Informe de proyecto final de carrera en
Ingeniería en Sistemas de Información

Desarrollo de prototipo de **aplicación** multiplataforma para la optimización del proceso de **inventariado de inmuebles**

ALUMNO: TOMÁS VALENTINIS

Director: Ing. Ezequiel Aranda

UTN

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL DE SANTA FE

MARZO 2022

1. Introducción

El siguiente documento tiene como objetivo presentar el proyecto final de carrera del alumno Tomás Alfredo Valentinis titulado: "**Desarrollo de Prototipo de Aplicación Multiplataforma para la Optimización del Proceso de Inventariado de Inmuebles**".

A fines del año 2019, quien suscribe, en carácter de alumno de la carrera de Ingeniería en Sistemas de Información entabló una conversación con una inmobiliaria de la ciudad de Santa Fe, Argentina conocida como "SAMAR", la cual se define en este documento como "el cliente".

A través de diversas reuniones se determinaron necesidades y requerimientos para el desarrollo de un prototipo de aplicación multiplataforma que proporciona las herramientas necesarias para la gestión de inventarios de inmuebles y organizar datos ingresados.

En el presente documento se incluirán todos los procedimientos involucrados en el desarrollo de un prototipo de aplicación y el testing de la misma. Sin embargo, no se incluyeron los procesos de implementación ni de mantenimiento.

1.1 Alcance del proyecto

Comenzando el año 2020 y en carácter de alumno avanzado de la carrera Ingeniería en Sistemas de información, se presentó la oportunidad de trabajar en algunas mejoras y optimización de flujos de trabajo de la inmobiliaria "SAMAR" desde un punto de vista tecnológico, a través de una solución de software.

El proyecto brindó al cliente la posibilidad de llevar a cabo algunas de sus tareas primordiales, como el relevamiento de inventarios, minimizando tiempos en relación a cómo se realizaban dichas tareas en un orden tradicional. Este trabajo forma parte de una serie de mejoras generales vinculadas con la digitalización y apoyo tecnológico que el cliente comenzó a realizar, en pos de mejorar su productividad empresarial.

1.2 Objetivo general

El objetivo principal del proyecto se basó en desarrollar una solución de software que optimice los recursos involucrados en el proceso de gestión de inventarios.

1.3 Objetivos específicos

Para alcanzar el objetivo general propuesto anteriormente se identificaron los siguientes objetivos específicos:

- Crear una aplicación multiplataforma accesible y adaptable a todos los dispositivos.
- Efectuar un seguimiento del proyecto a lo largo de todo el proceso, desde las etapas tempranas de comunicación con el cliente hasta la ejecución del plan de pruebas.
- Detectar problemas e identificar oportunidades de mejora en el proceso de relevamiento de inventarios tradicional.

1.4 Población

El producto desarrollado se encuentra destinado a los empleados de la inmobiliaria, equipo compuesto por 4 integrantes que trabajan en conjunto con la abogada de la empresa para definir los contratos de alquiler.

Se identificaron como clientes regulares a los dueños de los inmuebles a los que la inmobiliaria se encuentra prestando servicios, a dichas personas, se les envía una copia de todos los contratos que se realizan al alquilar un inmueble de su propiedad.

Se aclara que los principales clientes de la empresa "SAMAR" son todas aquellas personas que disponen de un inmueble para su venta o alquiler.

2. Marco teórico

2.1 Proceso de relevamiento de inventario

Se describe a este proceso como aquel destinado para relevar todos los elementos de un inmueble, creándose un documento compartido entre la inmobiliaria y los dueños de las viviendas. Dicho proceso requiere tiempo por parte de los empleados de la inmobiliaria, quienes son los encargados de dirigirse hasta los hogares para tomar fotos y recolectar información textual sobre el lugar.

Este proceso se realiza para relevar las condiciones del inmueble y para que cuando se termine el contrato de locación, los inquilinos tengan la obligación jurídica de entregar la propiedad tal cual consta en este inventario[1].

2.2 Situación del proceso de relevamiento de inventarios anterior al desarrollo de la aplicación multiplataforma

La inmobiliaria "SAMAR" disponía de un proceso de relevamiento de inventario de un inmueble que iniciaba al momento en que 2 empleados se dirigían a un inmueble, haciendo uso de 1 notebook para el ingreso de datos en un documento de texto y un celular para obtener imágenes del inmueble. Una vez terminado el relevamiento, los empleados daban formato al documento de texto generado, anexando las respectivas fotos. Este proceso queda reflejado en la Figura 1.

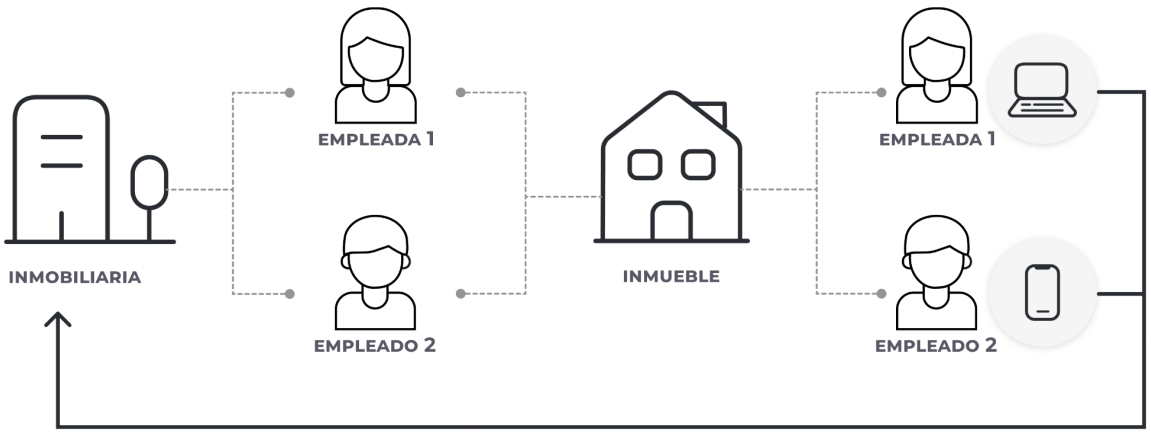


Figura 1 - Proceso de relevamiento de inventarios - Imagen de autoría propia

[1]Se recomienda consultar la sección 7.1 para evacuar dudas puntuales sobre roles de las personas mencionadas.

3. Fundamentación

3.1 Oportunidades de mejora del proceso original

Dentro del proceso de relevamiento previo a la implementación de la aplicación multiplataforma de la inmobiliaria, se observaron 2 oportunidades de mejora.

Para realizar el inventario, se necesitaba que 2 empleados concurrieran al inmueble, junto con 1 dispositivo móvil y 1 notebook (Figura 2).

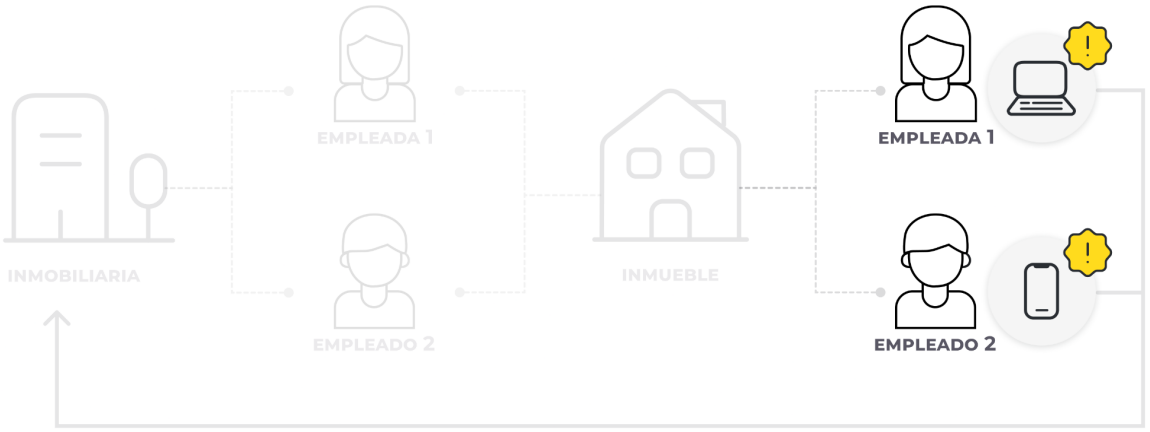


Figura 2 - Representación gráfica de la situación acutal con problemas resaltados (íconos en amarillo) - Imagen de autoría propia

Luego de un relevamiento inicial, los empleados se dirigían a la inmobiliaria donde llevaban a cabo el proceso de dar formato al documento de texto y anexar las fotos tomadas con el dispositivo móvil (Figura 3).

Figura 3 - Representación gráfica de la situación actual - Número 2 - Problema destacado (ícono en rojo) - Imagen de autoría propia

3.2 Propuesta de mejora

En base a los problemas detectados en la sección anterior (Figura 3), se explica cómo el desarrollo de una aplicación multiplataforma optimizó el proceso de relevamiento de un inventario.

Mejora 1: Con esta solución de software solo fue necesario que 1 empleado asista al inmueble, como se refleja en la Figura 4.

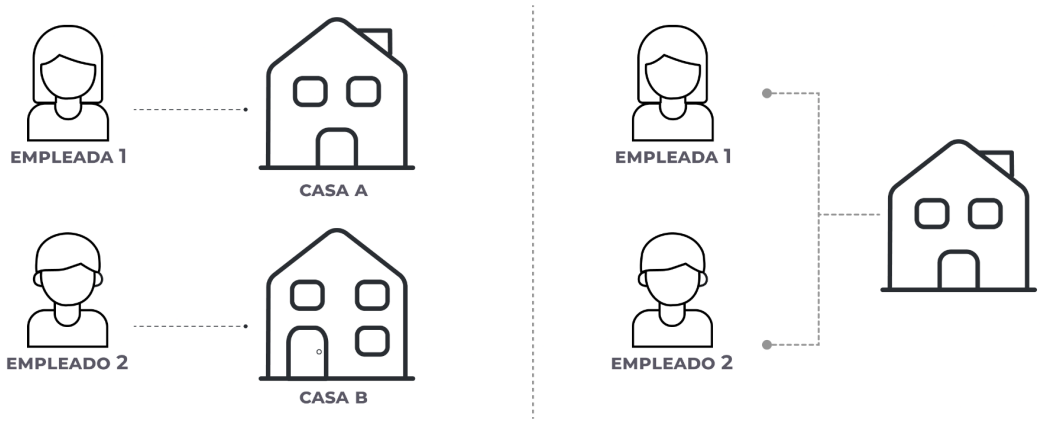


Figura 4 - Mejora 1 - Imagen de autoría propia

Mejora 2: Haciendo uso de la aplicación, los empleados ingresan desde un celular colocando solo los datos relevantes manteniendo un formato previamente definido y respaldado en la nube (Figura 5).

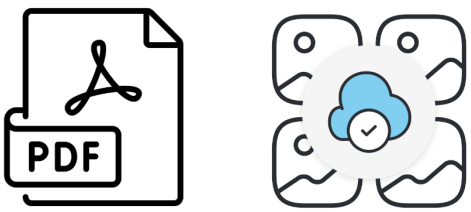


Figura 5 - Mejora 2 - Imagen de autoría propia

4. Gestión de proyecto

4.1 Modelo de desarrollo de software

4.1.1 Metodología Ágil

El dominio del problema se describe como "complejo" según el modelo cynefin creado por Dave Snowden (Figura 6). En base a esta complejidad y debido a que fue llevado a cabo por una sola persona, este proyecto utilizó la metodología Ágil, aunque se aclara que no se pudieron aplicar todos los aspectos y/o prácticas de las mismas como la ejecución de reuniones “Dailys”.

