UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EXTENSIÓN DE UNA HERRAMIENTA DE SCOPING DE PROYECTOS DE SOFTWARE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

ARIEL ERNESTO SUIL ARAVENA

PROFESOR GUÍA:
SERGIO OCHOA DELORENZI

PROFESOR CO-GUÍA:
DANIEL PEROVICH GEROSA

MIEMBROS DE LA COMISIÓN:
PABLO GONZÁLEZ JURE
ALEJANDRO HEVIA ANGULO

SANTIAGO DE CHILE
2022

Resumen

En la presente memoria se describe el desarrollo realizado sobre la herramienta conocida como Tablero Digital, herramienta que tiene como objetivo permitir a sus usuarios la definición del alcance de un producto tecnológico. La extensión realizada sobre el sistema, busca ofrecer a los usuarios del mismo, la habilidad de manejar el estado en que se encuentra su producto luego de la definición de su alcance.

Previo a la extensión realizada en esta memoria, los usuarios del sistema Tablero Digital debían duplicar de forma manual la información existente en el sistema, para ingresarla en otro que les permitiera manejar el estado de desarrollo de su producto. Esto se traducía en un consumo adicional de tiempo para el usuario, y por ende, una disminución en el tiempo a utilizar en el desarrollo de su producto.

Para mejorar esta situación, se introducen al sistema de Tablero Digital las herramientas necesarias para el manejo del estado de desarrollo de su producto. Específicamente, se permite a los usuarios definir hitos de desarrollo sobre los cuales pueden manejar el estado de las diversas características del producto siendo desarrollado. Esta extensión fue evaluada con usuarios expertos del sistema previo, para asegurar su utilidad y usabilidad, obteniendo resultados positivos en ambos casos.

Al contar con esta funcionalidad, los desarrolladores que utilicen el sistema pueden mantener toda su información centralizada en un solo lugar, quitando la necesidad de buscar sistemas externos; con la ventaja adicional de permitir al cliente mantenerse más cerca del desarrollo al presentar el avance del mismo en una interfaz ya conocida.

A mi familia

Agradecimientos

Gracias a mi abuela, mi hermana, mis madres, y mi prometida, por su apoyo incondicional. Y gracias a todas las personas que me ayudaron en el desarrollo de esta memoria, especialmente a mis profesores guía Sergio y Daniel.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Proceso de scoping

En el camino a recorrer al momento de desarrollar un producto tecnológico, la coordinación es fundamental para asegurar su correcta elaboración. En general, los productos son creados por un equipo llamado *desarrolladores*, para una entidad que llamamos *cliente*. Los desarrolladores deben entender qué es lo que el cliente espera del producto, y el cliente debe entender cuáles son las limitaciones y costos de la producción del mismo.

Para poder llegar a un mutuo entendimiento entre desarrolladores y clientes, nace el proceso llamado scoping [19], es decir, el análisis del alcance que tendrá un producto en términos de las funcionalidades a implementar. Este análisis es típicamente realizado de manera presencial, añadiendo tarjetas adhesivas o *post-its* a un tablero físico, donde cada post-it representa un elemento a considerar en el desarrollo del producto.

Dadas las limitaciones asociadas a la presencialidad, nace la necesidad de digitalizar este proceso para permitir su realización de manera virtual, en tiempo real, y con fácil acceso futuro a lo decidido durante la sesión. Es en este contexto que el ex-alumno Javier Gómez Peña [10] desarrolla la primera versión de la herramienta de scoping conocida actualmente como Tablero Digital[1].

1.2. Herramienta de scoping

El sistema Tablero Digital permite a los usuarios identificarse como desarrolladores o clientes, para trabajar de manera conjunta, y en tiempo real, en la definición de funcionalidades y elementos clave del producto a desarrollar. Un usuario de Tablero Digital puede ingresar al sistema y crear un tablero, ya sea definiendo las secciones y geometrías que lo componen o utilizando un template guardado, para luego poder añadir nuevos post-its o

[1]Tablero Digital es una iniciativa abierta de académicos del Departamento de Ciencias de la Computación de la Universidad de Chile, en colaboración con la industria.

visualizar, mover y/o editar los post-its creados por otros.

Los post-its creados, solo requieren de un título, pero pueden poseer también una descripción, un color personalizado, elementos adjuntos como imágenes o archivos de audio, etiquetas y más. Por otro lado, se permite a clientes y desarrolladores votar sobre cada post-it para decidir si este será aprobado o rechazado para el futuro desarrollo del producto.

1.3. Limitaciones de la herramienta

Al utilizar la herramienta de scoping, los usuarios son capaces de definir el alcance de un producto, es decir, cómo será este en el futuro. Este concepto suele referirse como el *to-be* del producto, traducido del inglés corresponde literalmente a "lo que será". Pero, también existe otro concepto relacionado a lo que es un producto: el concepto de *as-is* o "lo que está". Este corresponde a las funcionalidades y características del producto que ya se encuentran desarrolladas y disponibles, es decir, el estado actual del producto.

Si bien el concepto del to-be se encuentra presente en la herramienta de scoping mencionada, esta no posee ninguna referencia al as-is del producto siendo desarrollado, ni permite a los usuarios manejar el estado de sus productos luego de la etapa de scoping. La herramienta Tablero Digital actual no se desarrolló pensando en manejar el presente de un producto, solo definir su futuro, y es justamente el manejo del as-is lo que motiva el desarrollo de esta memoria.

1.4. Objetivos de la memoria

Con la realización de este trabajo de memoria se busca dinamizar el uso del sistema Tablero Digital, extendiéndolo para permitir no solo definir el to-be de un producto, sino también, manejar su as-is, facilitando la gestión del progreso en el desarrollo de sus características. Como resultado de este trabajo se busca llegar a una solución útil y usable, que visibilice el avance del equipo de manera global y entendible, tanto para desarrolladores como clientes. A partir del objetivo general, se definen los siguientes objetivos específicos para el sistema:

- *Permitir la gestión de proyectos:* Incorporar los conceptos de proyecto (periodos de desarrollo con un objetivo fijo), e hitos (periodos que dividen un proyecto en objetivos más alcanzables) al sistema Tablero Digital.
- *Permitir la gestión del estado de cada funcionalidad:* Incorporar un mecanismo que permita definir el estado en que se encuentra cada característica del producto respecto a su desarrollo. El estado de las características se define por la acción que se realiza sobre ella (adición, modificación o eliminación) y la etapa en que se encuentra en un flujo determinado.

1.5. Estructura del documento

Este documento de memoria presenta el trabajo realizado, dividido en distintos capítulos. El primer capítulo presenta la herramienta a extender junto con las limitaciones que motivan este nuevo desarrollo. Luego, en el capítulo 2 de marco teórico, se menciona la relevancia de las metodologías ágiles en el manejo del estado de un producto, y se estudian sistemas existentes como inspiración para el desarrollo de la extensión.

El capítulo 3 explicita los requisitos considerados en la extensión del sistema y sus usuarios objetivos, seguido del diseño de arquitectura, modelo de datos y interfaz de usuario en el capítulo 4. Los detalles de la implementación del back-end y front-end se encuentran en el capítulo 5, y en el capítulo 6 se analiza la usabilidad e utilidad de la extensión realizada. Finalmente, se concluye en el capítulo 7 con posibles mejoras en el trabajo realizado y el trabajo futuro.

Capítulo 2

Marco teórico

Cuando se habla del as-is y el to-be, siempre se refiere al desarrollo de un producto, su estado actual y futuro; y en el caso del sistema Tablero Digital, este producto corresponde a un software.

Dentro del desarrollo de software, no es una novedad el interés en manejar y visualizar el progreso efectuado sobre un producto. De hecho, en la mayoría de casos, los desarrolladores y equipos se rigen por metodologías ya diseñadas y comprobadas para asegurar una efectiva organización, que permita tener un proyecto viable según las necesidades de la industria tecnológica. Estas metodologías consisten principalmente en formas de estructurar y ordenar el desarrollo de tecnología en el tiempo, de una forma que permita que el producto crezca indefinidamente sin demorar su lanzamiento. Dentro de estas metodologías se destacan, por ser las más utilizadas, las metodologías Ágiles, el método Kanban y el proceso Scrum.

2.1. Instrumentos de apoyo al desarrollo

2.1.1. Las metodologías Ágiles

Como su nombre lo indica, estas metodologías tienen como objetivo agilizar los procesos de desarrollo para no demorar el lanzamiento de productos al mercado. La forma en que lo logran, es generando productos mínimos viables (conocidos por la sigla MVP) que encapsulen lo más relevante de un sistema, para que este pueda empezar a ser utilizado en paralelo a su desarrollo.

Para la generación de los MVPs, las metodologías Ágiles separan el producto final en diversos sub-proyectos: periodos de tiempo (sprints) para los que se define como objetivo la generación de un nuevo MVP. Estos sub-proyectos se dividen en un número acordado de hitos, periodos de tiempo aún más acotados (típicamente alrededor de 2 semanas) donde se planifica periódicamente y con alta granularidad las tareas a realizar para completar el proyecto a tiempo.

2.1.2. El método Kanban

Este método busca permitir una mejor visualización del estado en que se encuentra un producto durante su desarrollo (as-is). Para esto, se hace uso de un tablero separado en columnas sobre el cual se puede indicar el nivel de avance en la implementación de distintas funcionalidades, las cuales se representan con tarjetas adhesivas o *tickets*. Cada ticket representa una acción que debe ser realizada sobre el producto, y las columnas sobre las cuales avanza representan las etapas de un flujo de desarrollo. Estas etapas pueden incluir las siguientes categorías para indicar el nivel de avance de una funcionalidad: "Por hacer", "En desarrollo", "En revisión"; y suelen terminar en "Completado". De esta forma los desarrolladores son capaces de obtener una visión del avance del producto con solo observar la posición de los tickets.

