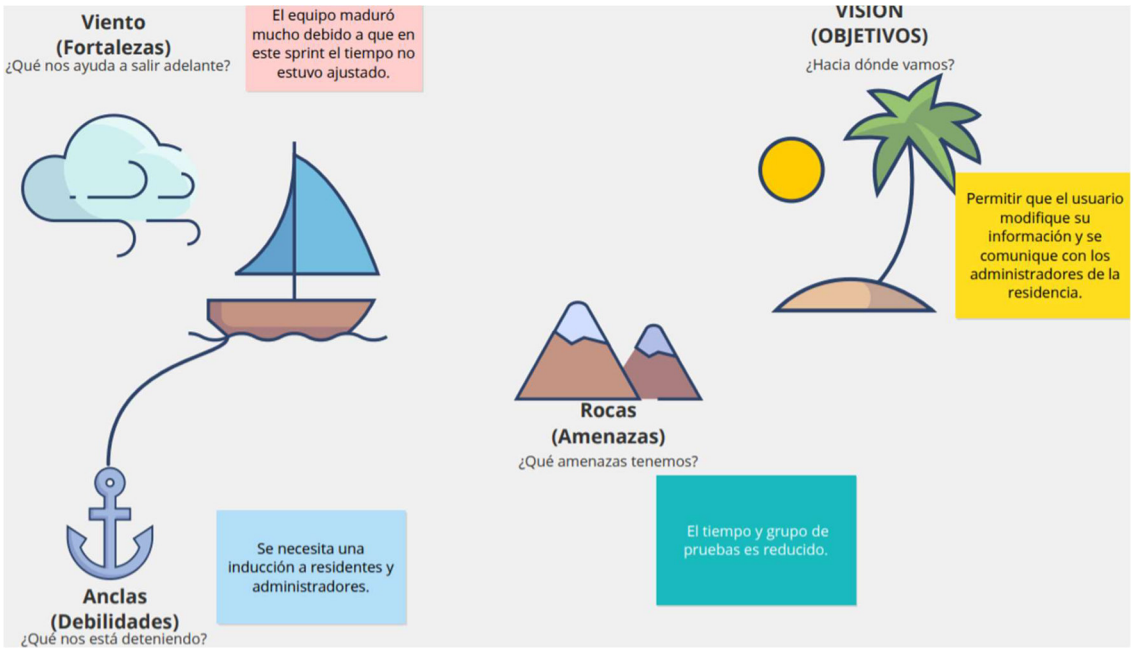

Ilustración 83. Retrospectiva Sprint 6

CAPÍTULO 5. RESULTADOS Y DISCUSIÓN

En este capítulo se presentan los resultados obtenidos en las pruebas de usabilidad (ANEXO 8), se definió una guía rápida para para los casos de usabilidad definidos (ANEXO 7) la cual sirvió para las personas escogidas para realizar pruebas de las funcionalidades más importantes y de uso más frecuente del sistema. Las pruebas de usabilidad se realizaron a un grupo de 24 personas, entre los cuales 6 forman parte de los administradores de la residencia, 4 exresidentes, 10 residentes y 4 externos.

La encuesta se realizó con el objetivo de determinar la facilidad de uso y entendimiento del sistema por parte de los posibles usuarios. Se realizaron un total de 10 preguntas cualitativas

y 2 preguntas de opción abierta. Las preguntas de opción abierta tuvieron el objetivo de identificar dificultades, errores y mejoras para el sistema. El resultado de dichas preguntas fue el siguiente:

- **Pregunta Nro. 1 – ¿Qué tan complejo fue registrar una postulación?**
  Se obtuvo que el 75% de los encuestados definieron a la funcionalidad con una complejidad media, mientras que el 16,7% indicó que la funcionalidad es fácil de usar y un 8,3% encontraron la funcionalidad como difícil de utilizar.