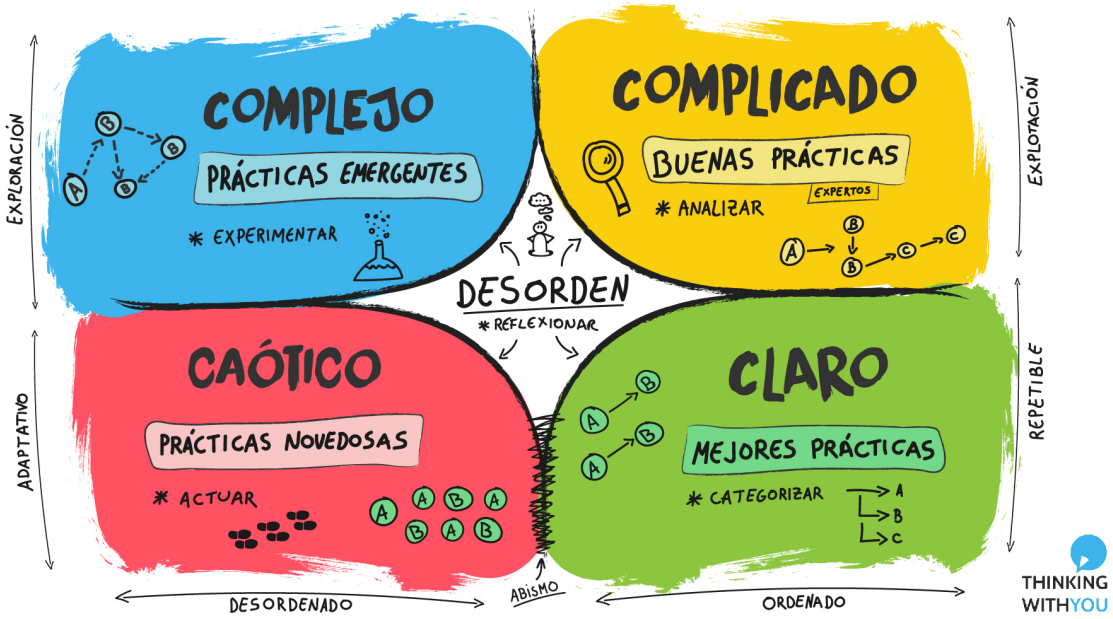


Figura 6 - Modelo Cynefin - Imagen extraída de El Modelo Cynefin. (2021, May 26). Thinking with You. https://thinkingwithyou.com/2021/05/modelo-cynefin-organizaciones/

Algunas de las prácticas del modelo ágiles que fueron adoptadas en el proyecto son:

- Retroalimentación constante: se presentó avances del producto al cliente de manera continua para identificar errores y comentarios.
- Listado de pendientes visible: Se observó en todo momento el flujo de historias de usuario en un RoadMap público.

Se tuvo en cuenta los siguientes aspectos en el desarrollo del proyecto:

- Reuniones periódicas con el cliente.
- Adaptación a circunstancias cambiantes y modificación de requisitos.
- División del trabajo en fases productivas, divididas en semanas.
- Medición de progreso.
- División de trabajo en tareas sencillas.

De esta forma se logró un flujo de trabajo más dinámico y adaptable a cambios, incorporándose pequeñas funcionalidades al software.

4.2 Metodología aplicada: Kanban

Escenario de partida que posibilitaron la elección de la metodología kanban:

- Falta de experiencia previa en la gestión de proyectos de desarrollo de sistemas, y en la definición y ejecución de planes de proyectos.
- Desconocimiento del dominio de negocio.
- Desarrollo del proyecto de forma unipersonal.

La elección de dicha metodología en el desarrollo del proyecto supuso las siguientes ventajas:

1. Mejora en la organización del flujo de trabajo: Proceso claro y consistente en cada fase del proyecto gracias a la organización de las tareas por estados y de la duración estimada para cada una de ellas.
2. Optimización de calidad del producto: Se detectaron problemas y se llevaron a cabo mejoras.
3. Reducción de carga de trabajo: al previsualizar todas las tareas se pudo organizar y asignar cada trabajo sin sobrecarga.
4. Simplificación de gestión visual: Se organizaron visualmente las tareas según sus estados.

4.2.1 Aplicando Kanban

Una vez relevados los requerimientos y definidas las historias de usuario (sección 4.7), se desarrollaron las tarjetas que las representan.

Los estados que tuvieron las historias de usuario fueron:

- **Pendientes**: se encontraban a la espera de su realización.
- **En curso**: pasan a este estado cuando se iniciaba una tarea relacionada.
- **Bloqueado:** se catalogaron como bloqueadas a las historias que debieron completar primero las tareas de otras historias para retomar su curso.
- **Testing**: se alcanzó este estado cuando se finalizaron las tareas asociadas a una historia, se realizaron las pruebas de las mismas y aquellas con las que se encontraron relacionadas.

- **Listo**: Se consideraron como lista cuando todas sus tareas se encontraban completas y las respectivas pruebas de cada una de ellas fueron resueltas con éxito.

A continuación se muestra una captura de pantalla del tablero en el momento inicial del proyecto (Figura 7). Si bien en la sección 4.7 se definieron las historias de usuario sus épicas no fueron representadas en este tablero debido a una limitación de la versión de la herramienta "Trello".

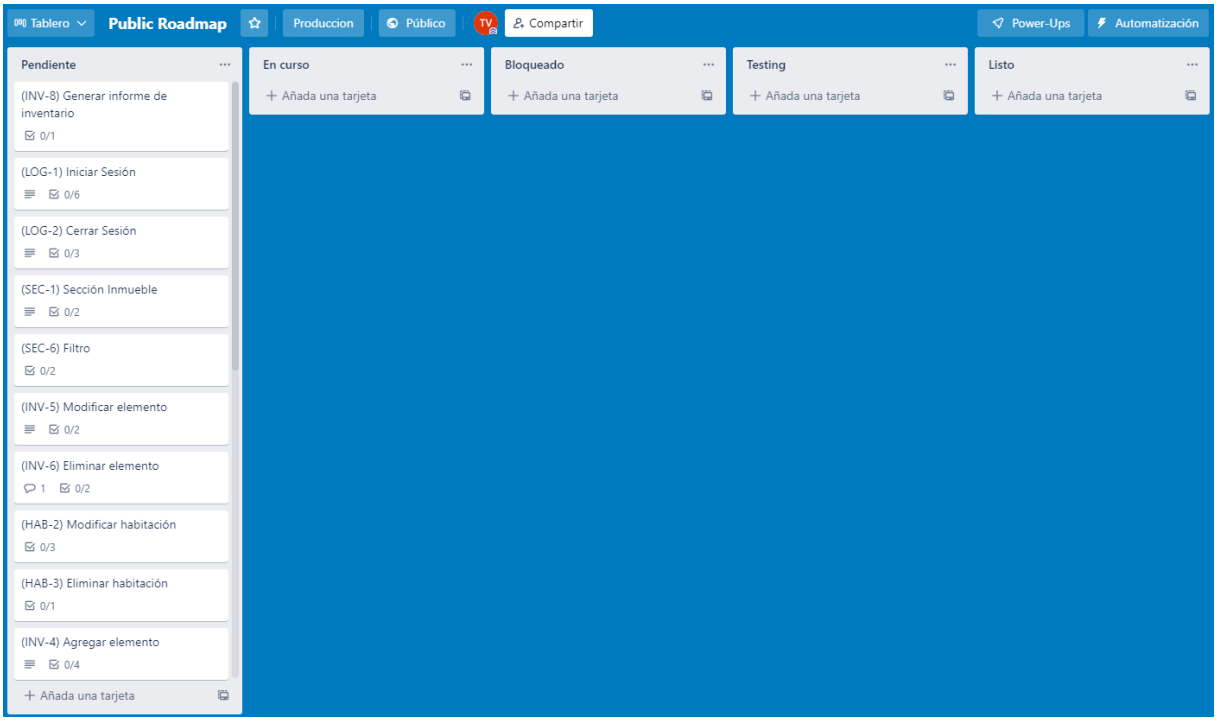


Figura 7 - Estado inicial del tablero - Captura de pantalla de autoría propia

Se agrega la siguiente imagen a modo de ejemplo de las tareas asignadas a la historia de usuario (SEC-1) Sección Inmueble (Figura 8).

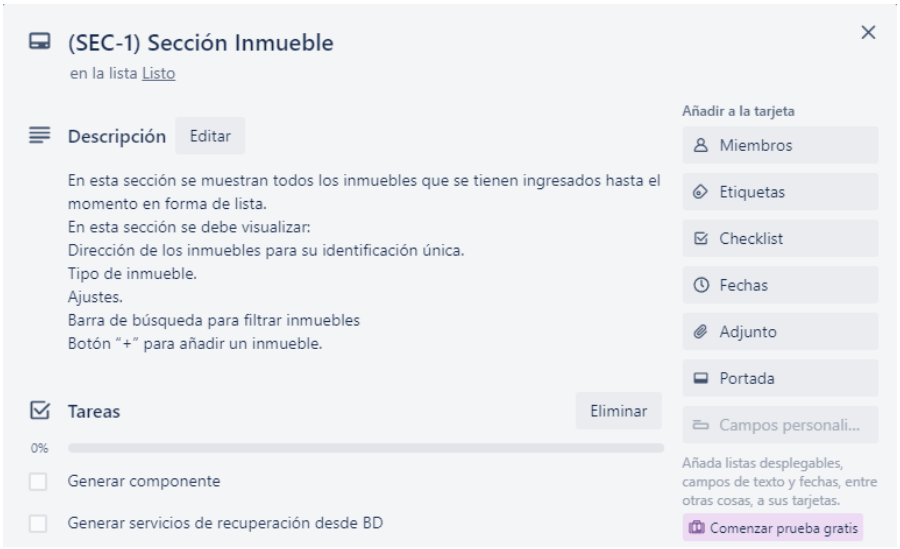


Figura 8 - Ejemplo de una historia de usuario junto con sus tareas - Captura de pantalla de autoría propia

En la siguiente imagen se puede observar el estado del tablero conforme fue avanzando el desarrollo del proyecto (Figura 9).