Esta metodología puede y suele ser utilizada en conjunto con la metodología ágil, para así poder observar el ritmo de desarrollo dentro de cada hito de manera individual, y así poder planificar de mejor manera los hitos futuros.

2.1.3. El proceso Scrum

El proceso Scrum consiste en una serie de actividades generalmente construidas sobre el uso de la metodolgía ágil y el método kanban para mejorar la productividad y bienestar de los desarrolladores. Dentro de estas actividades, podemos encontrar:

- La utilización de un *Backlog*: Una lista de todas las funcionalidades y características que serán parte del producto (to-be).
- La planificación de *Sprints*: Donde se define qué elementos del backlog formarán parte del próximo hito de desarrollo.
- La realización de *Daily Stand-Up Meetings*: Reuniones diarias donde cada miembro del equipo comenta en qué ha trabajado y puede levantar cualquier inquietud o necesidad referente a su desarrollo.
- La realización de *Sprint Retros*: Reuniones donde se analizan las situaciones tanto positivas como negativas del último hito (cierre del sprint) y se decide el camino a seguir para mejorar en el próximo hito o proyecto.

Todas estas actividades, junto con otras no mencionadas, permiten a los equipos de desarrollo trabajar de una manera más productiva y dinámica, mejorando así tanto su experiencia como la de sus clientes. En la actualidad, el proceso Scrum es ampliamente utilizado por equipos de todo el mundo, y por esto, existen múltiples herramientas para facilitar su utilización.

2.2. Herramientas de apoyo a Scrum

A continuación se presentan diversas herramientas de apoyo al uso de Scrum, describiendo sus características y funcionalidades.

2.2.1. Kanban Tool

Como su nombre sugiere, Kanban Tool [12] es una herramienta diseñada para la creación de tableros Kanban. Como se aprecia en la figura 2.1, la interfaz de este sistema presenta un tablero con 3 secciones por defecto, en las que el usuario puede añadir tarjetas y desplazarlas en función de su estado. La cantidad de columnas es modificable, y las tarjetas se componen de un título, descripción, miembro del equipo asignado, entre otras. Además, se ofrece al usuario la habilidad de añadir campos personalizados a los post-its para permitir una organización más específica.

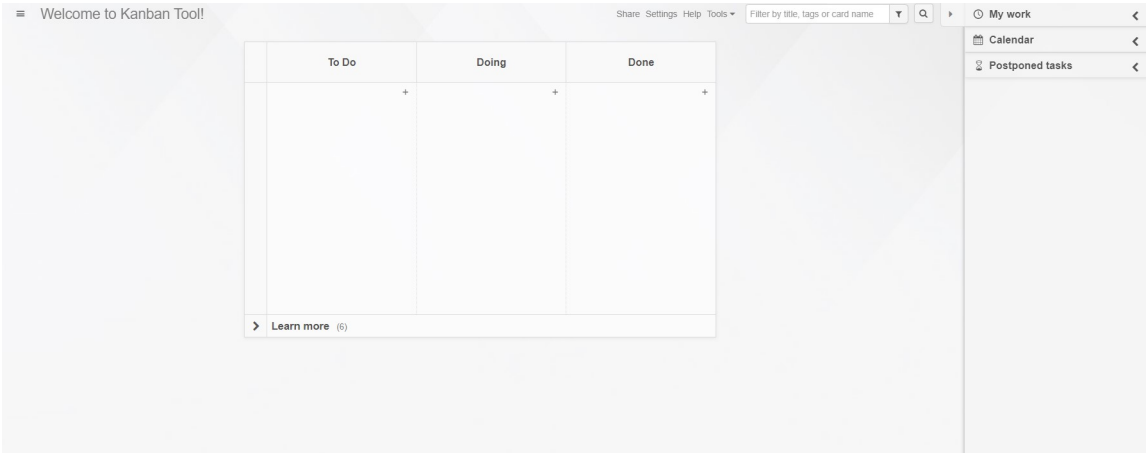

Figura 2.1: Visualización de un tablero en Kanban Tool

2.2.2. Notion

Notion [14] es una aplicación dedicada a ayudar a sus usuarios a organizar su vida, ofreciendo la habilidad de crear notas, recordatorios, listas, entre otras estructuras de texto enriquecido. Una de estas estructuras corresponde a la *Task list* que se puede apreciar en la figura 2.2. Esta estructura permite a los usuarios añadir post-its compuestos de títulos y campos personalizados a un tablero, el cual cuenta con tres columnas por defecto, cantidad extendible por el usuario.

2.2.3. Trello

Al igual que las aplicaciones vistas anteriormente, Trello [3] ofrece un tablero con columnas personalizables y post-its con diversos campos, además de la opción de añadir campos

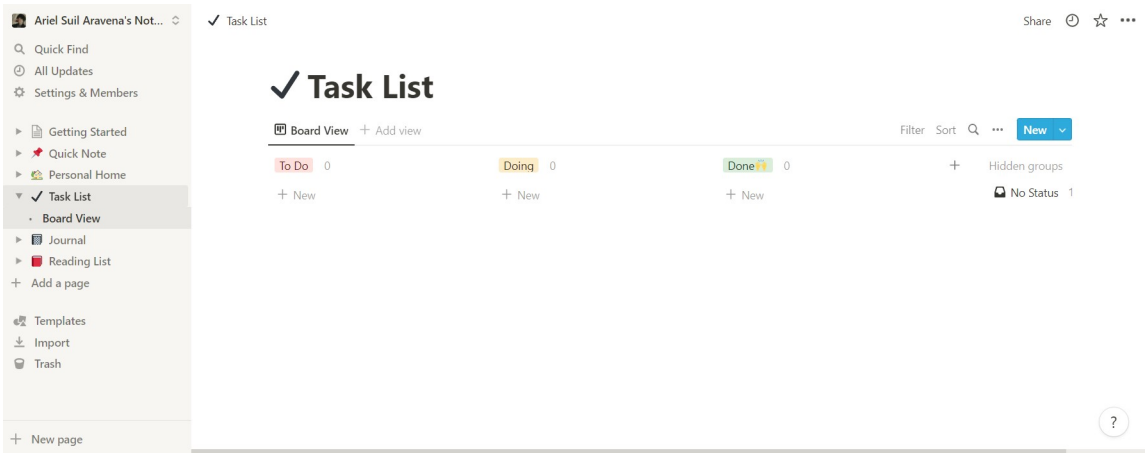

Figura 2.2: Visualización de un tablero en Notion

personalizados. Esta aplicación también ofrece opciones para filtrar las tarjetas mostradas, automatizar acciones tales como ordenar y/o modificar post-its en periodos de tiempo definidos, e incluso la habilidad de modificar el tipo de vista entre "Tablero", "Cronograma", "Panel", y otras.

Como se aprecia en la figura 2.3, esta herramienta cuenta con una gran cantidad de opciones para permitir a equipos de trabajo personalizar su experiencia para mejorar su productividad.

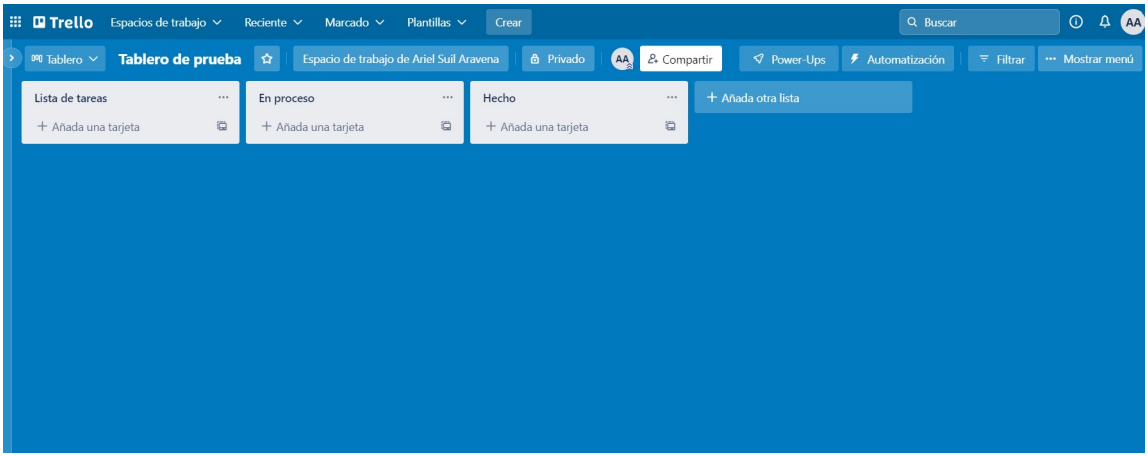

Figura 2.3: Visualización de un tablero en Trello

2.2.4. Jira

Este sistema desarrollado por Atlassian, empresa dueña de Trello desde 2017, cuenta con la gran mayoría de las funcionalidades ya mencionadas, además de una gran cantidad de integraciones de otros software que permiten la sincronización entre diversas herramientas de manera automática. Entre las aplicaciones integrables dentro de Jira [2] podemos encontrar herramientas de comunicaciones por texto, almacenamiento de código, testeo de código, diseño, entre miles de otras.