Ilustración 84. Resultado de la primera pregunta de la encuesta de usabilidad

- **Pregunta Nro. 2 – ¿Qué tan complejo fue registrar una solicitud de contáctate con nosotros?**
  Se obtuvo que el 54,2% de los encuestados definieron la funcionalidad como muy fácil de usar, mientras que el 45,8% indicó que la funcionalidad es fácil de usar.

Ilustración 85. Resultado de la segunda pregunta de la encuesta de usabilidad

- **Pregunta Nro. 3 – ¿Qué tan complejo fue iniciar sesión en el portal de gestión?**
  El 50% indicó que fue fácil de iniciar sesión, mientras que el 45,8% definieron a la funcionalidad como muy fácil de usar y el 4,2% señaló que tiene una complejidad normal de uso.

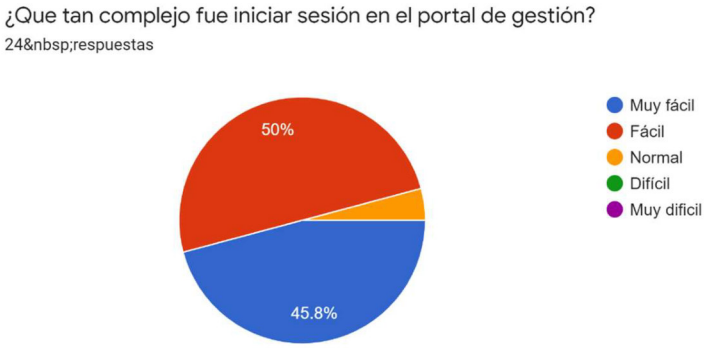

Ilustración 86. Resultado de la tercera pregunta de la encuesta de usabilidad

- **Pregunta Nro. 4 – ¿Qué tan complejo fue crear un residente?**
  El 54,2% de los encuestados definió la funcionalidad como fácil de utilizar, el 41,7% encontraron a la funcionalidad con una complejidad normal y el 4,1% indicaron que la funcionalidad fue muy fácil de utilizar.

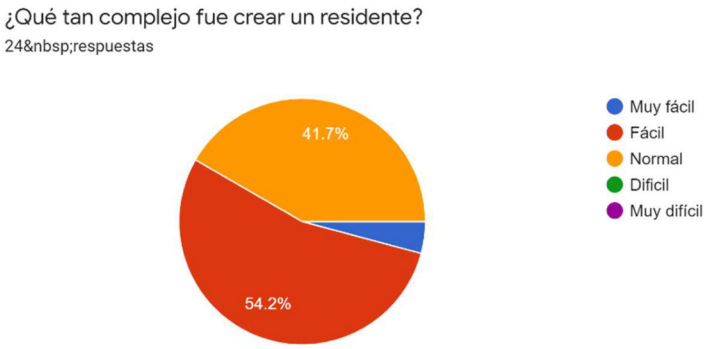

Ilustración 87. Resultado de la cuarta pregunta de la encuesta de usabilidad

- **Pregunta Nro. 5 – ¿Qué tan complejo fue crear un administrador?**

El 50% indicó que la funcionalidad cuenta con una complejidad normal, mientras que el 37,5% definió a la funcionalidad como fácil de usar y el 12,5% consideró que fue difícil de utilizar.

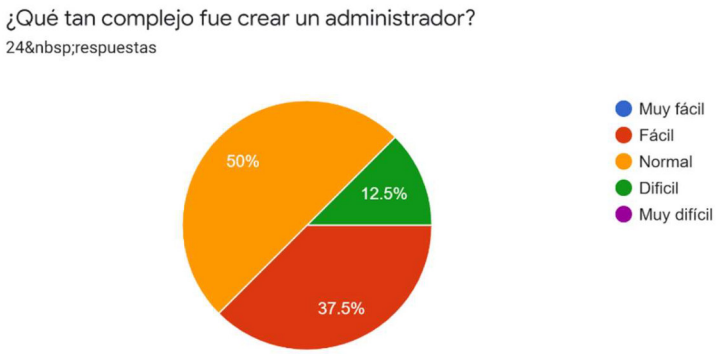

Ilustración 88. Resultado de la quinta pregunta de la encuesta de usabilidad

- **Pregunta Nro. 6 – ¿Qué tan complejo fue crear un evento?**
  El 58,3% indicó que la funcionalidad es fácil de usar, mientras que el 25% encontró a la funcionalidad con una complejidad normal y el 16,7% la señaló como muy fácil de completar.