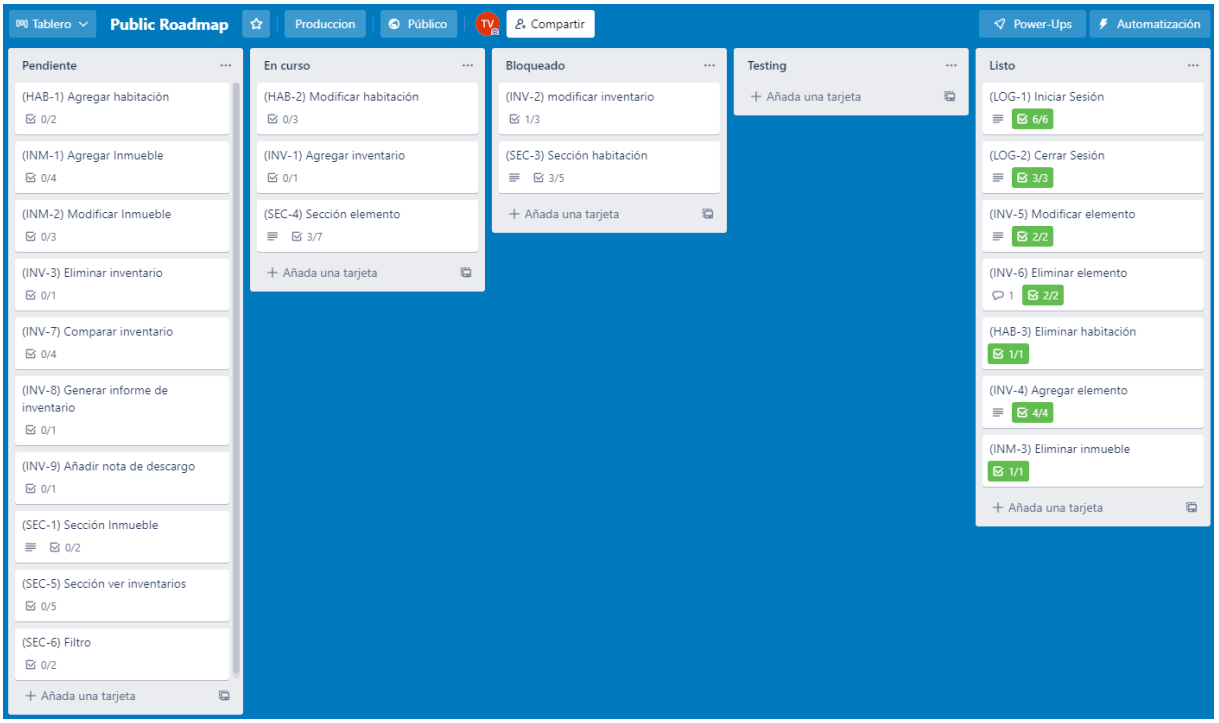


Figura 9 - Estado del tablero una vez iniciado el desarrollo - imagen de autoría propia

Por ejemplo, la historia “(INV-2) modificar inventario” requirió que la historia “(SEC-3) sección habitación” se complete y ésta a su vez, necesitó que la historia “(SEC-4) sección elemento” (el cual refiere principalmente a mostrar los elementos de habitación en pantalla) fuese desarrollada.

Para estos casos se tomó la decisión de desarrollar las tareas más generales de las historias que se encontraron bloqueadas, las cuales referían principalmente a cuestiones relacionadas a la interfaz de usuario, visibilidad, responsividad y recuperación de datos.

Posteriormente, cuando aparecía un bloqueante provocado por otra historia, se realizaba un cambio de enfoque, centrando los esfuerzos en las tareas que produjeron ese bloqueo. Una vez resueltas, se prosiguió con el desarrollo de la historia inicial.

Se concluyó que no existieron complicaciones con el uso de la metodología Kanban. La experiencia previa en relación al confeccionamiento de historias y tareas mantuvieron el orden en el desarrollo del proyecto.

4.3 Herramientas

Para el desarrollo de la aplicación, se utilizaron diferentes herramientas y tecnologías:

- *Trello*: Se utilizó en lugar de alternativas como Jira porque el tamaño de proyecto a desarrollar era pequeño, donde trello logra una mejor adaptación. Esta herramienta se adaptó perfectamente a la metodología kanban, visualizando las historias y tareas de manera sencilla las historias y sus tareas.
  RoadMap público[2].

- *VueJs*: Es un framework de Javascript con el cual se desarrolló el Front-End de la aplicación, se considera una de las 3 principales tecnologías de mercado actuales junto con Angular y React. Se seleccionó VueJs por ser el framework con la curva de aprendizaje más aplanada, y "progresivo".

- Vuex: Es una librería para gestión del estado (State Management) de aplicaciones Vue.js. Actúa en conjunto con el lenguaje VueJs. fue seleccionada por recomendación de la comunidad de desarrolladores por su aplicación del patrón Redux[3]. Esta herramienta fue fundamental en el desarrollo del producto.

- *Quasar*: Es un framework que fue aplicado por encima de vueJs para la creación de interfaces más claras. Se optó por su sistema de diseño y su enfoque hacia la responsividad de las aplicaciones

- *Firebase*: Es un sistema de gestión de bases de datos no relacional, en el cual se almacenaron los datos de la aplicación. No existe una diferencia significativa en relación a tiempos de funciones básicas CRUD[4] entre este sistema de gestión y otros debido al tamaño de la aplicación desarrollada, fue seleccionado para obtener experiencia en esta tecnología. Se adaptó correctamente a los requerimientos puesto que realiza un constante envío y recibo de datos en tiempo real y con interacción entre los usuarios.

- *IDE*: Visual Studio Code sirvió para desarrollar aplicaciones web en HTML, CSS, javascript y frameworks derivados como VueJS. Se usó esta herramienta por ser el entorno de desarrollo predeterminado del alumno.

- *GitHub*: El código del proyecto desarrollado fue almacenado de forma pública bajo licencias de código Open Source. Se utilizó por ser el repositorio de código predeterminado del alumno.
  Repositorio público[5]

- *Google Drive/ Google Docs*: Se utilizó para crear y almacenar y documentar la especificación del proyecto. Constituyen respectivamente el editor de texto y repositorio privado predeterminado del alumno.

---

[2] https://trello.com/b/8ThMRZ8d/public-roadmap
[3] Se realiza un detalle más extenso del patrón Redux en la sección 4.9.2.2
[4] CRUD: "Crear, Leer, Actualizar y Borrar" (del original en inglés: Create, Read, Update and Delete)
[5] https://github.com/ValentinisT/PF

- *NinjaMock*: Web utilizada para el armado de Spikes. Seleccionada por desconocimiento del alumno en otras tecnologías.

4.4 Aplicando la Ley de Little a este proyecto

Se consideró:

- **Tiempo de entrega**: El período entre la entrada de una petición en el sistema (petición solicitada) y la recepción de la petición.
- **Trabajo en curso (WIP)**: El número de peticiones que se están procesando, es decir las que han entrado en el sistema, pero todavía no han salido.
- **Rendimiento (Throughput)**: el número de unidades de trabajo que salen del sistema en un tiempo determinado.[6]

La ley demuestra las relaciones entre el Lead Time, el Trabajo en curso (WIP) y el Rendimiento (Throughput) (Figura 10):

$$\text{TIEMPO DE ENTREGA} = \frac{\text{TRABAJO EN CURSO}}{\text{RENDIMIENTO}}$$

Figura 10 - Fórmula que representa la ley de little - Diagrama de autoría propia

En este caso según las historias definidas en la sección 4.7, se necesitó desarrollar 24 historias de usuario.

Dada la experiencia del desarrollador y el tiempo limitado que se pudo dedicar al proyecto, se estimó un rendimiento de 1 historia de usuario por semana.

Aplicando la ley de little se estimó una duración de 24 semanas (Figura 11).

$$\text{TIEMPO DE ENTREGA} = \frac{\text{24 HISTORIAS DE USUARIO}}{\text{1 HISTORIA DE USUARIO / SEMANA}}$$

Figura 11 - Fórmula que representa la duración del proyecto a desarrollar - Diagrama de autoría propia

Según la ley de little, la duración del desarrollo del producto a entregar fue de 24 semanas. Se dedicaron 3 horas diarias al desarrollo, por lo que el tiempo en horas de elaboración del software consistió en 504 horas (Figura 12).

$$\text{TIEMPO EN HORAS DE ELABORACIÓN DEL SOFTWARE} = \frac{\text{3 HORAS}}{\text{DIA}} * \text{168 DIAS}$$

Figura 12 - Fórmula que representa las horas estimadas del desarrollo del producto - Diagrama de autoría propia

[6] La Ley de Little |Kanban. (2016, January 3). Berriprocess Agility. https://berriprocess.com/la-ley-de-little/

Además de las horas destinadas al desarrollo del software se estimaron los tiempos de ejecución y evaluación de un plan de pruebas y el desarrollo del informe del proyecto, a los cuales se les estimaron 2 y 6 semanas respectivamente (Figuras 13 y 14).

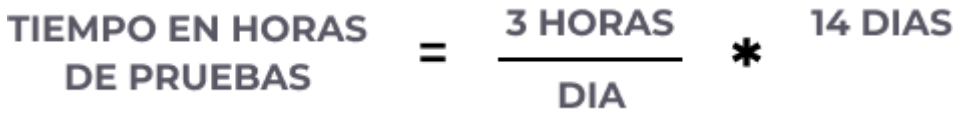


Figura 13 - Fórmula que representa las horas estimadas de la etapa de pruebas del producto - Diagrama de autoría propia

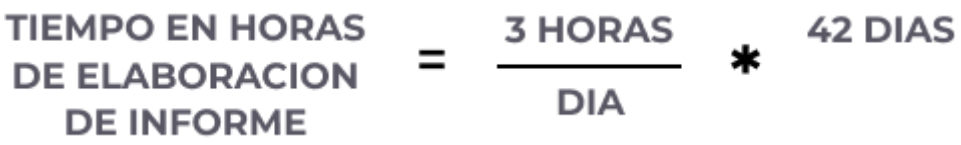


Figura 14 - Fórmula que representa las horas estimadas de la etapa de elaboración del informe del proyecto- Diagrama de autoría propia

La estimación final de esfuerzo del proyecto y todas sus actividades, incluyendo el desarrollo de producto, pruebas y elaboración del informe final resultaron en 672 horas.

4.5 Actividades realizadas

Dentro de las actividades realizadas se pueden destacar:

Comunicación: Se realizaron reuniones con el cliente con el fin de determinar problemáticas de la inmobiliaria, donde se hizo foco en el estado del proceso original de relevamiento de inventarios, pudiendo identificar problemas y por ende, oportunidades de mejora.

Planificación: En base a las oportunidades de mejora identificadas se realizó un análisis del dominio del problema, donde se usó una metodología basada en Kanban. Luego, se elicitaron los requerimientos y funcionalidades del sistema, una vez definidos se estimaron los tiempos destinados a cada tarea.

Diseño: En esta etapa se definió la arquitectura del sistema, donde se optó por una arquitectura de 3 capas, se definieron también las estructuras de datos y se diseñaron spikes de las interfaces de usuario.

Desarrollo: Se seleccionaron las herramientas para desarrollar la solución. Se realizaron todas las actividades de codificación correspondientes, incluyendo la creación de las capas de presentación, negocios y datos.

Pruebas: Se optó por realizar escenarios end-to-end para validar los requerimientos del prototipo, donde los casos de prueba de la aplicación se realizaron de manera continua, con la diferencia de que el alcance dependía de si las historias y tareas involucradas en cada caso se encontraban disponibles o no.

Posterior al end-to-end testing se utilizaron los mismos escenarios end-to-end usados previamente, en una modalidad de tipo UAT, organizando reuniones con el cliente para validar los requerimientos del prototipo.

Generación de informe final: Se estableció dentro de las 3 horas diarias de trabajo donde se generó el contenido del informe.