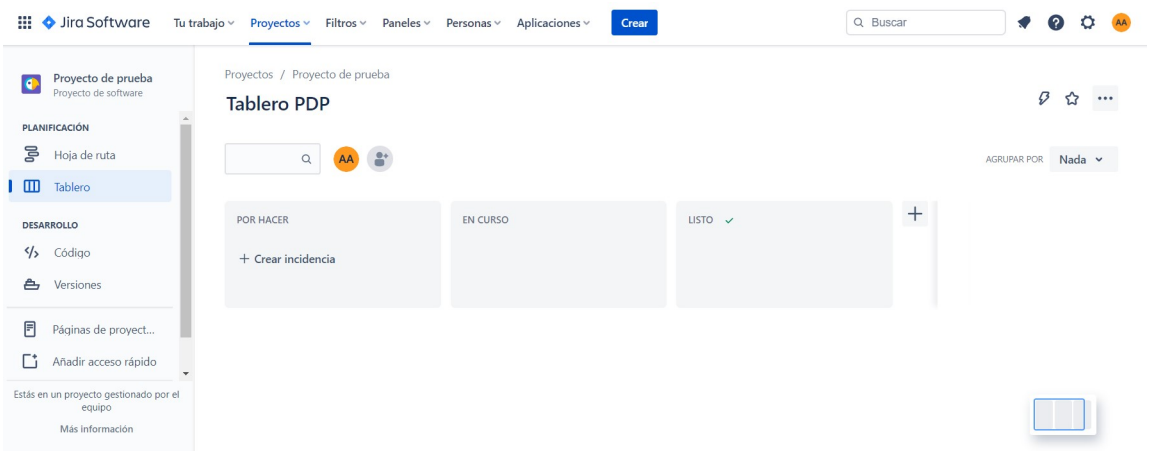

Figura 2.4: Visualización de un tablero en Jira

2.2.5. Tablero digital

En el estado actual del sistema Tablero Digital, el usuario puede definir el alcance de un producto generando así un backlog, pero al momento de querer aplicar Scrum, el sistema no ofrece la capacidad de trabajar como un tablero Kanban para la organización y realización de reuniones. Los usuarios de la herramienta de scoping pueden definir colores en sus post-its y así simular los estados por los que pasa cada funcionalidad en un tablero Kanban. Sin embargo, esto requeriría un previo acuerdo respecto a qué significa cada color; y aún así, el tablero no ofrece ninguna forma de definir proyectos ni hitos, lo que significa que los usuarios de todos modos deben duplicar la información de su tablero dentro de otra aplicación, antes de poder aplicar Scrum en su totalidad.

2.3. Limitaciones de opciones actuales

Si bien todas las opciones externas al sistema Tablero Digital cuentan con lo necesario para entrar a una reunión y realizar cualquier actividad de Scrum, ninguna posee el enfoque en el proceso de scoping que encontramos en este sistema.

Actualmente no existe una herramienta que unifique los conceptos de scoping en el marco de Scrum, y como se mencionó anteriormente, sólo es posible cumplir este objetivo al utilizar múltiples herramientas en paralelo, causando así una mayor carga de trabajo en los usuarios, al tener que actualizar manualmente la información en las distintas herramientas, incluyendo además la nueva responsabilidad de asegurar que no existan inconsistencias de información entre las mismas.

2.4. Tecnologías a utilizar

Dado el contexto de este desarrollo, siendo esta una extensión de un sistema ya existente, y ya que no existe la necesidad de incluir nuevas tecnologías con fines específicos, se decide continuar utilizando las tecnologías utilizadas en el desarrollo del sistema Tablero Digital actual.

Capítulo 3

Concepción de la solución

3.1. Descripción del proceso a apoyar

Una vez que se define el scoping inicial de un producto junto a un cliente, el equipo desarrollador precisa de una herramienta que le permita organizar su trabajo durante el periodo de desarrollo del producto. Con la extensión al sistema Tablero Digital realizada en esta memoria, se busca ofrecer las herramientas necesarias para que los desarrolladores pueden manejar sus proyectos con el proceso Scrum, sin tener que recurrir a una herramienta externa.

3.2. Requisitos funcionales

A continuación se presentan los requisitos funcionales considerados en el desarrollo de la extensión al sistema Tablero Digital:

- El sistema debe permitir al usuario modificar post-its para cambiar su estado dentro de la herramienta Kanban.
- El sistema debe permitir al usuario modificar post-its para (si se requiere) indicar el tipo de acción a realizar sobre la funcionalidad que representa. Permitiendo las opciones de "Adición", "Eliminación" o "Modificación".
- El sistema debe permitir al usuario crear hitos con una descripción editable. Los hitos deben almacenar el estado y tipo de cambio a realizar sobre cada post-it.
- El sistema debe permitir al usuario crear proyectos con un nombre, y una descripción editables. Los proyectos deben agrupar una cantidad de hitos definida por el usuario.
- El sistema debe propagar automáticamente la modificación del estado de cada post-it asociado a los hitos de desarrollo futuros.

- El sistema debe permitir visualizar los post-its, con su estado definido en cada hito, de manera independiente.
- El sistema debe definir el as-is automáticamente, considerando todos los post-its completados en el hito actual o anteriores.
- El sistema debe permitir filtrar los post-its mostrados, para solo presentar el as-is actual.
- Las funcionalidades y capacidades derivadas de esta extensión deben ser de carácter opcional para el usuario.

3.3. Requisitos no funcionales

A continuación se presentan los requisitos no funcionales considerados en el desarrollo de la extensión:

- El sistema debe presentar el estado de cada post-it de forma legible y eficiente, permitiendo a los usuarios comprender la información de manera rápida.
- El sistema debe ser intuitivo y fácil de usar.

3.4. Usuario objetivo

En el desarrollo del sistema Tablero Digital original, el usuario objetivo corresponde a los equipos de clientes y desarrolladores, en la etapa de definición del alcance de un producto. La extensión a realizar posee como usuario objetivo principal a los desarrolladores, y secundario a los clientes, ambos después de alcanzar un acuerdo inicial respecto al alcance de un producto. Las funcionalidades a incorporar ayudarán a los desarrolladores en la organización de su trabajo sobre el desarrollo del producto, pero al mismo tiempo, también permitirá la comunicación directa con el equipo de clientes durante el desarrollo, en un entorno ya conocido.

Capítulo 4

Diseño del sistema

A continuación se describe el diseño del sistema, particularmente enfocado en la extensión a realizar. Se comienza con la arquitectura del sistema; luego se revisa el modelos de datos y sus relaciones; las interfaces de usuario y sus consideraciones de experiencia de usuario; y finalmente el flujo de información entre servicios.

4.1. Arquitectura de la solución

Debido a la naturaleza de la extensión, y que no requiere de nuevos servicios, se decide mantener la arquitectura existente por el tablero actual. Como se puede observar en la figura 4.1, esta consiste de dos servidores, aplicaciones clientes y almacenamiento de información tanto en la nube como en una base de datos.

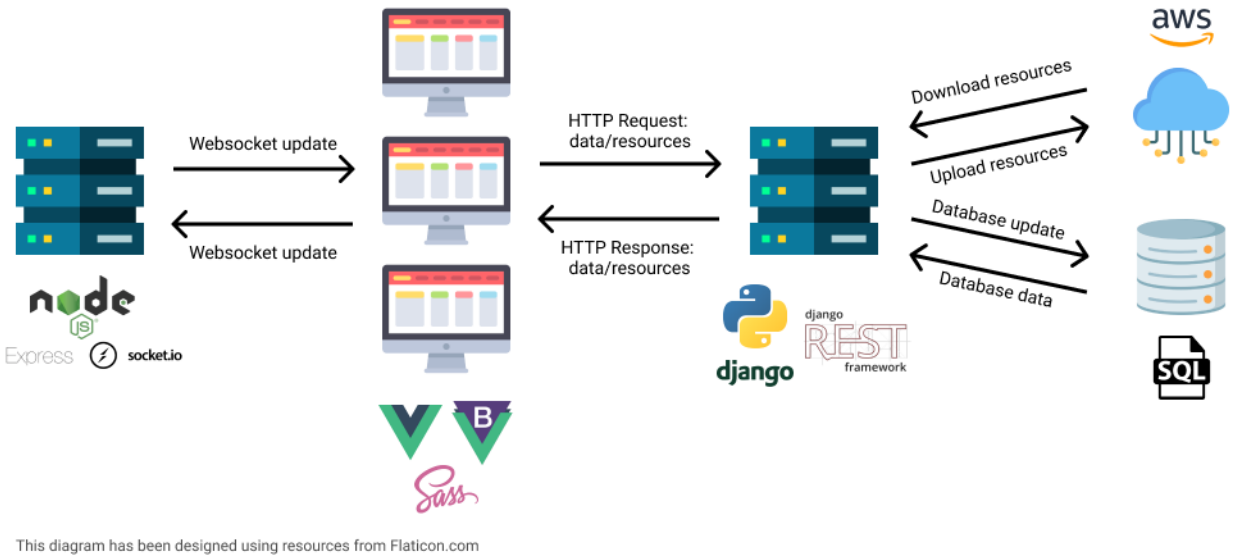

Figura 4.1: Arquitectura del tablero digital

El software que corre en el servidor de la derecha en la figura 4.1, se programa en el lenguaje Python [9], utilizando la librería Django [7] y el framework Django Rest [6] para manejar la conexión entre los clientes y los servicios de almacenamiento de datos. Los recursos adjuntos a post-its, como imágenes o archivos de audio, son almacenados en la nube con servicios de AWS [1], mientras que los datos descritos en el modelo de datos se almacenan en una base de datos relacional MySQL [5].

El software que corre en el otro servidor presente en la figura 4.1, se programa en NodeJS [8], utilizando las librerías Express [18] y socket.io [4] para manejar las conexiones en tiempo real entre usuarios del sistema. Los clientes se conectan al servidor con el protocolo *websocket* y emiten sus actualizaciones a todos los usuarios del sistema que se encuentren en el mismo tablero.

Finalmente, los clientes utilizados por el usuario se construyen con el framework VueJS [20] para JavaScript [15], con componentes heredados de BootstrapVue [16] y estilos definidos con el lenguaje Sass [11].

4.2. Modelo de datos

El modelo de datos del tablero digital actual consiste en los modelos de User, Layout, LayoutVersion, Board, WorkIn, PostIt, VoteIn, Tag, TagAtPostit y File. Las relaciones entre estos modelos se pueden observar en la figura 4.2.