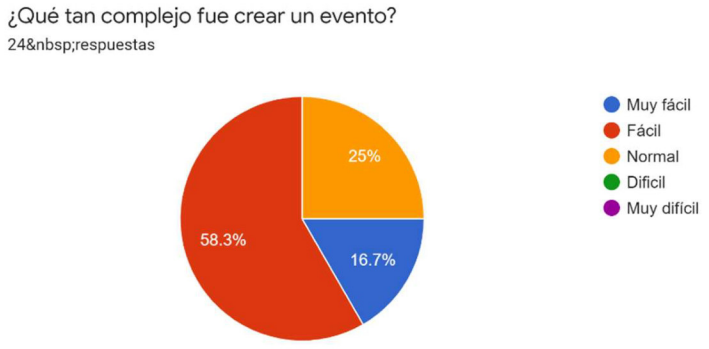

Ilustración 89. Resultado de la sexta pregunta de la encuesta de usabilidad

- **Pregunta Nro. 7 – ¿Qué tan complejo fue iniciar sesión en la aplicación móvil?**
  El 50% indicó que fue fácil iniciar sesión en la aplicación móvil, mientras que el 41,7% señaló que la funcionalidad tiene una complejidad normal de uso y el 8,3% la definió como muy fácil de usar.

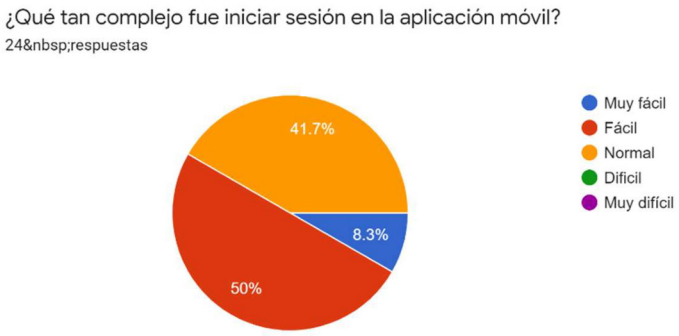

Ilustración 90. Resultado de la séptima pregunta de la encuesta de usabilidad

- **Pregunta Nro. 8 – ¿Qué tan complejo fue registrar una asistencia a una comida diaria?**

  Se obtuvo que el 45,8% de encuestados encontraron a la funcionalidad como fácil de usar, el 41,7% definió que la funcionalidad cuenta con una complejidad normal, el 8,3% tuvo problemas con la funcionalidad y la declaró como difícil de utilizar y el 4,2% la definieron como muy fácil de utilizar.

Ilustración 91. Resultado de la octava pregunta de la encuesta de usabilidad

- **Pregunta Nro. 9 – ¿Qué tan complejo fue registrarse a un evento?**

  El 58,3% de los encuestados encontrar a la funcionalidad como fácil de usar, mientras que el 25% indicó que cuenta con una complejidad media y el 16,7% declaró que fue muy fácil de realizar.

Ilustración 92. Resultado de la novena pregunta de la encuesta de usabilidad

- **Pregunta Nro. 10 – ¿Qué tan complejo fue generar el informe de asistencia a comidas?**

  El 62,5% definió a la funcionalidad como fácil de realizar, el 20,8% indicó que fue fácil de usar, el 12,5% la señaló como una funcionalidad de complejidad media y el 4,2% la consideró como difícil de utilizar.