A continuación se presenta el diagrama de Gantt que representa el tiempo dedicado a cada actividad. (Figura 15)

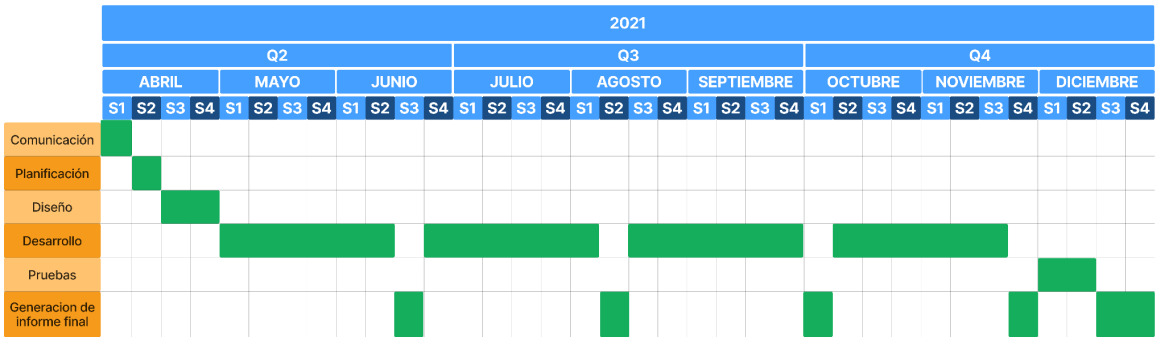


Figura 15 - Diagrama de Gantt - Cuadro de autoría propia

4.6 Recursos humanos

El presente informe contó con recursos humanos que actuaron como asesores y creadores del proyecto, siendo los mismos:

4.6.1 Asesores

Esta clasificación incluye a todas aquellas personas que con su experticia en un determinado tema, aconsejaron y guiaron al alumno cuando se requirió. Entre ellos, podemos mencionar:

- Director de proyecto. guió la evolución del proyecto.
- Docentes de la facultad. Asesoraron en cuestiones relacionadas al diseño del proyecto.

4.6.2 Creador del proyecto

1. Tomás Alfredo Valentinis. Alumno de la carrera Ingeniería en Sistemas de Información. El cual dedicó 3 horas diarias para la realización del proyecto según lo visto en la sección 4.4.

El alumno adoptó los siguientes roles:

- ➢ Analista funcional: realizó el relevamiento de los requisitos y funcionalidades del sistema, generando la especificación de requerimientos de software.
- ➢ Arquitecto de software: se encargó del diseño técnico, definiendo la arquitectura de la aplicación.
- ➢ Administrador de base de datos: se encargó de administrar la estructura de la base de datos participando en el diseño inicial de la misma, su puesta en práctica así como también su control.

- ➢ Desarrollador: Generó el código necesario para implementar el sistema que satisfaga los requerimientos, considerando la arquitectura y diseño previos.
- ➢ Tester: Ejecutó las pruebas, evaluó los resultados y confeccionó un registro con los defectos identificados.

4.7 Historias de usuario definidas originalmente

Se realizó el nombramiento de las épicas en base a recomendaciones del sitio web atlassian[7].

A cada historia de usuario se le agregó una identificación dentro de la épica.

Se decidió que el número de una historia de usuario perteneciente a una épica no representa orden ni prioridad de desarrollo, sino, patrones como es el caso de "**INM-1** agregar inmueble", "**INV-1** agregar inventario" y "**HAB-1** agregar habitación" donde las 3 historias de usuario refieren a una creación.

Finalmente para dar nombre a las historias se decidió usar el patrón "Como [perfil], [quiero] [para]".[8] Se usó esta estructura para definir historias como:

- ***"Como** usuario **quiero** una sección habitación **para** gestionar los elementos que componen una habitación."*
- ***"Como** usuario **quiero** modificar un inmueble **para** editar aspectos generales, como su nombre y su imagen."*

A continuación, se describen las historias de usuario detalladas en la sección 7.8.2 del plan de proyecto final (Figura 16).[9]

Épica	Nombre	Descripción
Login	*(LOG-1) Iniciar Sesión*	Pantalla Login que le permite al usuario ingresar un mail y contraseña para acceder al sistema
	(LOG-2) Cerrar Sesión	Funcionalidad que le permite al usuario salir del sistema
Secciones	*(SEC-1) Sección Inmueble*	Pantalla principal de la aplicación donde se muestran los inmuebles a los que el usuario puede ingresar
	(SEC-2) Sección Inventario	Pantalla donde se muestran los inventarios pertenecientes a un inmueble a los que el usuario puede entrar
	(SEC-3) Sección Habitación	Pantalla que da visibilidad de las habitaciones del inventario seleccionado y que le permite

[7] Atlassian. (n.d.). Learn how to use epics in Jira Software. Atlassian. https://www.atlassian.com/agile/tutorials/epics

[8] Atlassian. (n.d.). Historias de usuario | Ejemplos y plantilla. Atlassian. https://www.atlassian.com/es/agile/project-management/user-stories

[9] Plan Proyecto Final. (Tomás Alfredo Valentinis). Google Docs. Retrieved August 19, 2022, from https://docs.google.com/document/d/1IO-mwDxwXi7md8okYDEUCsB6byu5IA1TkD2J12bnbsU/edit

		al usuario gestionar los elementos que componen una habitación
	(SEC-4) Sección Elemento	Pantalla que muestra los elementos de una habitación y permite su modificación por parte del usuario
	(SEC-5) Sección Ver Inventarios	Pantalla que muestra el último inventario de cada inmueble, donde el usuario puede ingresar
	(SEC-6) Filtro	Funcionalidad que le permite al usuario buscar el inmueble, inventario, habitación que desee
Inmueble	*(INM-1) Agregar Inmueble*	Funcionalidad que le permite al usuario añadir un inmueble a la aplicación
	(INM-2) Modificar Inmueble	Funcionalidad que le permite al usuario editar aspectos generales del inmueble, como su nombre y su imagen
	(INM-3) Eliminar Inmueble	Funcionalidad que le permite al usuario borrar el inmueble seleccionado
Inventario	*(INV-1) Agregar Inventario*	Funcionalidad que le permite al usuario crear un nuevo inventario perteneciente a un inmueble
	(INV-2) Modificar Inventario	Funcionalidad que le permite al usuario editar los elementos de una habitación
	(INV-3)Eliminar Inventario	Funcionalidad que le permite al usuario borrar el inventario seleccionado
	(INV-4) Agregar Elemento	Funcionalidad que le permite al usuario crear elementos de una habitación
	(INV-5) Modificar Elemento	Funcionalidad que permite al usuario editar los elementos de una habitación
	(INV-6) Eliminar Elemento	Funcionalidad que permite al usuario borrar los elementos de una habitación
	(INV-7) Estado Inventario	Funcionalidad que permite al usuario ver y modificar el estado de un inventario
	(INV-8) Comparar Inventario	Funcionalidad que permite al usuario comparar dos inventarios

	(INV-9 Generar Informe Inventario	Funcionalidad que permite al usuario comparar dos inventarios
	(INV-10) Añadir Nota de Descargo	Funcionalidad que al usuario permite agregar una nota de descargo a un inmueble
Habitación	*(HAB-1) Agregar Habitacion*	Funcionalidad que permite al usuario crear una habitación
	(HAB-2) Modificar Habitación	Funcionalidad que permite al usuario editar una habitación
	(HAB-3) Eliminar Habitación	Funcionalidad que permite al usuario borrar una habitación

Figura 16 - Tabla que representa las historias de usuario definidas en el plan de proyecto final - Lista de autoría propia

4.7.1 Historias de usuario eliminadas

Conforme se avanzó en el desarrollo del proyecto, algunas de las funcionalidades de las historias de usuario se repetían en distintas secciones de la aplicación, lo que podría confundir al usuario o no generaba valor a la aplicación y por eso se optó por permitir la eliminación

4.7.2 Sección ver inventario

Originalmente, ésta era una sección donde accedía exclusivamente el administrador para visualizar los inventarios actuales de todos los inmuebles y aprobarlos.

Se descartó ésta historia debido a que la información mostrada y sus funcionalidades ya se encontraban disponibles en las secciones de inmuebles e inventarios.

4.7.3 Comparar inventarios

La comparación del inventario actual con anteriores no resultaba útil debido a que los cambios que se realizaron de un inventario a otro no se consideran extensos (a excepción de que se realice una reforma de algún tipo en el inmueble).

4.8 Spikes

Se definieron Spikes[10] a modo de ayuda visual de las historias descritas en la sección 4.7. Las mismas difieren de la versión final y solo son utilizadas a modo de ejemplo de maquetado inicial de la aplicación.

Las siguientes Spikes refieren a “Inicio Sesión” y “Sección Inmuebles” (Figura 17).

Figura 17 - Spikes de las secciones ”inicio” e “inmuebles”. NinjaMock online wireframe and mockup tool. (n.d.). NinjaMock. https://ninjamock.com/

Las siguientes Spikes refieren a “Sección Inventario” y “Sección Habitación” (Figura 18).

[10]Spikes: Bocetos o diseños en baja.

Figura 18 - Spikes de las secciones "inventario" y "habitación". NinjaMock online wireframe and mockup tool. (n.d.). NinjaMock. https://ninjamock.com/

Las siguientes Spikes refieren a “Sección Elemento” y “Sección Ver Inventario” (Figura 19).

Figura 19 - Spikes de las secciones “elemento” y “ver inventario”. NinjaMock online wireframe and mockup tool. (n.d.). NinjaMock. https://ninjamock.com/

Como se mencionó anteriormente, también se creó una versión web y de escritorio para la aplicación. Cabe aclarar, que los spikes móviles fueron adaptados de forma respectiva según resolución.

4.9 Arquitectura del sistema

El producto se desarrolló sobre una arquitectura de 3 niveles.

Capa de presentación:

- El sistema despliega la información al usuario para que el mismo agregue, modifique y elimine datos de la misma.

Capa de negocio:

- El producto posee las funciones necesarias para recibir las peticiones del usuario y usar la lógica de negocio y la comunicación con la base de datos mediante de la llamada a servicios API REST.

Capa de acceso a datos:

- La base de datos se encuentra alojada en Firestore, el cual provee la posibilidad de utilizar la base de datos Firebase y firebase storage para el almacenamiento de imágenes.

En los siguientes puntos se detalla en profundidad el funcionamiento de cada una de las capas utilizadas. Para una mejor comprensión del funcionamiento del sistema se inicia la descripción por la capa de "acceso a datos".

4.9.1 Capa de acceso a datos

4.9.1.1 Modelo de base de datos

Como se menciona en la sección 4.3, la base de datos que provee Firebase es del tipo no relacional, por lo que no existe un diagrama de tablas asociado al mismo. Sin embargo, se puede realizar una muestra de las colecciones y documentos involucrados en el desarrollo del sistema. Antes de eso, se debe realizar una vista detallada de la forma en la que Firebase maneja sus archivos en su base de datos (Figura 20).