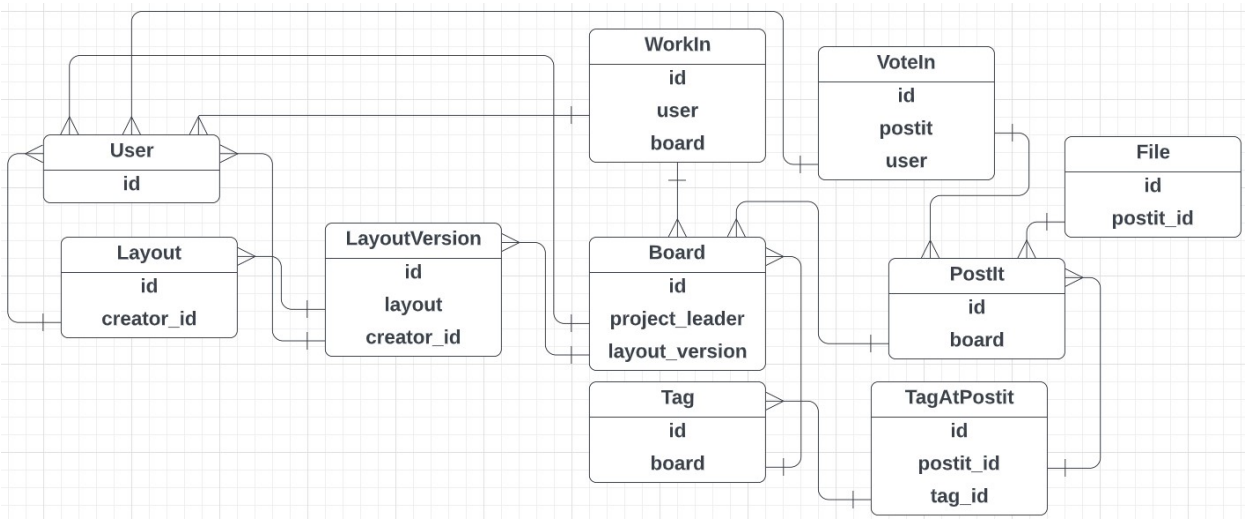

Figura 4.2: Diseño del modelo de datos actual

Para simplificar el trabajo a realizar, se planifica un modelo de datos poco intrusivo al modelo actual, que permita una incorporación gradual de las nuevas capacidades. Particularmente, se definen los modelos de Django para proyectos (*Project*) e hitos (*Milestone*), siguiendo las relaciones de la figura 4.3. Estas relaciones permiten que cada tablero se componga de múltiples proyectos, los cuales se componen a su vez de múltiples hitos.

Para el manejo de estados y acciones a realizar, se evalúan dos opciones: 1) Crear un

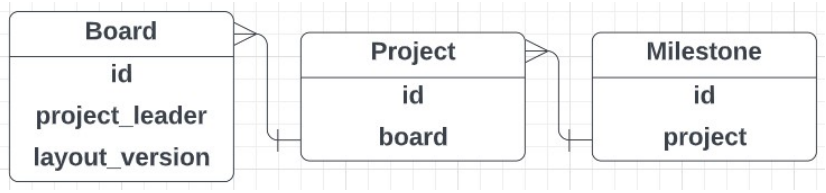

Figura 4.3: Adición de Proyectos e Hitos a la base de datos

modelo adicional para relacionar Postit y Milestone, definiendo campos para estados y tipo de acción en el mismo; o 2) asociar Postits y Milestone directamente, agregando los campos adicionales al modelo de Postit. El primer acercamiento tiene la ventaja de evitar la duplicación de información del post-it, ya que con el segundo acercamiento esta debería duplicarse para cada hito definido de un tablero. Pero al mismo tiempo, cuenta con la desventaja de añadir pasos extra al momento de solicitar la información de post-its, ya que la información referente a estados deberá solicitarse en una segunda llamada a la base de datos para cada uno de los post-its.

Con las ventajas y desventajas en consideración, se decide optar por la primera opción, ya que al ser el hito a visualizar una opción seleccionable por el usuario, es preferible que esta información se obtenga de manera separada a la información general de cada post-it. Este nuevo modelo *Action* sigue la relación mostrada en la figura 4.4.

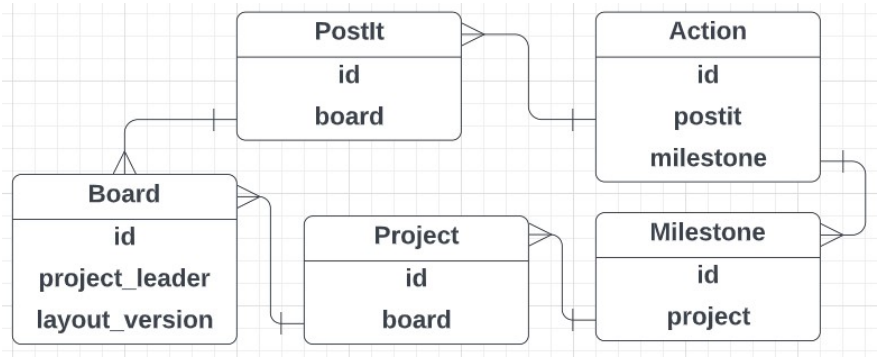

Figura 4.4: Relación entre modelos de Django nuevos y previos

4.3. Interfaz y Experiencia de Usuario

4.3.1. Creación de tableros

Al momento de crear un nuevo tablero no se requiere de nueva información del usuario, ni se planea modificar el flujo de creación. Sin embargo, al integrar la capacidad de administrar proyectos e hitos dentro del sistema, se requiere aclarar al usuario que el uso esperado del tablero es para la definición de un producto y su desarrollo futuro, en lugar de un proyecto o hito en particular. Por esto, se modifica el lenguaje utilizado de 'Mis Tableros' a 'Tableros de producto'.

4.3.2. Manejo de proyectos e hitos

Al diseñar la interfaz de manejo de proyectos e hitos, se decidió contener estas capacidades en una menú lateral desplegable. Esta decisión surge de la baja frecuencia de uso que tendrían estos controles, ya que el uso esperado de los mismos sería principalmente al momento de abrir el tablero para seleccionar el proyecto e hito actuales, y poder realizar actividades Scrum, sin mayores interacciones posteriores.

Actualmente, el tablero digital cuenta con controles en ambas esquinas del encabezado de la página. A la izquierda, se encuentran un botón con icono de "home" para volver a la sección 'Tableros de producto', donde se visualizan todos los tableros; además, hay un botón de menú desplegable con el nombre del tablero, que al ser presionado muestra las opciones de ver la información del tablero o eliminarlo. A la derecha, se encuentra otro botón de menú desplegable, esta vez con el nombre del usuario conectado, que al ser presionado muestra las opciones de volver a 'Tableros de producto', o cerrar sesión.

Considerando que el manejo de proyectos e hitos corresponde a acciones realizadas sobre el tablero en lugar del usuario, se decidió modificar los controles del lado izquierdo, para así mantener el diseño en línea con el mapa mental de los usuarios. La opción de volver a 'Tableros de producto' fue removida, ya que esta se encuentra disponible en el menú desplegable de la derecha, y el botón de menú desplegable fue modificado a un botón con icono de barras horizontales, utilizado frecuentemente para indicar la existencia de un menú lateral desplegable. Al eliminar el botón de menú desplegable, se elimina también el acceso a la información del tablero y la opción de eliminarlo, por lo que se incorporan estas opciones al del diseño del menú lateral, como se observa en la figura 4.5.

Figura 4.5: Menú lateral desplegable

Los controles para seleccionar y crear un nuevo proyecto corresponden a una sección con el título de "Proyectos", junto a un botón con el icono +, seguidos de un componente de tipo radio que permite seleccionar entre todos los proyectos creados. Cada opción de proyecto se representa con un botón con su nombre y un icono de verificación, en caso de ser el seleccionado actualmente. Al ubicar el cursor sobre uno de estos botones, se despliega un *pop-up* con la descripción del proyecto y un botón con un icono de edición. Este diseño es equivalente al utilizado en el manejo de hitos, y ambos se pueden observar en la figura 4.5. Al presionar el icono + o el icono de edición en el pop-up de un proyecto o hito, se abre un modal de edición con los mismos estilos que el resto de modales del sistema. Este modal contiene un formulario con los campos de título y descripción, para la creación/edición de proyectos, o con solo la descripción, para la creación/edición de hitos (ya que el número de hito se determina automáticamente). Este modal se modifica dependiendo de si es una creación o edición, para mostrar un título diferente y permitir o no la opción de eliminar el elemento seleccionado.

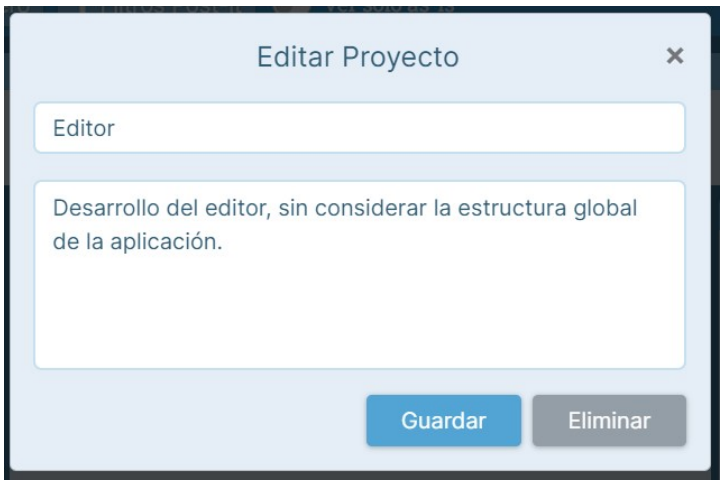

Figura 4.6: Modal de edición de un proyecto

4.3.3. Manejo de estados y tipos

Creación y edición de un post-it

En el tablero digital actual, los usuarios pueden crear post-its indicando solo su título y descripción en un modal simple y directo. Para editar o añadir información adicional a un post-it, se debe presionar sobre el mismo y seleccionar la opción de "Editar", lo que abre el mismo modal anterior, pero con una mayor cantidad de controles sobre el post-it. Considerando que la modificación de estados y tipos corresponde a una edición a un post-it, se toma la decisión de integrar los controles necesarios en esta sección de edición ya existente.