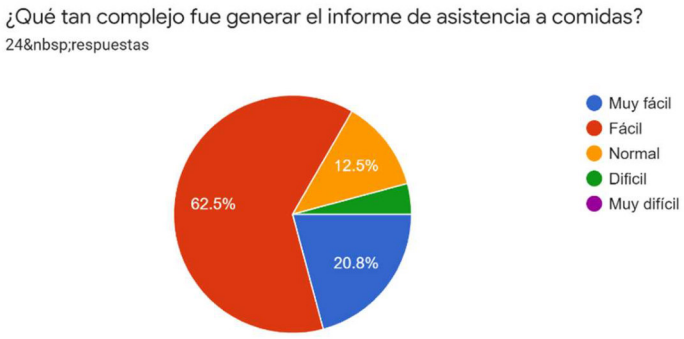

Ilustración 93. Resultado de la décima pregunta de la encuesta de usabilidad

Finalmente, se muestra el total de respuestas obtenidas por cada opción de respuesta. Donde, el 47,9% de los encuestados definieron a las funcionalidades como fáciles de utilizar, a diferencia del 31,7 que declaró las funcionalidades con una complejidad media, el 17,1% indicaron que las funcionalidades fueron muy fáciles de utilizar y el 3,3% señaló que las funcionalidades fueron difíciles de realizar. Dentro del 3,3% se evidenció que estos resultados fueron receptados en su gran mayoría en el formulario de inscripción, lo cual se contrasta con las preguntas abiertas, en donde los usuarios recomendaron mejores en el formulario de

inscripción debido a la gran cantidad de información que se requiere y a su vez, a componentes que no tenían un comportamiento intuitivo.

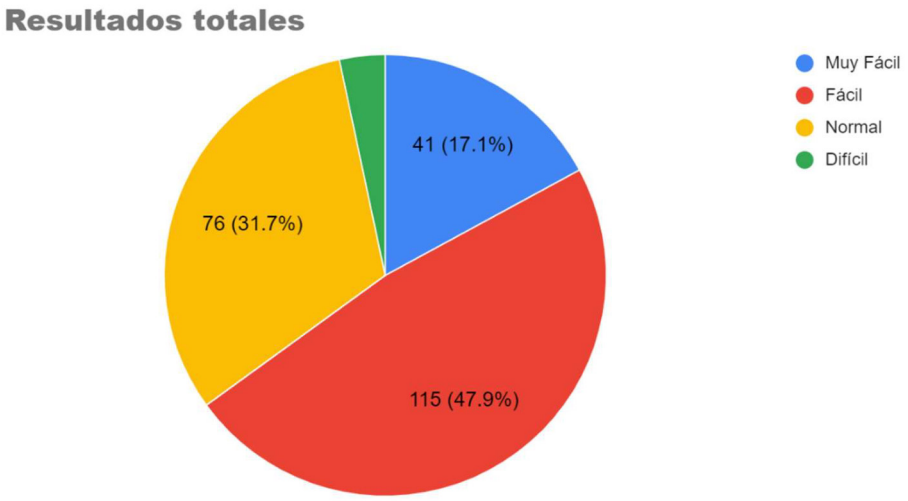

Ilustración 94. Resultados totales de la encuesta de usabilidad

CAPÍTULO 6. CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

- Se desarrolló un sistema compuesto por dos aplicaciones web y una móvil, capaz de reemplazar la gestión manual en la cual los registros de asistencia a comidas eran los más vulnerables, siendo modificados sin permiso diariamente. Gracias al sistema se eliminó este riesgo y posibles gastos generados por la modificación de la información.
- Se comprobó que las historias de usuario definidas en el proyecto no cumplían con todas las características de una historia de usuario (INVEST), lo que derivó en un alto porcentaje de ambigüedad al momento de realizar el desarrollo y por ende, causó que los usuarios requieran una mayor explicación de ciertas funcionalidades.
- Se comprobó que el uso de un patrón de arquitectura en el desarrollo de un producto software permite mantener una mejor estructura y organización del código. Por lo que, realizar cambios en el código no se vuelve una tarea compleja y no se necesitan perder tiempos en volver a analizar todo nuevamente.
- Se desarrolló la aplicación móvil multiplataforma utilizando Flutter, el cual permitió avanzar en el desarrollo siguiendo los diseños de los Mockups sin tener una mayor complicación o riesgo de cumplimiento, ya que no existió tareas remanentes de la aplicación móvil en el transcurso de los sprints.