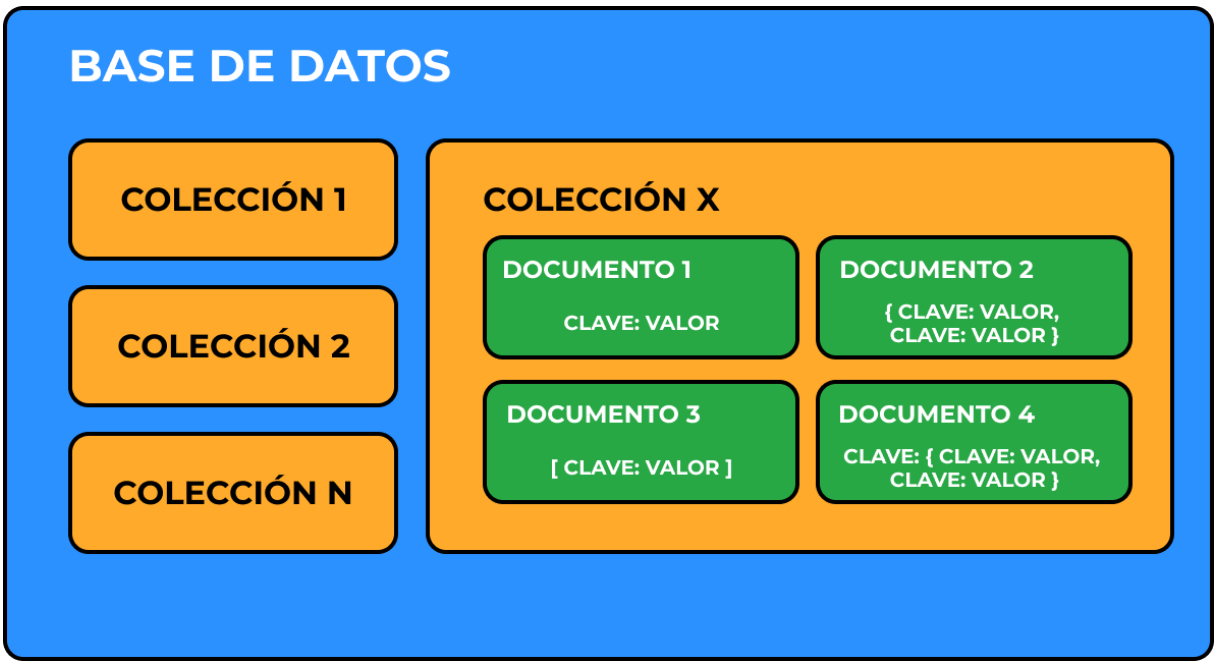


Figura 20 - Representación de una base de datos no relacional - Imagen de autoría propia

4.9.1.2 Firebase

Firebase permite almacenar datos en una base de datos no relacional. Los datos se almacenan en documentos que acumulan cualquier tipo de dato, desde los tipos primitivos como number, string, hasta tipos de datos más complejos como listas.

A su vez, los documentos son almacenados en colecciones y no es requerido que dispongan de los mismos atributos.

Un ejemplo básico que proporciona la página de asistencia de Firebase[11]:

"[...] podrías tener una colección llamada usuarios con los distintos usuarios de tu app, en la que haya un documento que represente a cada uno".

[11] Documentation | Firebase. (2019). Firebase. https://firebase.google.com/docs

En el transcurso del desarrollo del sistema se han encontrado problemas con algunas de las formas de trabajar de Firebase. Como se establece en la documentación, se cita:

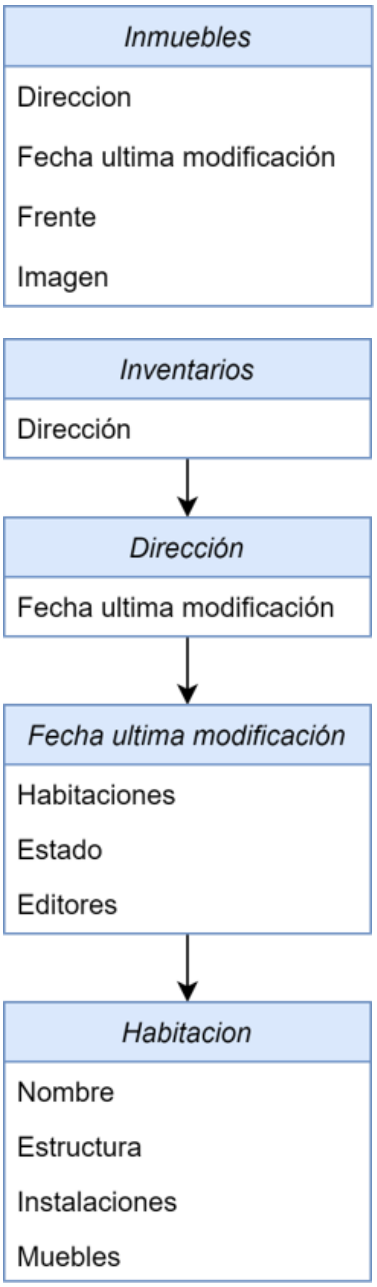
"Los nombres de documentos dentro de una colección son únicos. Puedes proporcionar tus propias claves, como los ID de usuario, o puedes dejar que Cloud Firestore cree ID aleatorios de forma automática.

No es necesario 'crear' ni 'borrar' las colecciones; Cuando se crea el primer documento de una colección, esta pasa a existir. Si borras todos los documentos de una colección, esta deja de existir".[12]

[12] Documentation | Firebase. (2019). Firebase. https://firebase.google.com/docs

4.9.1.3 Colecciones y documentos

Teniendo en cuenta lo anteriormente dicho, se realizó una descripción de las colecciones y documentos creados (Figura 21).[13]



Inmuebles: Representa a la propiedad, colocando su nombre y detallando su frente, una imágen que se incluye en el reporte generado y una fecha de última modificación para habilitar el filtro por inmuebles de más nuevo al más viejo y viceversa.

Inventarios: La colección de inventarios posee datos conformados por una dirección que actúa como colección.

Dirección: Colección que alberga otra colección de nombre “fecha”, la misma se identifica como “fecha de última modificación”, esto contribuye al orden de los inventarios y el conocimiento de cuál es el inventario que se está editando.

Fecha última modificación: Esta colección almacena el estado del inventario que indica si se encuentra en curso, pendiente de aprobación o aprobado. También contiene los editores que participaron del inventario y una colección de habitaciones.

Habitación: Se definió que los nombres de una habitación son únicos, para lo cual, esta colección se autodenomina "habitación". Además dispone de una lista de estructuras, instalaciones y muebles de una habitación, es decir, todos sus elementos con sus propiedades y valores.

[13] Nota del alumno: Las flechas no indican una “relación” sino que la colección desde donde parte la flecha incluye la colección/documento/datos donde finaliza la flecha

Habitaciones prefabricadas: Esta colección detalla las habitaciones estándar que un inmueble puede tener, las mismas son creadas por el usuario en la aplicación.

Habitación: Esta colección dispone de los mismos documentos que la que se encuentra dentro de la colección de inventarios.
Importante: difiere porque se desea evitar que se establezca una relación dentro de la base de datos

Lista instalaciones: Colección que incluye una lista de los elementos del tipo "instalación" que puede tener una habitación. Contiene el nombre del elemento propiamente dicho y una lista de propiedades.

Lista muebles: Colección que incluye una lista de los elementos del tipo "mueble" que puede tener una habitación. Contiene el nombre del elemento propiamente dicho y una lista de propiedades.

Usuarios: Esta colección contiene los datos de los usuarios del sistema, los cuales consisten en un nombre que se utiliza para marcarlo como autor en los inventarios, un email para el acceso al sistema y una propiedad marcada como "admin" el cual habilita al usuario a aprobar los inventarios que se encuentren pendientes de aprobación.

Figura 21 - Diagrama de colecciones y documentos de la base de datos - Diagrama de autoría propia

4.9.1.4 Problemas encontrados

En esta sección se comentan los problemas que se tuvieron con el uso de la base de datos que proporciona Firebase a lo largo del desarrollo del sistema

Se deduce que una de las principales causas de las dificultades surgieron debido a la inexperiencia en el diseño y uso de una base de datos no relacional.

- **Implementación en tiempo real**

  Muchas de las funcionalidades del sistema requirieron que la base de datos escuche los cambios que se realizaron sobre ciertos datos y que los mismos fueran enviados a otros usuarios.

  La problemática asociada a esta funcionalidad consiste en que los datos debían guardarse en los estados de cada aplicación, además de en la base de datos. Si bien Firebase proporciona las APIs para comunicarse con el servidor e indicarle que escuche cuando se agrega, edita y/o elimina un dato, la misma no es del todo claro en el alcance de su funcionamiento, por lo que hubo casos donde se tuvo que declarar variables o realizar chequeos de consistencia para corroborar que los mismos fueron modificados con éxito tanto en la aplicación como en el servidor de bases de datos.

- **Colecciones y documentos inalterables**

  Como se mencionó en la sección 4.9.1.1, Firebase indica que la forma de actuar de su base de datos se basa en colecciones, documentos y datos. También aclara que las claves de estos son inalterables, lo cual desencadena un problema puesto que en algunos casos es necesario modificar el nombre de ciertas colecciones para que sirvan al propósito de la aplicación. Cabe destacar que esta situación no ocurre la mayoría del tiempo, son contados los casos donde se aplica. La aplicación no fue destinada a un público general, por lo que no existió posibilidad de que se realice una escalabilidad desmesurada en términos de usuarios, posibilitó desarrollar una solución poco eficiente, que consistió en eliminar la colección y agregar otra que ocupe su lugar con su nuevo nombre.

- **Atributos null y undefined**

  En la sección 4.9.2.1 se aclaró que al borrar todos los documentos de una colección, ésta deja de existir. Algo similar ocurre con los datos que se encuentran dentro de estos documentos, los cuales, si se encuentran marcados como “null” o “undefined”, son eliminados del documento. Normalmente, ésto no sería un problema puesto que es probable que se conozca el dato que se envía a la base de datos, el problema surge cuando se desconoce qué dato se enviará, ésto ocurrió porque se le dió la posibilidad al usuario de crear los elementos (muebles e instalaciones), fue imposible definir todos los componentes que se pueden encontrar dentro de una casa, la solución a este problema consistió en asignar un string vacío (“”) a los datos que se encuentren como null o undefined antes de cargarlos en la base de datos

4.9.2 Capa de negocio

En esta sección se detalla cómo se llevan a cabo las funcionalidades de la aplicación, estableciendo la comunicación entre la capa de presentación y la capa de acceso a datos.

Como se mencionó en la sección 4.1, se usó de VueJs para el desarrollo de los componentes del sistema.

En el apartado que sigue, se hace foco en las funcionalidades de los componentes y en la próxima capa, como se disparan estas acciones.

4.9.2.1 VueJs

VueJs es un framework de javascript que permite crear componentes, los cuales son el esqueleto del sistema.

Por lo general, existe un componente padre de toda la aplicación y sub-componentes o componentes hijos que poseen su propio estado, es decir, las variables que maneja internamente.

Los componentes padres e hijos se comunican por medio de props (padre a hijo) y eventos (hijos a padre). Si bien esto proporciona una estructura bien definida y un alcance de las funciones de cada componente en un inicio, conforme el número de componentes aumenta, el disparo de props y eventos empieza a resultar más complejo. Se produce una duplicidad de variables en los componentes y suele ocurrir que componentes pequeños o principales aumenten su tamaño considerablemente solo por declarar las funciones de prop y event.

En algunos casos no es posible lograr una comunicación entre los componentes que se desea, como por ejemplo la de dos componentes hermanos que deben comunicarse a través del padre, complejizando aún más la situación (Figura 22).