Considerando que generalmente los tableros Kanban ubican sus columnas de estado de manera lineal, ordenadas según el flujo de estados utilizado, al diseñar el control para estados se decide utilizar un control de tipo carrusel. Este control muestra el estado anterior, el seleccionado y el siguiente, y cuenta con flechas para controlar el desplazamiento a cada lado.

Con este control, se indica al usuario que los estados por los que pasa un post-it corresponden a una secuencia lineal, tal como las columnas en un tablero Kanban. Por otro lado, para el control de tipo de acción a realizar sobre el post-it se opta por un componente tipo radio, con el mismo formato que los selectores de proyecto e hito. En la figura 4.7, se observa la versión extendida del modal de edición de un post-it; con los campos de título y descripción en la parte superior izquierda, los botones encargados de adjuntar archivos de audio, imágenes y etiquetas en la parte superior derecha, los nuevos controles en la parte central, y el control de sección y botones de guardado y eliminación en la parte inferior.

Figura 4.7: Modal de edición de un post-it

Visualización de post-it

Una vez el usuario selecciona el estado y tipo de un post-it, es necesario mostrar esta información de manera clara y global, para permitir obtener una idea del estado en que se encuentra el producto. Por esto, se decide mostrar la información como parte de cada post-it del tablero. Específicamente, se decide utilizar el color como un indicador del estado de un post-it, y un icono en la esquina superior derecha como indicador del tipo de acción a realizar, como se observa en la figura 4.8.

Iconografía de estados y tipos

Para aclarar a los usuarios el significado del color como el estado de cada post-it, estos fueron utilizados también como el fondo de la opción seleccionada en el control de estados.

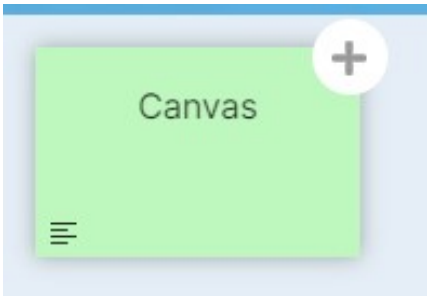

Figura 4.8: Post-it para una adición en desarrollo

De la misma forma, los iconos mostrados en la esquina del post-it se utilizan acompañando el nombre del tipo de acción en su control correspondiente. La asignación de colores e iconos se encuentran descritas en las tablas 4.1 y 4.2.

Color	Estado
Gris	No considerado
Amarillo	Por hacer
Verde	En Desarrollo
Lila	En Revisión
Naranjo	En Lanzamiento
Azul	Completado

Tabla 4.1: Asociación entre colores y estados de un post-it

Icono	Tipo de acción a realizar
Sin icono	Acción no definida
+	Añadir
↻	Modificar
⊘	Eliminar

Tabla 4.2: Asociación entre iconos y tipo de acción a realizar

4.3.4. Otros controles

Para facilitar el objetivo de visualizar el as-is del sistema, se permite a los usuarios filtrar los post-its para solo mostrar aquellos que ya fueron completados en este, o algún hito anterior. Este control consiste en una casilla de verificación con el texto "Ver solo as-is" y se encuentra en el centro del encabezado de la página para simplificar la navegación durante reuniones con clientes.

4.3.5. Consideraciones de experiencia de usuario

Para mejorar la experiencia de usuario en las funcionalidades de esta extensión del sistema Tablero Digital, algunas de las tareas necesarias para utilizar el sistema se realizan de manera autónoma. Al momento de crear un tablero, el sistema crea su primer proyecto e hito, de modo que este se pueda comenzar a utilizar inmediatamente sin configuraciones adicionales. De la misma forma, cada vez que un usuario crea un nuevo proyecto, se crea automáticamente su primer hito; y además, toda modificación realizada por un usuario al estado de un post-it es propagada a los hitos futuros para asegurar la consistencia temporal en el tablero.

Por otro lado, se optó por el uso de los componentes de tipo radio por sobre componentes de selección con un menú desplegable, ya que tanto para proyectos e hitos, como para la elección de tipos, se espera contar con pocas opciones sobre las que elegir. Esto significa que podemos mostrar todas al mismo tiempo sin saturar la navegación, y permitiendo selecciones más ágiles.

Finalmente, al momento de cargar la información de proyectos e hitos, se muestra una animación de carga en lugar del botón de creación, para indicar al usuario que el sistema está obteniendo información, y sus cambios sí tuvieron un efecto sobre el tablero.

Capítulo 5

Implementación de la solución

5.1. Back-end

En esta sección se describe en detalle la implementación utilizada para los modelos de Django incorporados (siguiendo el patrón Modelo-Vista-Controlador), y cómo estos son manejados por *viewsets* de Django REST Framework.

5.1.1. Modelo de datos

A continuación se especifican los modelos de datos utilizados en la implementación de la extensión realizada.

Project

Cada modelo de proyecto se compone de 4 campos: `id`, `name`, `description` y `board`. El campo de `id` es generado automáticamente cada vez que una nueva instancia del modelo es creada, mientras que los otros deben ser entregados por el sistema al crear la instancia. Los campos de `name` y `board` son requeridos, pero `description` es opcional. Al eliminar un tablero, también se eliminan todos sus proyectos asociados, gracias al uso de `on_delete=CASCADE` en su llave foránea. La definición del tipo de dato para cada campo se observa en la tabla 5.1.

Project	
`id`	`AutoField(primary_key=True)`
`name`	`CharField(40)`
`description`	`TextField(blank=True)`
`board`	`ForeignKey(Board, on_delete=CASCADE)`

Tabla 5.1: Modelo asociado a un proyecto

Milestone

En el modelo para hitos, mostrado en la tabla 5.2, se observa que este también cuenta con el campo `id` generado automáticamente, aparte de los campos: `number`, `description` y `project`; donde el campo `description` es opcional, y los otros requeridos. Dado que también se utiliza `on_delete=CASCADE`, al eliminar un proyecto todos los hitos asociados a este son eliminados.

Milestone	
`id`	`AutoField(primary_key=True)`
`number`	`PositiveSmallIntegerField`
`description`	`TextField(blank=True)`
`project`	`ForeignKey(Project, on_delete=CASCADE)`

Tabla 5.2: Modelo asociado a un hito

Action

El modelo de `Action` corresponde a la relación *many-to-many* existente entre hitos y post-its. Esta relación cuenta con llaves foráneas a instancias de ambos modelos, y especifica `on_delete=CASCADE` para eliminar la relación cuando cualquiera de las instancias relacionadas sea eliminada. Este modelo también cuenta con los campos de `id`, `as_is`, `state` y `action_type`; nuevamente con el campo `id` siendo generado automáticamente. Los campos de `as_is`, `state` y `action_type` son opcionales y poseen valores por defecto indicados en la tabla 5.3.

Action	
`id`	`AutoField(primary_key=True)`
`as_is`	`Boolean(default=false)`
`state`	`PositiveSmallIntegerField(default=0)`
`action_type`	`PositiveSmallIntegerField(default=0)`
`postit`	`ForeignKey(Postit, on_delete=CASCADE)`
`milestone`	`ForeignKey(Milestone, on_delete=CASCADE)`

Tabla 5.3: Modelo de relación many-to-many entre hito y post-it

Los valores que pueden tomar los campos de `state` y `action_type` corresponden a las opciones ya mencionadas para los estados y tipos de acción de un post-it. Estas opciones son asignadas a valores numéricos como se aprecia en las tablas 5.4 y 5.5.

Postit

En el sistema Tablero Digital, existía la capacidad de elegir el color de un post-it dentro una lista de seis colores definidos; además, se permitía definir un campo para "hito" en cada

Valor	Estado
0	No considerado
1	Por hacer
2	En Desarrollo
3	En Revisión
4	En Lanzamiento
5	Completado

Tabla 5.4: Asociación entre valores numéricos y estados de un post-it

Valor	Tipo de acción a realizar
0	Acción no definida
1	Añadir
2	Modificar
3	Eliminar

Tabla 5.5: Asociación entre valores numéricos y tipo de acción a realizar

post-it, como un valor numérico sin interacciones asociadas. Estas funcionalidades fueron eliminadas dada la integración de manejo de estados utilizando color, y la nueva versión del manejo de proyectos e hitos. Debido a esto, se modifica el modelo de Postit, eliminando los campos `color` y `sprint` utilizados por estas funcionalidades.

5.1.2. Viewsets

Dentro del framework Django REST existe la clase `ModelViewset` que extiende las funcionalidades de la clase `View` de Django, configurando automáticamente las rutas para la creación, eliminación y modificación de instancias de modelos de datos. Esta abstracción permite agilizar la creación de rutas solo definiendo cómo obtener los datos a manejar, y el *serializer* a utilizar para generar una respuesta, en este caso en formato `JSON`. Esta abstracción permite también la sobre-escritura de sus métodos para personalizar el comportamiento de las rutas a generar. Utilizando esto se generó una nueva abstracción llamada `CreateRestrictedViewSet` para generalizar las restricciones aplicadas en la creación de instancias de los modelos usados en la extensión del sistema, evitando así duplicación de código entre sus viewsets.