- Se realizó una encuesta dirigida a los miembros de la residencia, la cual permitió evidenciar que, si bien el sistema cumple con las necesidades presentadas, existen funcionalidades a las cuales se pueden aplicar mejoras tanto de diseño como de flujo, sin embargo, al no tener preguntas dirigidas a la mejora de funcionalidades, se tuvo que realizar un análisis manual para entender e interpretar estos resultados.
- Se evidenció que el tamaño de descarga de la tienda de la aplicación móvil multiplataforma fue alto; una vez instalada y con información almacenada, se duplicó. Mientras que el tamaño de una aplicación nativa que incluía una mayor cantidad de funcionalidades fue mucho más pequeño.
- El uso de la metodología Scrum en el proyecto permitió agilizar el proceso de implementación y desarrollo de las aplicaciones, de igual manera, permitió que cada iteración realizada, tenga como salida un producto potencialmente desplegable y puesto en producción. Esto fue resultado de la priorización y división de tareas para cada sprint, lo cual permitió evitar sobrecargas de trabajo en el equipo de desarrollo.
- El uso de prácticas DevOps permitió automatizar varias fases del ciclo de vida del producto. Si bien al inicio la configuración del ambiente ocupó un tiempo considerablemente alto en investigación e implementación, el beneficio que aportó a lo largo del proyecto fue inmenso. A partir del primer despliegue continuo, no existió intervención del equipo de desarrollo, lo cual brindó la facilidad para que el tiempo de desarrollo no sea perdido en tareas repetitivas y sea aprovechado al máximo. Lo que derivó en cumplir a tiempo el objetivo de cada iteración del proyecto.
- El desarrollo de pruebas unitarias permitió validar que las funciones más importantes del proyecto, tales como Inicio de sesión, configuración y registro de asistencia a comidas diarias, y registro a eventos; se comportaban de manera correcta y cumplían con su objetivo. Esto permitió validar la eficiencia del código del proyecto y a su vez, mejorar la calidad de este, consiguiendo un código mucho más mantenible y escalable para futuros requerimientos.

5.2. RECOMENDACIONES

- En caso de utilizar frameworks que permitan crear aplicaciones móviles híbridas (Ionic, React Native, Flutter), se recomienda mantener actualizadas las versiones de cada librería y del framework, debido a que las plataformas nativas se mantienen cambiando, por lo que, se llegan a presentar problemas causados por compatibilidad de versiones. Estos problemas son muy difíciles de resolver porque no se encuentra el origen de este. Mientras que, si se mantiene actualizadas las versiones, en caso de dar un problema, se puede buscar una solución guiándose por la versión específica.