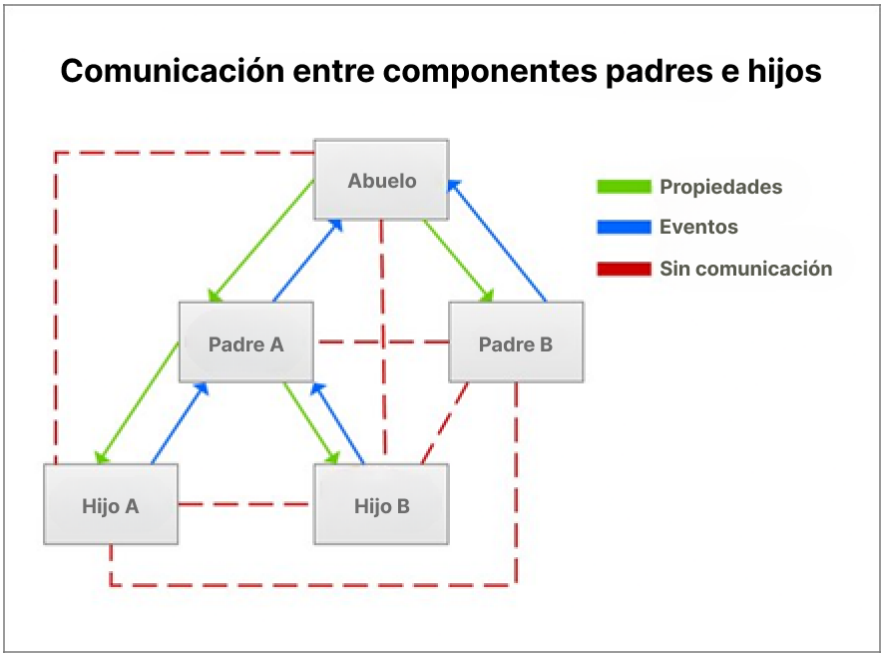


Figura 22 - Patrones de diseño para la comunicación entre componentes Vue.js. (n.d.). Code Envato Tuts plus. https://code.tutsplus.com/es/tutorials/design-patterns-for-communication-between-vuejs-component--cms-32354

4.9.2.2 Vuex

Vuex es una librería diseñada por los creadores de VueJs para manejo de estados, los cuales se inspiraron en el patrón Redux.

La función de Vuex es la de mantener un estado o store global y compartido para todos los componentes, de esta manera se solucionan los problemas anteriores, puesto que ahora se accede a la lectura y escritura del estado de la aplicación desde cualquier componente, de manera que no es necesaria la comunicación entre padres e hijos directamente, sino que se hace a través de la store.

Como ventaja, Vuex es reactivo, esto quiere decir que si un componente modifica un elemento del estado global, este se modificará en todos los componentes que se encuentren usando el elemento.

Para explicar el manejo de estados en Vuex, se hace uso de la figura 23. Se parte desde un componente de Vue, el cual dispara una acción, las cuales son asincrónicas, esta realiza una llamada a una API para leer y/o escribir determinado dato. Posteriormente realiza un commit, es decir, llama a una mutation, las cuales son sincrónicas. Las mutations o mutaciones, son las que modifican el estado propiamente dicho según la lógica que se le indique. Al finalizar, el estado es enviado al componente con sus nuevos datos.

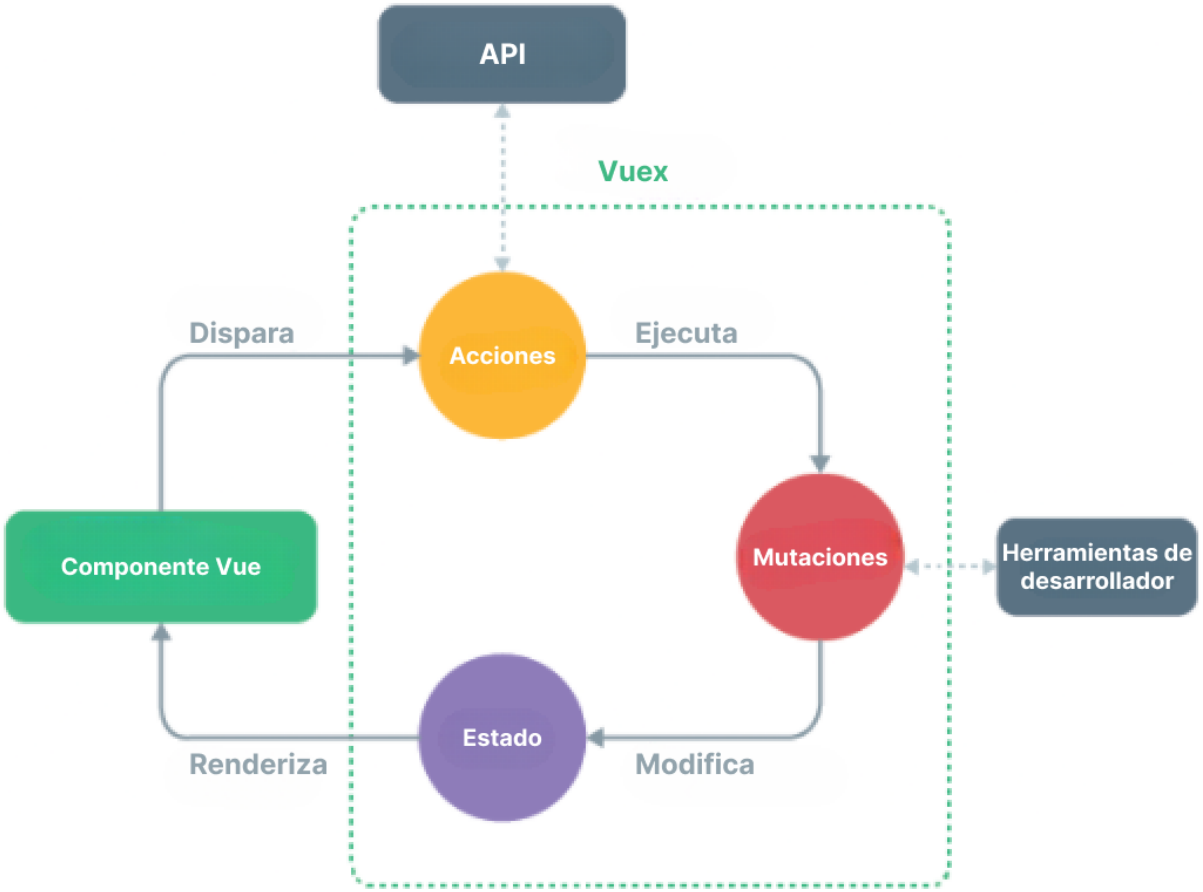


Figura 23 - Representación gráfica del funcionamiento interno de vuex. What is Vuex? | Vuex. (n.d.). Vuex.vuejs.org. https://vuex.vuejs.org/#what-is-a-state-management-pattern

4.9.2.3 Aplicando VueJs y Vuex

Se mencionó que VueJs define la estructura de un proyecto en base a sus componentes. En las historias de usuario se habían definido las secciones del sistema por las cuales los usuarios interactúan con la misma.

Se tomó la decisión de que cada sección constituya un componente de VueJs. Se decidió realizar la modularización de componentes para su reutilización y simplificación, por lo que, conforme un componente se complejizaba, se realizó el desarrollo de otro componente para encapsular ese set de funcionalidades.

Para aplicar el manejo de estados con Vuex se creó una store, la cual contiene los estados que son utilizados en uno o más componentes o aquellos estados que es necesario que sean reactivos, como por ejemplo la lista de inmuebles.

La conexión con las APIs que provee Firebase se realiza mediante “actions”, a partir de estas es que se puede realizar la lectura y escritura de sus elementos. Las APIs proporcionadas por Firebase ayudan a lograr que las interacciones de la aplicación ocurran en tiempo real y sin errores de concurrencia, estas funcionalidades consisten en que una acción escucha los cambios que ocurren en determinada colección en la base de

datos, por lo que, si otro usuario modifica algún documento, Firebase enviará los datos actualizados y realizará un commit hacia un “mutator” encargado de gestionar los datos y actualizar el respectivo estado.

4.9.3 Capa de presentación

En esta sección se detallan las interfaces gráficas que la aplicación presenta al usuario, comunicando información para llevar a cabo sus funciones.

4.9.3.1 Sección Inmuebles

En ésta pantalla (Figuras 24 y 25) se presentan los inmuebles que se encuentran creados actualmente en el sistema, éstos se muestran en formato lista o con sus respectivas imágenes.

El usuario puede filtrar y ordenar alfabéticamente o por antigüedad, también puede crear, modificar y eliminar inmuebles, éstas acciones se visualizan en el siguiente video: https://youtu.be/5jU-mihSE4U.

Figura 24 - Sección inmuebles - Formato móvil - Captura de autoría propia

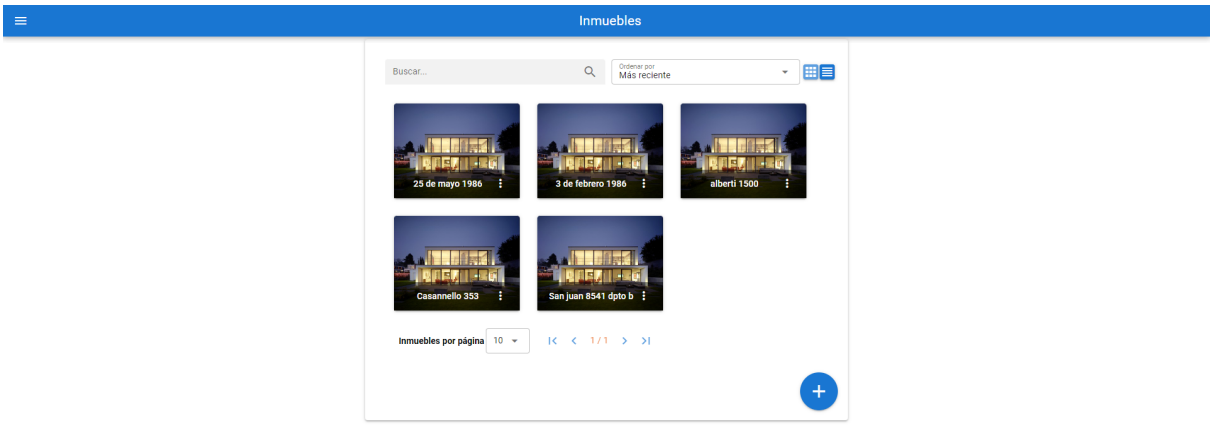


Figura 25 - Sección inmuebles - Formato Web - Captura de autoría propia

4.9.3.2 Sección Inventarios

Esta interfaz (Figuras 26 y 27) muestra las habitaciones de un inventario. El usuario puede crear, modificar, eliminar habitaciones y generar el reporte de un inventario.

Este flujo puede visualizarse en el siguiente video: https://youtu.be/I0YxpY_eXDs.

El usuario también puede modificar el estado de un inventario, visualizar versiones anteriores y crear uno nuevo a partir del anterior, esto puede verse en el siguiente video: https://youtu.be/WKz6ghz_o7Q.