ModelViewSet

Esta clase cuenta con múltiples métodos y campos que pueden ser sobre-escritos para el manejo personalizado de modelos de datos. A continuación se describen el funcionamiento de aquellos que fueron sobre-escritos en esta implementación.

- `permission_classes`: Campo que recibe como valor un arreglo de clases que describen los requerimientos necesarios para que un usuario pueda realizar acciones sobre instan-

cias de un modelo de datos. Este campo es sobre-escrito en todas las clases que requieren que el usuario haya iniciado sesión, gracias a la clase `IsAuthenticated` ofrecida por Django REST.

- `serializer_class`: Campo que recibe como valor una clase que extienda la clase `Serializer` ofrecida por Django REST. Esta clase cuenta con una referencia a un modelo de datos y cómo convertirlo al formato `JSON`.
- `get_queryset`: Método que no recibe argumentos y retorna un `QuerySet` de Django con las instancias a considerar en la generación de una respuesta.
- `create`: Método que recibe como argumento la `HTTPRequest` enviada al servidor y extrae los campos necesarios para crear una nueva instancia del modelo de datos asignado. El método retorna la instancia creada.
- `update`: Método que al igual que el método `create`, recibe y extrae información de una `HTTPRequest`, pero la utiliza para actualizar una instancia existente. El método retorna la instancia modificada.
- `destroy`: Método que recibe un identificador de una instancia y la elimina del sistema.

CreateRestrictedViewSet

Esta clase extiende `ModelViewSet`, y sobreescribe su método `create` para:

1. Verificar si la instancia del modelo ya fue creada llamando un nuevo método `already_created` y retornando error código `400_BAD_REQUEST` en el caso `True`.
2. Verificar si el usuario puede crear instancias del modelo llamando un nuevo método `can_create` y retornando error código `401_UNAUTHORIZED` en el caso `False`.
3. Ejecutar el método `create` definido en `ModelViewSet` ejecutando `super().create(...)`.
4. Ejecutar un nuevo método `on_create` que recibe como argumento la instancia del modelo recién creado.

La implementación por defecto de estos nuevos métodos se indica a continuación:

- `already_created`: Retorna `False`.
- `can_create`: Retorna `False`.
- `on_create`: `pass`. No ejecuta código.

Además, se sobre-escribe `permission_classes` para solo permitir modificaciones provenientes de usuarios que ya iniciaron sesión.

ProjectViewSet

El viewset para el modelo de proyectos solo debe permitir a usuarios crear proyectos sobre los tableros de los que son parte, por lo que se sobre-escribe `can_create` para verificar la relación del usuario y el tablero en el modelo `WorkIn`. De la misma forma, se sobre-escribe `get_queryset` para extraer de la `HTTPRequest` el `id` del tablero seleccionado, y así solo enviar de respuesta los proyectos de dicho tablero. Finalmente, se sobre-escribe el método `on_create` para también crear el primer hito del proyecto creado.

MilestoneViewSet

En el caso del viewset para hitos también sobre-escriben el método `can_create`, el método `get_queryset` y el método `on_create`, pero, además sobre-escribe el método `already_created`. Los primeros tres siguen razones similares al caso de proyectos, evitando que usuarios creen hitos en proyectos que pertencen a tableros donde el usuario no ha sido añadido; asegurando que solo se incluyan los hitos del proyecto seleccionado; y creando las instancias del modelo `Action` que relacionan el nuevo hito con los post-its existentes. Por otro lado, el método `already_created` se sobre-escribe para no permitir la creación un hito con el mismo número que uno ya existente en el proyecto seleccionado, asegurando así que los números que identifican un hito sean únicos en cada proyecto.

ActionViewSet

Para el viewset del modelo `Action`, se sobre-escribe el método `get_queryset` para solo incluir las instancias relacionadas al hito seleccionado. Y además, se sobre-escriben los métodos de `create`, `update` y `destroy` para siempre retornar error `401_UNAUTHORIZED`, ya que este modelo solo se maneja de manera interna por los viewsets de `Postit` y `Milestone`.

BoardViewSet

Por otra parte, es necesario actualizar el viewset que controla el modelo de tableros, para generar un proyecto y un hito iniciales luego de cada creación. Este viewset ya sobre-escribe el método `create` para generar la relación `WorkIn` con el usuario actual, por lo que solo se requiere incorporar el código para crear el proyecto e hito luego de lo actualmente ejecutado.

PostitViewSet

Al igual que en la creación de un nuevo hito, al crear un post-it es necesario crear la relación entre dicho post-it y todos los hitos del tablero, es decir, crear las instancias necesarias del modelo `Action` sobre-escribiendo el método `create`. Por otro lado, al momento de editar un post-it es necesario actualizar el estado y tipo ingresados por el usuario en la relación con del hito seleccionado. Para esto se sobre-escribe el método `update`, para que luego de

aplicar las modificaciones a la instancia del modelo post-it, se obtengan de la `HTTPRequest` los valores de `state` y `action_type` para actualizar en el hito seleccionado, y propagar los cambios a los hitos futuros, modificando el valor del campo `as_is` en los casos donde un post-it pasa al estado completado.

Las instancias del modelo Action que relacionan los post-its con sus hitos podían ser creadas junto con las instancias de los modelos Postit y Milestone, o al momento de editar los campos en el método `update` de Postit. Se decidió crear a todas de manera preventiva, al crear Postit y Milestone, aumentando el tiempo de respuesta de la llamada de creación. De lo contrario, al realizarlo en el método `update`, este aumentaría aún más su tiempo de respuesta al tener que primero verificar si la instancia del modelo ya existe, y de no ser así, además crearla.

Serializers

Los serializers utilizados por los nuevos viewsets extienden la clase `ModelSerializer` ofrecida por Django REST, y se configuran a su modelos correspondientes, manteniendo la configuración de campos como el valor `'__all__'`, para incluirlos todos en la respuesta.

5.2. Front-end

El desarrollo en el área de front-end se encarga de ofrecer al usuario una interfaz con la cual interactuar, y con ello la permitir una conexión simplificada con el back-end. A continuación se describen los componentes creados para la generación de esta interfaz, y el flujo de información actualizado, considerando los nuevos modelos de datos.

5.2.1. Componentes creados

Al utilizar un framework moderno de JavaScript, como lo es VueJS, se trabaja definiendo diversos componentes, los cuales son entidades que encapsulan lógica y/o estilos, para permitir su reutilización en diversas áreas de un sistema.

Pill

El componente `<pill />` corresponde a un botón *HTML* con un icono opcional y título modificables. Este botón posee un fondo blanco, bordes redondeados, configura el uso de `ellipsis` (puntos suspensivos) en caso de poseer un título muy largo, y posee un estilo desactivado que se activa cuando recibe el *prop* correspondiente[1]. Al ser presionado, este botón emite un único evento llamado `pill-click`, el cual debe ser manejado por su componente padre.

[1]Prop también entregado al botón nativo generado por el componente para desactivar su interactividad.

Dentro de la estructura de este componente, al igual que todos los creados para esta memoria, se definen estilos `scoped`, que solo afectan al componente actual; así, se mantiene un mayor orden dentro del código, sin efectos secundarios inesperados. Los únicos casos en que se utilizan estilos globales es al pasar una clase como parámetro para otro componente, ya que este no sería capaz de detectarlo; en estos casos, se opta por incluir el nombre del componente base como prefijo, para evitar conflictos de nombres.

PillChooser

Se utiliza este componente para permitir al usuario manejar los proyectos e hitos del tablero. Este cuenta con un título modificable, un botón de creación asociado a un modal configurable, y una secuencia de componentes Pill por cada opción recibida como prop. También, se recibe como prop la variable `loading` que determina si se debe mostrar el botón de creación de proyecto/hito, o un `<b-spinner />`[2] para indicar que se están actualizando las opciones mostradas, además, esta variable se entrega como el prop de deshabilitación de los componentes Pill. Cada vez que se selecciona una opción, se modifica su icono para indicar que es la seleccionada, y se emite un evento configurable hacia el evento padre. Por otro lado, este componente también genera el tooltip a mostrar cuando un usuario ubica su cursor sobre una opción, mostrando su título, descripción, y botón de edición.

Modal

Este componente no posee lógica propia, y se define como un contenedor con los estilos necesarios para crear un modal consistente con los estilos del sistema Tablero Digital. La creación de este componente permite aplicar modificaciones globales a todos los elementos que lo utilizan, agilizando así el rediseño futuro del sistema.

Se utiliza el componente `<b-modal>` de la librería Bootstrap Vue como base, y se utilizan los props y estilos definidos en los modales ya existentes en el sistema. Para permitir que el componente sea utilizado como contenedor, se utiliza la funcionalidad de `<slot />` de VueJS, que permite marcar dónde deberá ser reemplazado el contenido entregado como hijo a este componente.

ModalInput

Similar al caso del componente Modal, este componente hereda la base `b-input` de Bootstrap Vue, y la complementa con los estilos utilizados en los modales actuales del tablero; con la consideración especial, de que al ser un componente que maneja un campo de texto, este debe ser configurado para permitir el uso del prop `v-model`, que permite asociar una variable de un componente directamente al valor del campo de texto, optimizando así el proceso de renderización del mismo.

[2]Componente de indicador de carga ofrecido por la librería Bootstrap Vue

CreateMilestoneModal y CreateProjectModal

Estos componentes utilizan los componentes de Modal y ModalInput para generar los modales mostrados al momento de crear o editar un proyecto o hito. Para esto, reciben como prop el hito/proyecto que se debe editar; en caso de no recibir un hito/proyecto, se considera una creación. Las diferencias entre creación y edición se observan en el título del modal y los botones de acción.

BoardSidebar

Este componente es el encargado de generar el menú lateral desplegable mostrado al usuario para el manejo de proyectos e hitos. Se utiliza una `<b-sidebar />` de Bootstrap Vue como base, y se construye el contenido del menú con los componentes `<pill-chooser />`, `<create-milestone-modal />` y `<create-project-modal />`, pasando a PillChooser todos los props necesarios para que una instancia maneje los hitos y otra los proyectos.