- En caso de necesitar enviar correos electrónicos automáticos, no se recomienda utilizar una cuenta Gmail, debido a que Google considera que el acceso de un tercero (servicio de envío de correo automático) a una cuenta de correo electrónico es una brecha de seguridad para el usuario.
- En caso de implementar Firebase en una aplicación, se recomienda implementar el servicio gratuito de Crashlytics debido a que registra de manera automática cualquier tipo de error que se presente en la aplicación.
- Para agrupar varias aplicaciones como un solo servicio, se recomienda usar Docker Compose, ya que permite crear varios contenedores en una sola imagen, por lo que, el sistema se vuelve portable y facilita la ejecución de este en un nuevo entorno.
- En caso de requerir exponer una aplicación local o una base de datos local a la web, se recomienda utilizar la herramienta **ngrok** la cual crea túneles seguros hacia la dirección local del equipo.
- En caso de utilizar fastlane para integrar el despliegue continuo de una aplicación móvil (Android, iOS, Flutter), se recomienda utilizar un equipo con sistema operativo Linux o Mac, debido a que se necesita configurar variables globales para el correcto funcionamiento de la herramienta. Estas configuraciones son ignoradas en un equipo Windows, por lo que, se debe invertir más tiempo de investigación y pruebas para solucionar dichos problemas.
- Se recomienda implementar la práctica de integración continua, ya que permite validar las nuevas y las previas funcionalidades del sistema, sin realizar tareas adicionales. Haciendo que el código en los repositorios sea solamente código testeado y sin posibles errores.
- Se recomienda realizar un análisis de las aplicaciones que son desplegadas en servicios de la nube, ya que estos servicios gratuitos en la mayoría de los casos cuentan con límites de uso o almacenamiento; por lo que, es recomendable mantener una revisión constante para que no se realizan facturas o problemas con las aplicaciones desplegadas.

REFERENCIAS BIBLIOGRÁFICAS

[1] «Residencia Universitaria Ilinizas,» [En línea]. Available: http://www.ilinizas.org/index.html. [Último acceso: 14 Mayo 2020].

[2] K. Schwaber y J. Sutherland, «Scrum Guides,» [En línea]. Available: https://www.scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf. [Último acceso: 3 12 2020].

[3] P. Deemer, G. Benefield, C. Larman y B. Vodde, «A Lightweight Guide to the Theory and Practice of Scrum,» 2012. [En línea]. Available: http://goodagile.com/scrumprimer/. [Último acceso: 11 Enero 2021].

[4] S. Sharma, D. Sarkar y D. Gupta, «Agile Processes and Methodologies: A Conceptual Study,» *International Journal on Computer Science and Engineering,* vol. 4, nº 0975-3397, p. 895, 2012.

[5] J. D. &. K. Daniels, Effective DevOps.

[6] M. Soni, DevOps Bootcamp.

[7] W. Sanchez, «La usabilidad en Ingeniería de Software: definición y características,» *Ingnovación. Revista de Ingeniería e Innovación de la Facultad de Ingeniería, Universidad Don Bosco,* vol. 1, nº 2, pp. 7-21, 2011.

[8] N. Rozentals, Mastering TypeScript, Birmingham: Packt Publishing Ltd, 2015.

[9] M. contributors, «What is JavaScript?,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript. [Último acceso: 19 Enero 2021].

[10] S. Sinha, Quick Start Guide to Dart Programming, Howrah: Apress, Berkeley, CA, 2020.

[11] M. contributors, «CSS,» MDN Web Docs, 25 Octubre 2020. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/CSS. [Último acceso: 19 Enero 2021].

[12] M. contributors, «HTML: HyperText Markup Language,» MDN Web Docs, 10 Diciembre 2020. [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML. [Último acceso: 19 Enero 2021].

[13] O. Ben-Kiki, C. Evans y I. döt Net, «YAML Ain't Markup Language (YAML™) Version 1.2,» 1 Octubre 2009. [En línea]. Available: https://yaml.org/spec/1.2/spec.html. [Último acceso: 19 Enero 2021].

[14] «Visual Studio Code: Getting Started,» [En línea]. Available: https://code.visualstudio.com/docs. [Último acceso: 19 Enero 2021].

[15] «Meet Android Studio,» 28 Octubre 2020. [En línea]. Available: https://developer.android.com/studio/intro. [Último acceso: 19 Enero 2021].

[16] Docker, «Docker overview,» [En línea]. Available: https://docs.docker.com/get-started/overview/. [Último acceso: 19 Enero 2021].

[17] Google, «FIrebase fundamentals,» [En línea]. Available: https://firebase.google.com/docs/projects/learn-more. [Último acceso: 30 04 2021].