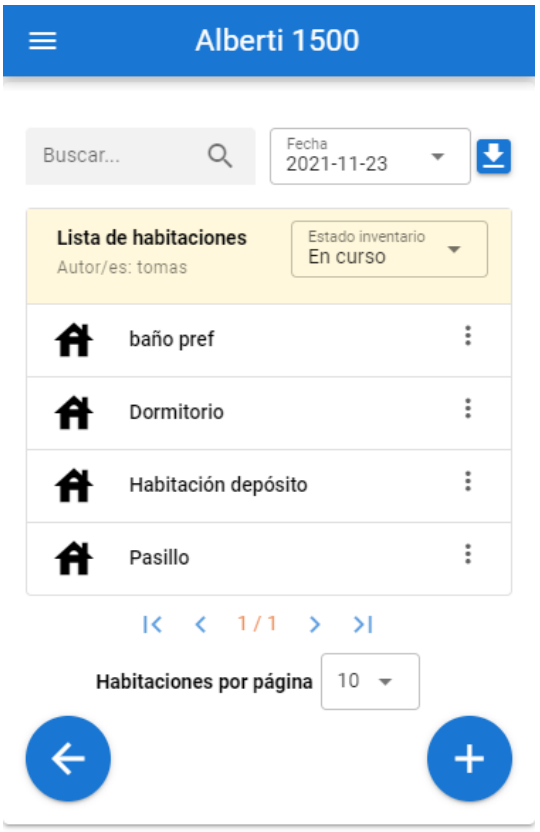


Figura 26 - Sección Inventarios- Formato móvil - Captura de autoría propia

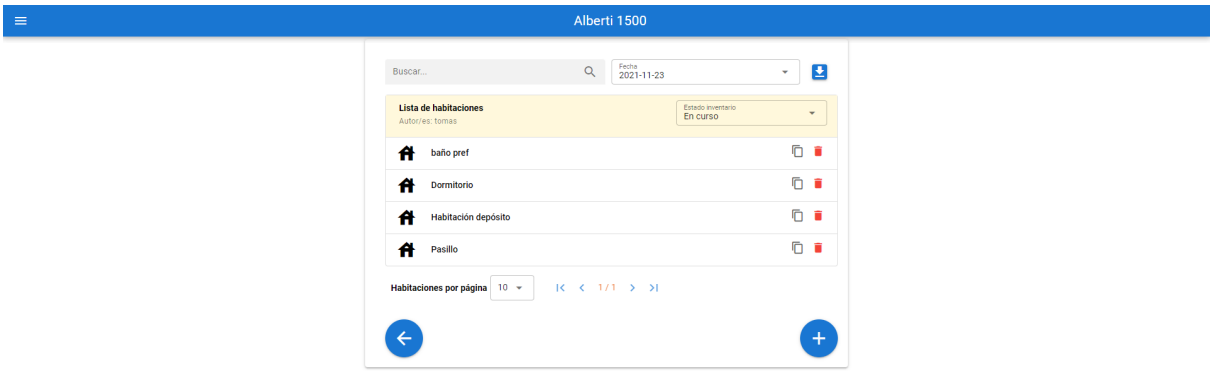


Figura 27 - Sección Inventarios- Formato web- Captura de autoría propia

4.10 Testing

Se optó por realizar escenarios end-to-end para validar los requerimientos del prototipo donde los casos de prueba de la aplicación se realizaron de manera continua, con la diferencia de que el alcance dependía de si las historias y tareas involucradas en cada caso se encontraban disponibles o no.

En la Figura 28 se observa un diagrama que representa “casos de pruebas”, “acciones” que son los pasos a seguir para llevar a cabo este caso y “decisiones” que son variantes que pueden ocurrir en este caso.

El testing se realizó recorriendo el gráfico, realizando las “acciones” del usuario y sus variantes.

Por ejemplo, se prueba el inicio de sesión con el usuario porque la historia “(LOG-1) Iniciar sesión” se encuentra desarrollada pero, sí las funcionalidades de la historia “(SEC-1) Sección inmueble” no se pueden llevar a cabo, el testing llega a su fin en este momento.

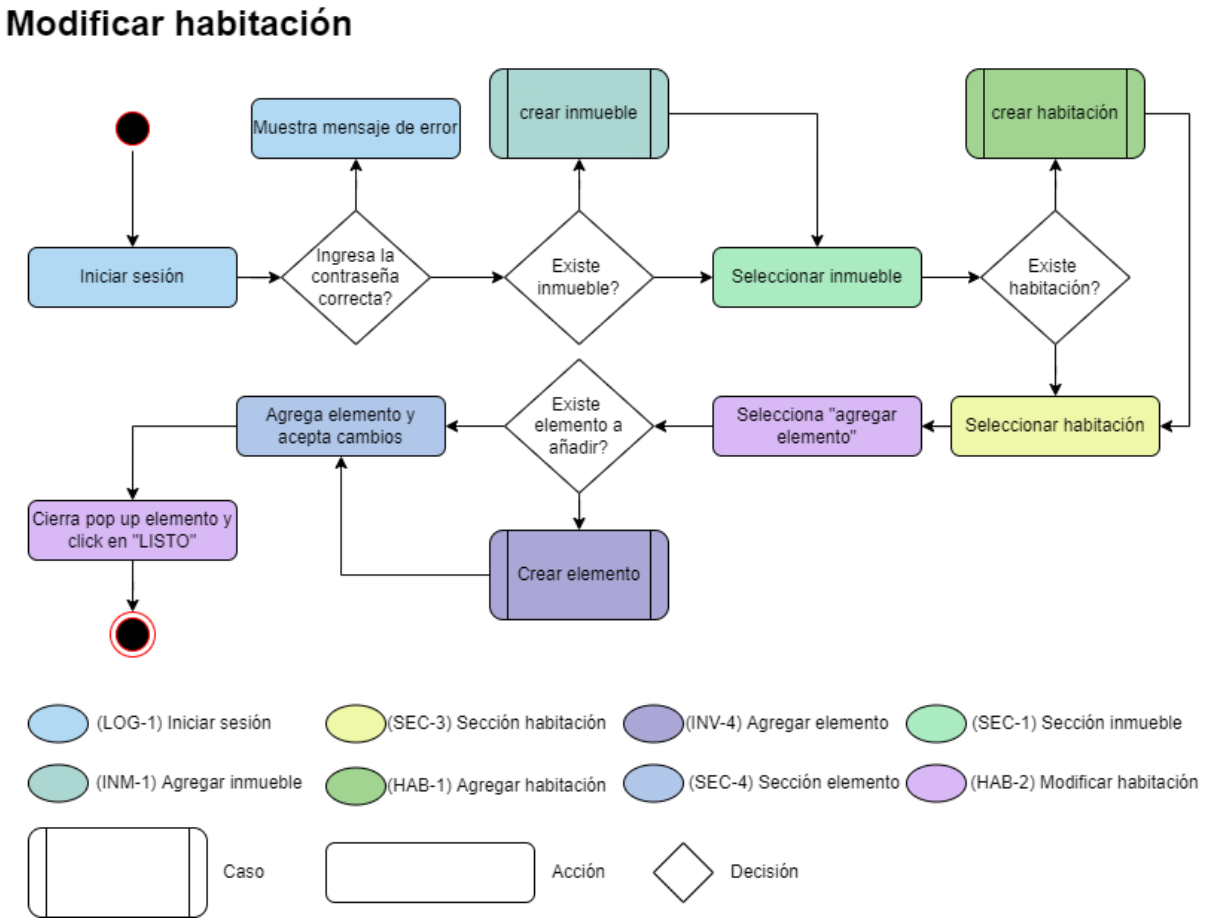


Figura 28 - Diagrama de casos de uso - Diagrama de autoría propia

4.10.1 Evolución del caso de prueba

El caso de prueba base, consiste en iniciar la sesión y se ingresa a la "sección inmueble" que constituye la página principal de la aplicación. El caso finaliza porque aún no existe ninguna funcionalidad relacionada a la gestión de inmuebles (Figura 29).

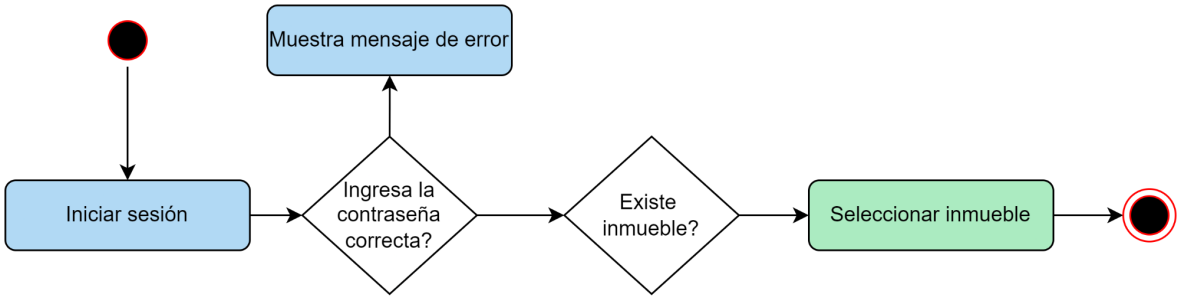


Figura 29 - Tercera iteración - Diagrama de casos de uso - Diagrama de autoría propia

En esta iteración, el usuario puede crear inmuebles. Se realizó el testing sobre esta función, siempre partiendo desde el inicio de sesión de manera que se realice un control de que la incorporación de esta nueva funcionalidad no cause errores en las anteriores (Figura 30).

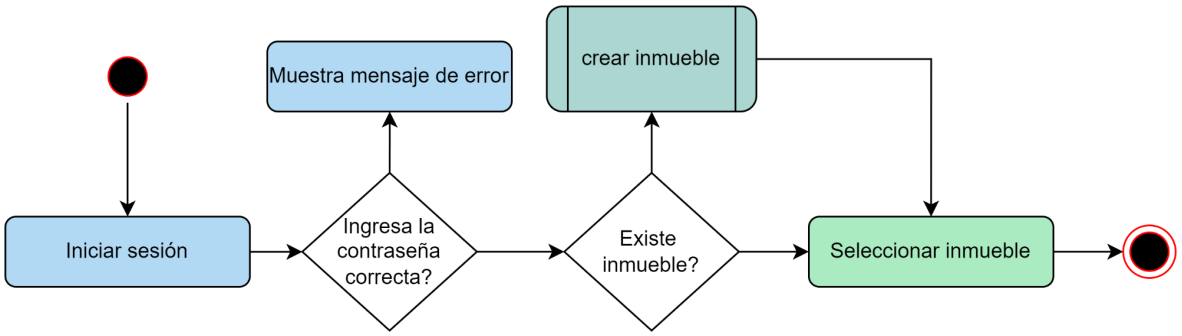


Figura 30 - Cuarta iteración - Diagrama de casos de uso - Diagrama de autoría propia

En esta iteración, el usuario puede visualizar y crear habitaciones (Figura 31).

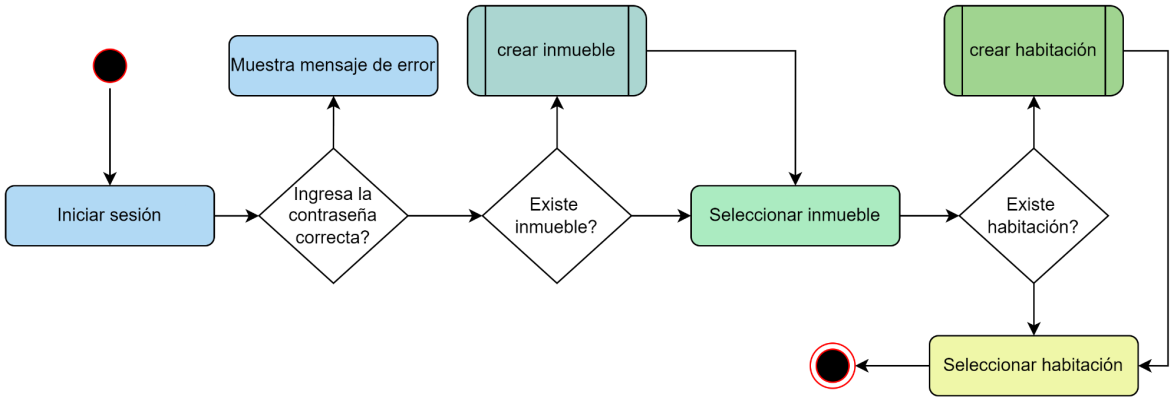


Figura 31 - Sexta iteración - Diagrama de casos de uso - Diagrama de autoría propia

En esta iteración, se agregó la funcionalidad de editar la habitación y añadir elementos, también se incorporaron las funcionalidades para aceptar los cambios y guardarlas en la base de datos (Figura 32).