StateCarrousel

Para definir el estado en que se encuentra un post-it, se genera este componente con dos botones con icono de flecha apuntando a cada extremo, y tres indicadores de estados. Los botones avanzan el carrusel en la dirección indicada, cambiando el valor del estado seleccionado por el anterior o siguiente en el arreglo de estados posibles. Las flechas se muestran u ocultan dependiendo de si existen más estados en la dirección apuntada. Y el estado central seleccionado posee un estilo con su color correspondiente de fondo. Cada vez que se modifica el estado, se emite un evento indicando este cambio.

ActionTypePicker

Para definir el tipo de acción a realizar sobre un post-it, se cuenta con este componente, el cual muestra componentes Pill por cada tipo de acción disponible, utilizando el icono y texto asociados a cada una. Cada vez que se selecciona un nuevo tipo, se emite un evento indicando el valor seleccionado.

PostitActionTypeIndicator

Al igual que ActionTypePicker, este componente utiliza el icono asociado a cada acción, pero esta vez de manera no interactiva, rodeándolo de un círculo blanco y ubicándose en la esquina superior derecha de su componente padre. Este componente cuenta con dos tamaños disponibles, para ser utilizado por la visualización de post-its normal, y la visualización de post-its por sección[3].

[3]El sistema Tablero Digital cuenta con un modo de visualización en que solo se observan los post-its de una sección en particular, con un tamaño mucho mayor al del tablero normal.

5.2.2. Componentes modificados

Para la integración de los nuevos componentes y el nuevo flujo información, se requirió modificar algunos de los componentes ya existentes en el sistema. A continuación se describen las modificaciones más relevantes.

EditPostitModal

En el componente de edición de post-its se añade el uso de los nuevos controles de estados y tipos con los componentes StateCarousel y ActionTypePicker, al mismo tiempo que se eliminan los controles obsoletos de color y sprint. En este componente se realiza la llamada al back-end para la edición de post-its, por lo que se añade a esta llamada la información referente a estados y tipos obtenidos por los eventos emitidos desde ActionTypePicker y StateCarousel.

NavBarProject

En este componente se define la barra de navegación del encabezado de la página. Aquí, se elimina el menú que existía previamente y se incorpora el nuevo menú lateral con el componente BoardSidebar; además de incorporar el control de filtro para visualizar solo el as-is. El filtro para visualizar el as-is corresponde a una input de tipo casilla de verificación, la cual emite un evento cada vez es modificada.

Section

Cada sección del tablero determina qué post-its debe mostrar con la lógica definida en el componente Section. Esta lógica se modifica para no solo considerar los filtros existentes del sistema, sino también el nuevo filtro por as-is. Además, en este componente se asocia cada post-it con su información de la relación Action para el hito seleccionado, evitando así la necesidad de buscar repetidas veces esta información en componentes hijos.

PostitSmall y PostitLarge

Estos componentes son los encargados de dibujar en pantalla cada post-it del tablero, en sus versiones normal y grande. En ambos componentes se actualizó el color de fondo y se incorporó el uso del componente PostitActionTypeIndicator, utilizando la información de color y tipo asociada al post-it por el componente Section.

Board

El componente Board es el encargado de entregar los props necesarios y recibir los eventos emitidos por los componentes más importantes del sistema: Section, EditPostitModal y NavBarProject. Por lo que, se modifica este componente para también manejar la información referente a los modelos Project, Milestone y Action, como se describe en la sección 5.2.3.

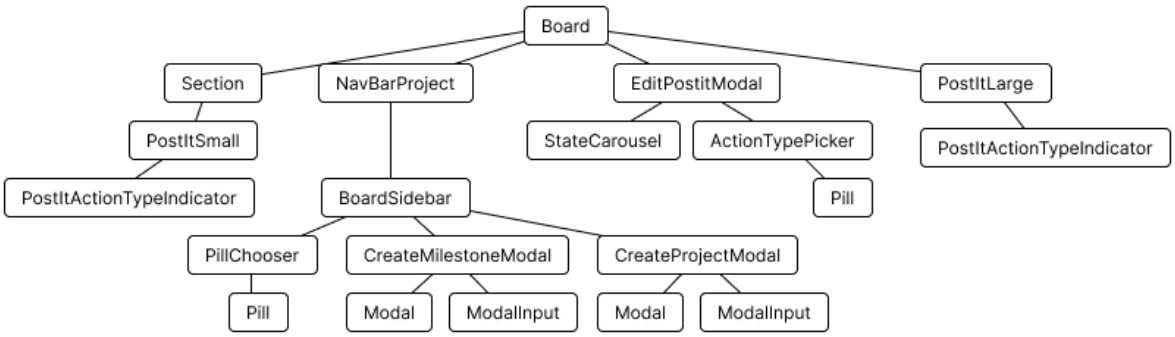

Figura 5.1: Árbol de componentes utilizados a partir de Board

5.2.3. Flujo de información

Al momento de hablar del flujo de información, podemos diferenciar entre tres escenarios: la información transmitida entre componentes del front-end (interna), la información que se comparte entre front-end y back-end (externa), y la información enviada entre clientes que editan el tablero en tiempo real (a través de websocket).

Interna

Como ya se mencionó al hablar sobre el componente Board, este recibe gran parte de la información utilizada para generar la interfaz de usuario del sistema Tablero Digital. Por lo mismo, este es el encargado de almacenar y administrar los datos más relevantes para distribuirlos a sus hijos a través de props.

En la figura 5.1 se aprecia la relación existente entre los diversos componentes del front-end. En este esquema, cada línea representa una posible comunicación en ambos sentidos, donde el componente superior es el padre, que puede enviar props a su hijo; y el componente inferior es el hijo, que puede emitir eventos a su padre.

Para evitar el movimiento de información de manera innecesaria, se decide almacenar los datos en el primer componente que sea antecesor de todos los componentes que la requieren (excepto de si mismo). Por ejemplo, la variable asociada al campo de la descripción actual del componente CreateProjectModal solo es requerida por su ModalInput y el modal en sí al momento de realizar una llamada al back-end, por lo tanto, la variable se almacena en el mismo componente del modal. Por otro lado, el hito de desarrollo seleccionado, es un valor necesario para el componente PillChooser que lo define, pero también para el componente

Section al momento de filtrar los post-its que mostrará, por lo que este valor debe almacenarse más arriba en la raíz del árbol de componentes, el componente Board.

Externa

Para la extensión del tablero se mantuvo la lógica de la versión actual del tablero, donde se cuenta con una comunicación al back-end siguiendo dos lógicas de ejecución. La primera corresponde a cuando el usuario realiza una modificación que provoca un efecto en los datos almacenados en el sistema. En estos casos, el componente que maneja los datos realiza una llamada HTTP al back-end con los métodos POST, PUT o DELETE, y al obtener una respuesta positiva, inicia la segunda lógica de ejecución. Esta segunda lógica realiza llamadas HTTP con el método GET, para obtener los datos actualizados del sistema, para así mostrar al usuario los efectos de sus acciones. Estas llamadas GET suelen provocar la ejecución de otras llamadas para obtener información sobre los modelos relacionados a los solicitados por la llamada anterior. En la figura 5.2 se puede observar la secuencia de llamadas GET a realizar para obtener la información del tablero actual, y los eventos ejecutables por el usuario que vuelven a activar la secuencia a partir de algún punto.

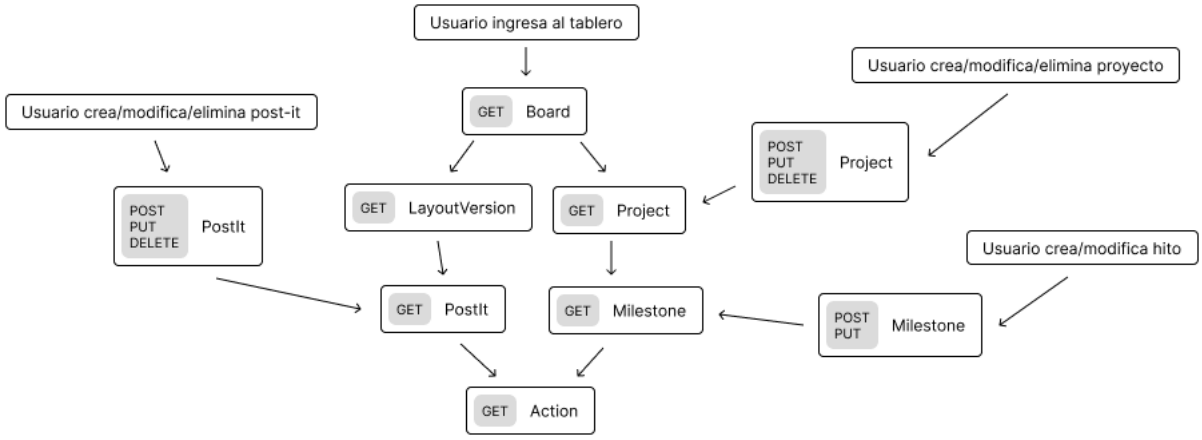

Figura 5.2: Flujo de información solicitada al back-end en secuencias

WebSocket

La conexión existente entre usuarios en un tablero digital, previo a la extensión, está directamente conectada con la obtención regular de información desde el back-end. Toda creación o actualización a un post-it realizada por un usuario, es transmitida por el servidor de websocket para ejecutar la misma secuencia de solicitud de información utilizada cuando el usuario en el cliente actual realiza los cambios.