[18] GitLab, «GitLab Docs,» [En línea]. Available: https://docs.gitlab.com/ee/README.html. [Último acceso: 19 Enero 2021].

[19] Dbeaver, «Debeaver Community About,» [En línea]. Available: https://dbeaver.io/about/. [Último acceso: 30 04 2021].

[20] Adobe, «Adobe XD,» [En línea]. Available: https://www.adobe.com/es/products/xd.html. [Último acceso: 30 04 2021].

[21] Salesforce, «Heroku,» [En línea]. Available: https://www.heroku.com/home. [Último acceso: 30 04 2021].

[22] Postman, «About Postman,» [En línea]. Available: https://www.postman.com/company/about-postman/. [Último acceso: 30 04 2021].

[23] Node.js, «Acerca de Node.js,» [En línea]. Available: https://nodejs.org/es/about/. [Último acceso: 30 04 2021].

[24] «What is V8?,» [En línea]. Available: https://v8.dev/. [Último acceso: 30 04 2021].

[25] Sails, «What is Sails?,» [En línea]. Available: https://sailsjs.com/whats-that/. [Último acceso: 30 04 2021].

[26] Angular, «Introduction to Angular Docs,» [En línea]. Available: https://angular.io/docs. [Último acceso: 30 04 2021].

[27] Flutter, «Flutter Get Started,» [En línea]. Available: https://flutter.dev/. [Último acceso: 30 04 2021].

[28] A. Villacís, Interviewee, *Servicios de Residencia Ilinizas.* [Entrevista]. 25 06 2020.

[29] D. Radigan, «ATLASSIAN Agile Coach,» [En línea]. Available: https://www.atlassian.com/agile/scrum/backlogs. [Último acceso: 5 7 2021].

[30] I. Sommerville, «Software Engineering,» de *Software Engineering*, 10th ed., Scotland, Pearson, 2015, pp. 160-162.

[31] S. Peyrott, «JWT HANDBOOK,» de *The JWT Handbook*, Auth0 Inc., 2016, pp. 8-9.

[32] S. Suri, «FlutterPub,» 26 08 2018. [En línea]. Available: https://medium.com/codechai/architecting-your-flutter-project-bd04e144a8f1. [Último acceso: 01 11 2020].

[33] V. Driessen, «Nvie,» 05 01 2010. [En línea]. Available: https://nvie.com/posts/a-successful-git-branching-model/. [Último acceso: 31 07 2020].

ANEXOS

ANEXO 1: ENTREVISTA

Enlace al documento de la transcripción de reuniones con el director de la residencia: https://bit.ly/2VOV9Zd

ANEXO 2: HISTORIAS DE USUARIO

Enlace al tablero en trello: https://bit.ly/3yTgONw

ANEXO 3: DIAGRAMA DE BASE DE DATOS

Enlace a la imagen del diagrama de base de datos: https://bit.ly/36zRUGV

ANEXO 4: BOCETOS DE INTERFAZ GRÁFICA DE LA APLICACIÓN WEB INFORMATIVA

Enlace al documento en formato digital: https://bit.ly/2VjJbGX

ANEXO 5: BOCETOS DE INTERFAZ GRÁFICA DE LA APLICACIÓN WEB DE GESTIÓN DEL SISTEMA

Enlace al documento en formato digital: https://bit.ly/3nnnPEv

ANEXO 6: BOCETOS DE INTERFAZ GRÁFICA DE LA APLICACIÓN MÓVIL

Enlace al documento en formato digital: https://bit.ly/3BMc4ev

ANEXO 7: CASOS DE USABILIDAD

Enlace al documento en formato digital: https://bit.ly/3rekSpl

ANEXO 8: RESULTADOS DE LA ENCUESTA DE USABILIDAD

Enlace al documento en formato digital: https://bit.ly/3hKAvBK

ANEXO 9: DIAGRAMA DE COMPONENTES

Enlace al documento en formato digital: https://bit.ly/2Xg1bTB