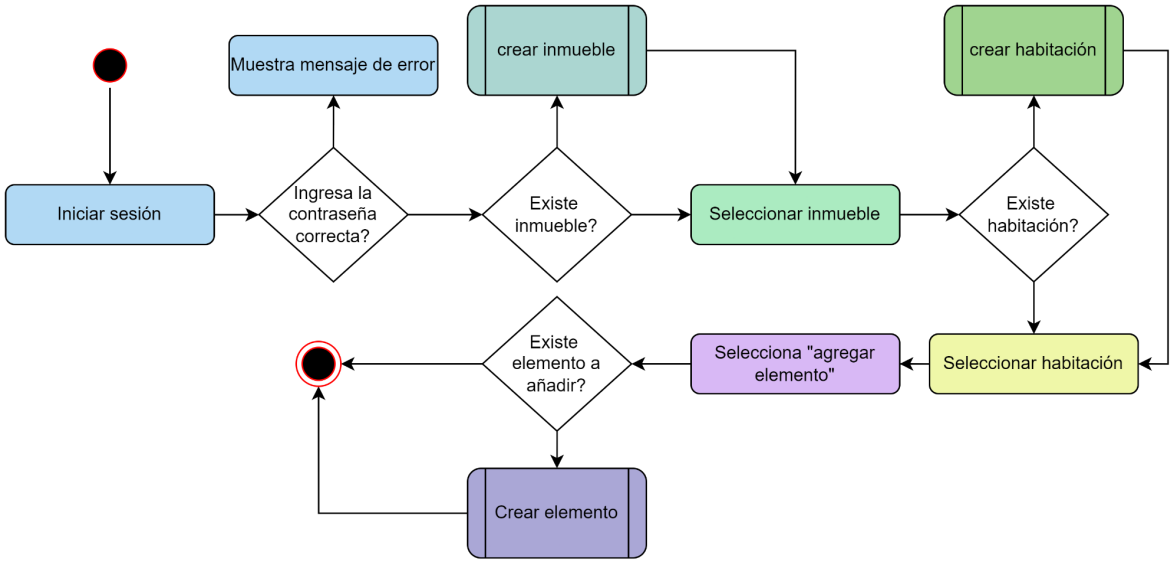


Figura 32 - Iteración Final - Diagrama de casos de uso - Diagrama de autoría propia

4.10.2 Problemas

El testing contribuyó a que el desarrollo del sistema sea más ordenado y se utilizó como una herramienta extra de la metodología de desarrollo, ya que el flujo del mismo contribuía a designar el orden de las tareas de las historias de usuario que debían desarrollarse.

Además de optimizar el tiempo de desarrollo al identificar bugs en edades tempranas del software donde el rastreo y corrección eran más sencillos.

Sin embargo, a pesar de realizar un testing continuo durante el desarrollo del sistema, el número de bugs descubiertos en cada iteración fue en aumento, principalmente relacionados con las variables de estado y su interacción con la base de datos de firebase.

Se concluye que los bugs ocurrieron por la falta de experiencia en el uso de Vuex y el uso de una base de datos no relacional, sumado a la aplicación de los datos en tiempo real.

4.10.3 User Acceptance Testing (UAT)

Posterior al end-to-end testing y, debido a que la aplicación fue desarrollada en su totalidad por una única persona, se utilizaron los mismos escenarios end-to-end usados previamente, pero en una modalidad de tipo UAT, organizando reuniones con el cliente para validar los requerimientos del prototipo. Debido a que el cliente no se encontraba aún familiarizado con el producto, las sesiones de testing fueron guiadas por el desarrollador[14].

[14] Nota del alumno: Según los principios de testing, no es aconsejable que la persona que ejecuta las pruebas sea la misma que realizó el código, pero en esta oportunidad dicha práctica no pudo evitarse.

5. Conclusiones

En esta sección se detallan las conclusiones a las que se llegó luego del desarrollo del proyecto.

5.1 Introducción

Este proyecto surge en base a la experiencia en primera persona de haber ejecutado un proceso de relevamiento de inventarios tradicional, en conjunto con una de las empleadas de la inmobiliaria. A partir de esta experiencia y en base a las necesidades de la empresa se comenzó a pensar en formas de optimización y futuras soluciones para este proceso.

5.2 Cliente

El prototipo desarrollado aportó una nueva visión de mejora de los procesos de la inmobiliaria. A través de la ejecución del plan de pruebas el cliente reconoció que los procesos tradicionales pueden ser mejorados con el fin de aliviar la carga de trabajo de los empleados.

5.3 Alumno

El desarrollo de este proyecto ha representado un desafío personal y profesional. Gracias a la implementación de esta aplicación multiplataforma, se alcanzaron importantes experiencias, conocimientos y manejo de herramientas y tecnologías desconocidas.

5.4 Proyección

Se simuló la creación de un inventario junto con el cliente, usando inventarios antiguos.

El tiempo de relevamiento por inventario en la aplicación se aproximó a los 120 minutos, mientras que el inventario original fue realizado en 90 minutos, en este último caso se tiene que añadir el tiempo que se tarda en dar formato al documento generado y anexar las respectivas fotos, lo cual deja el tiempo de relevamiento original entre 150 y 180 minutos.

Estos tiempos eran esperables independientemente de la aplicación que se desarrolle, debido a 3 razones:

- El tiempo de adaptación del usuario al sistema.
- La nueva forma de relevamiento guiado por el sistema en lugar de usar de un editor de texto.
- La creación y modificación de habitaciones prefabricadas y elementos de una habitación, los cuales en un inicio extienden el tiempo de relevamiento pero en un futuro contribuirán a reducirlo.

Se espera que el tiempo de relevamiento disminuya conforme los usuarios se adapten a la aplicación, con este fin se recomendó elaborar un plan de capacitación para los empleados.

Se estima que luego de ésta formación el tiempo de relevamiento sea de entre 60 y 90 minutos.

Cabe destacar que si se realizan 2 inventarios a la vez, la diferencia temporal entre el proceso de relevamiento tradicional versus el relevado mediante la aplicación disminuiría entre 90 y 180 minutos.

5.5 Trabajo a futuro

De cara al futuro y en base a lo conversado con la inmobiliaria, se pretende alcanzar las siguientes mejoras:

- Realizar mejoras en la interfaz de usuario de la aplicación para agilizar el relevamiento como ser adaptación a principios accesibles y optimización de la interfaz de usuario.
- Modificaciones en los informes finales para reducir su tamaño y facilitar su lectura.
- Incorporación de nuevas funcionalidades para disponer de notas de audio en el relevamiento de las propiedades.
- Alineación estética con sistema de branding e identidad de la institución.

5.6 Oportunidades de mejora de la aplicación

Se aclara que la aplicación posee limitaciones, que pueden ser resueltas en una versión final completa. Por ejemplo:

- La interfaz: Este prototipo es una versión básica sin requerimientos o particularidades específicas que puede mejorarse mediante el rediseño de la interfaz de usuario, teniendo en cuenta la retroalimentación provista por el cliente.
- Los reportes: Esta versión cuenta con reportes de inmuebles extensos debido a la distribución de imágenes embebidas. La mejora puede darse a través de una reestructuración de la arquitectura de la información.
- Multimedia: El prototipo desarrollado contiene limitaciones técnicas que no permite realizar videos y/o notas de autos. Una futura versión podría contemplar la incorporación de estos formatos de archivos.

6. Referencias bibliográficas[15]

1. El Modelo Cynefin. (2021, May 26). Thinking with You. https://thinkingwithyou.com/2021/05/modelo-cynefin-organizaciones/
2. Documentation | Firebase. (2019). Firebase. https://firebase.google.com/docs
3. Patrones de diseño para la comunicación entre componentes Vue.js. (n.d.). Code Envato Tuts plus. Retrieved August 18, 2022, from https://code.tutsplus.com/es/tutorials/design-patterns-for-communication-between-vuejs-component--cms-32354
4. Atlassian. (n.d.). Learn how to use epics in Jira Software. Atlassian. https://www.atlassian.com/agile/tutorials/epics
5. What is Vuex? | Vuex. (n.d.). Vuex.vuejs.org. https://vuex.vuejs.org/#what-is-a-state-management-pattern
6. Kanban. (n.d.). IONOS Digitalguide. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-kanban/
7. Cómo Gestionar Tus Proyectos con Kanban Personal. (n.d.). Facilethings.com. Retrieved August 18, 2022, from https://facilethings.com/blog/es/personal-kanban
8. Cómo elegir el enfoque de gestión de proyectos adecuado. (n.d.). GetApp. Retrieved August 18, 2022, from https://www.getapp.es/blog/673/agile-kanban-scrum-cascada-gestion-de-proyectos-adecuadodesarrollo
9. Ley de Little y la Gestión del trabajo - Kanban | Saraclip. (n.d.). Retrieved August 18, 2022, from https://www.saraclip.com/ley-de-little/
10. La Ley de Little |Kanban. (2016, January 3). Berriprocess Agility. https://berriprocess.com/la-ley-de-little/
11. Plan Proyecto Final. (Tomás Alfredo Valentinis). Google Docs. Retrieved August 19, 2022, from https://docs.google.com/document/d/1IO-mwDxwXi7md8okYDEUCsB6byu5lA1TkD2J12bnbsU/edit
12. Gracia, L. (2019, May 9). ¿Qué es Vuex? Un Poco de Java. https://unpocodejava.com/2019/05/09/que-es-vuex/

[15] Formato de citado según norma APA 7ma edición.

7. Anexos

7.1 Terminología

- Propietario: Persona física o jurídica que cuenta con los derechos de propiedad sobre un bien.
- Inquilino: Persona que ha alquilado un inmueble o parte del mismo.
- Usuario: Individuo que hace uso de la aplicación. Se incluyen 2 perfiles, uno con permisos de administrador y otro de empleado sin éstos permisos.
- Inmueble: Bien perteneciente a uno o más individuos que no puede ser trasladado física o jurídicamente.
- Inventario: Documento donde se registran todos los bienes tangibles de, en este caso, un inmueble.
- Habitación: Lugar destinado a vivienda, se componen de:
    - Estructura: Son las paredes, techo y piso de la habitación.
    - Instalaciones: Son los elementos que una habitación tiene incorporados en alguna de sus estructuras (ejemplo: aire acondicionado).
    - Muebles: Bienes pertenecientes al propietario, los cuales pueden ser trasladados.
- Contrato de alquiler: Documento legal en el que existen dos partes, donde una cede temporalmente el uso de uno o más inmuebles a cambio del pago de una renta.
- Relevamiento: Proceso formal en el que se reúnen y analizan datos.