Capítulo 6

Evaluación de la solución

A continuación se describe el proceso de evaluación de la extensión desarrollada sobre el sistema Tablero Digital. Para esto se mide el rendimiento de la herramienta en términos de tres métricas: *Correctitud operacional* (indica si la aplicación realiza las tareas comprometidas), *usabilidad* (indica si la aplicación es clara y simple de navegar para los usuarios), y *utilidad* (indica si la aplicación es útil para cumplir los objetivos del usuario).

6.1. Evaluación de la correctitud

Para la evaluación de la correctitud de la funcionalidad del sistema, se dispone de casos de prueba sobre tanto las funcionalidades previas como las nuevas. De este modo se busca asegurar que ninguna funcionalidad se ve comprometida por el nuevo desarrollo[1]. A continuación se detallan estos casos de prueba.

6.1.1. Casos de prueba para funcionalidades previas

1. Título: Creación de una cuenta.
   Descripción: Crear una nueva cuenta de usuario en el sistema.
   Requisitos:
   (a) Poseer un correo electrónico.
   Pasos:
   (a) Acceder al sitio web.
   (b) Presionar el enlace en la palabra 'aquí' de la pantalla de 'Log in'.
   (c) Completar los datos solicitados.
   (d) Presionar el botón de 'Crear'.

[1]No se incluyen pruebas sobre el creador de layouts ya que este se encuentra en un proyecto separado, no involucrado en la extensión realizada.

Resultado esperado: El usuario es redireccionado a la pantalla de 'Log-in'.

2. Título: Acceso a una cuenta.
   Descripción: Acceder a una cuenta de usuario del sistema desde la pantalla de 'Log-in'.
   Requisitos:
   (a) Poseer una cuenta en el sistema.

   Pasos:
   (a) Acceder a la pantalla de 'Log-in'.
   (b) Ingresar el correo y contraseña utilizados en la creación de la cuenta.
   (c) Presionar el botón de 'Entrar'.

   Resultado esperado: El usuario es redireccionado a la pantalla de 'Tableros de producto'.

3. Título: Creación de un tablero.
   Descripción: Crear un nuevo tablero digital desde la pantalla de 'Tableros de producto'.
   Requisitos:
   (a) Iniciar sesión en el sistema.

   Pasos:
   (a) Acceder a la pantalla de 'Tableros de producto'.
   (b) Presionar el botón '+' debajo del título 'Tableros de producto'.
   (c) Ingresar título, descripción y cliente del tablero a crear.
   (d) Seleccionar el layout previamente creado 'Concepción de producto'.
   (e) Presionar el botón de 'Crear'.

   Resultado esperado: El usuario es redirigido al tablero digital creado y se le solicita seleccionar su equipo.

4. Título: Acceso a un tablero.
   Descripción: Ingresar a un tablero digital previamente creado desde la pantalla de 'Tableros de producto'.
   Requisitos:
   (a) Iniciar sesión al sistema.
   (b) Poseer un tablero creado.

   Pasos:
   (a) Acceder a la pantalla de 'Tableros de producto'.
   (b) Presionar sobre la tarjeta con el nombre y descripción del tablero creado.

Resultado esperado: El usuario es redirigido al tablero digital.

5. Título: Creación de un post-it.
   Descripción: Crear un post-it dentro de un tablero digital.
   Requisitos:
   (a) Iniciar sesión al sistema.
   (b) Poseer un tablero creado.
   Pasos:
   (a) Acceder a un tablero previamente creado.
   (b) Seleccionar el equipo 'Desarrolladores'.
   (c) Presionar el botón '+' de cualquier sección dentro del tablero.
   (d) En la ventana emergente, ingresar título y descripción.
   (e) Presionar el botón 'Guardar'.
   Resultado esperado: La ventana de emergente se cierra y el post-it se muestra en la sección correspondiente.

6. Título: Edición de un post-it.
   Descripción: Modificar título, descripción y sección de un post-it.
   Requisitos:
   (a) Iniciar sesión al sistema.
   (b) Poseer un tablero creado.
   (c) Poseer un post-it creado.
   Pasos:
   (a) Acceder a un tablero previamente creado.
   (b) Presionar el post-it previamente creado.
   (c) En la ventana emergente, presionar 'Editar'.
   (d) En la nueva ventana emergente, modificar título, descripción y sección.
   (e) Presionar el botón 'Guardar'.
   Resultado esperado: La ventana de emergente se cierra y el post-it se actualiza en la sección correspondiente.

7. Título: Colaboración en tiempo real.
   Descripción: Realizar modificaciones a un post-it en un tablero de una cuenta 'A' conectada al sistema y observar cambios en otro cliente con la cuenta 'B', siendo 'B' un colaborador en el tablero de 'A'.
   Requisitos:

(a) Iniciar sesión al sistema en dos cuentas diferentes ('A' y 'B').

(b) Poseer un tablero creado en la cuenta 'A'.

Pasos:

(a) Acceder a un tablero previamente creado en cuenta 'A'.

(b) Presionar el botón 'Invitar'.

(c) Ingresar el correo electrónico de la cuenta 'B'.

(d) En el cliente con la cuenta 'B' iniciada, ingresar al tablero de 'A'.

(e) Desde el cliente 'A', crear un post-it.

(f) Desde el cliente 'B', editar el post-it creado por 'A'.

Resultado esperado: La creación y edición de post-its se ve reflejada en ambos clientes.

6.1.2. Casos de prueba para nuevas funcionalidades

1. Título: Creación de un proyecto.
   Descripción: Crear un proyecto dentro de un tablero digital.
   Requisitos:

   (a) Iniciar sesión al sistema.

   (b) Poseer un tablero creado.

   Pasos:

   (a) Acceder a un tablero previamente creado.

   (b) Presionar el icono de barras a la izquierda del título del tablero.

   (c) En el menú desplegable lateral, presionar el símbolo '+' al lado del título 'Proyectos'.

   (d) En la ventana emergente, completar los campos solicitados.

   (e) Presionar el botón 'Guardar'.

   Resultado esperado: La ventana de emergente se cierra y el proyecto creado aparece en el listado bajo el título de 'Proyectos'.

2. Título: Creación de un hito.
   Descripción: Crear un hito dentro de un tablero digital.
   Requisitos:

   (a) Iniciar sesión al sistema.

   (b) Poseer un tablero creado.

   Pasos:

   (a) Acceder a un tablero previamente creado.

(b) Presionar el icono de barras a la izquierda del título del tablero.

(c) En el menú desplegable lateral, presionar el símbolo '+' al lado del título 'Hitos'.

(d) En la ventana emergente, completar los campos solicitados.

(e) Presionar el botón 'Guardar'.

Resultado esperado: La ventana de emergente se cierra y el hito creado aparece en el listado bajo el título de 'Hitos'.

3. Título: Edición de un proyecto.

   Descripción: Editar un proyecto dentro de un tablero digital.

   Requisitos:

   (a) Iniciar sesión al sistema.

   (b) Poseer un tablero creado.

   Pasos:

   (a) Acceder a un tablero previamente creado.

   (b) Presionar el icono de barras a la izquierda del título del tablero.

   (c) En el menú desplegable lateral, ubicar el cursor sobre un proyecto existente.

   (d) En la descripción emergente, presionar el icono de lápiz.

   (e) En la ventana emergente, completar los campos solicitados.

   (f) Presionar el botón 'Guardar'.

   Resultado esperado: La ventana de emergente se cierra y se actualiza el proyecto modificado.

4. Título: Edición de un hito.

   Descripción: Editar un hito dentro de un tablero digital.

   Requisitos:

   (a) Iniciar sesión al sistema.

   (b) Poseer un tablero creado.

   Pasos:

   (a) Acceder a un tablero previamente creado.

   (b) Presionar el icono de barras a la izquierda del título del tablero.

   (c) En el menú desplegable lateral, ubicar el cursor sobre un hito existente.

   (d) En la descripción emergente, presionar el icono de lápiz.

   (e) En la ventana emergente, completar los campos solicitados.

   (f) Presionar el botón 'Guardar'.

Resultado esperado: La ventana de emergente se cierra y se actualiza el hito modificado.

5. Título: Edición de estados y tipos de un post-it.

   Descripción: Modificar el estado y tipo de un post-it para ubicarlo dentro del proceso kanban.

   Requisitos:

   (a) Iniciar sesión al sistema.

   (b) Poseer un tablero creado.

   (c) Poseer un post-it creado.

   Pasos:

   (a) Acceder a un tablero previamente creado.

   (b) Presionar el post-it previamente creado.

   (c) En la ventana emergente, presionar 'Editar'.

   (d) En la nueva ventana emergente, modificar el estado y tipo del post-it.

   (e) Presionar el botón 'Guardar'.

   Resultado esperado: Los cambios realizados se aplican para futuros hitos dentro del mismo proyecto, pero no tienen efecto sobre otros proyectos ni hitos previos.

6.1.3. Resultados

Cada caso de prueba se ejecuta siguiendo los pasos especificados en su desglose, y se observa que para cada uno, tanto para la evaluación de funcionalidades anteriores, como para la evaluación de funcionalidades nuevas, las respuestas son positivas, y el resultado obtenido es el mismo al resultado esperado. Con esto, damos por aprobada la evaluación de correctitud a un 100 %.

6.2. Evaluación de usabilidad y utilidad

Esta evaluación consta de dos etapas a realizar con la ayuda de usuarios expertos del sistema, previo a la extensión desarrollada en esta memoria. La etapa de familiarización de los usuarios con respecto a los cambios aplicados al sistema, y la etapa de mediciones de usabilidad/utilidad. En la etapa de familiarización se solicita a los usuarios completar una serie de tareas sobre las nuevas funcionalidades del sistema. Esta etapa es importante para asegurar que los usuarios posean una buena visión del impacto provocado por las nuevas implementaciones. En la etapa de medición, se realizan dos estudios. En primer lugar, se analiza el rendimiento de los usuarios durante la familiarización, para obtener métricas referentes a la velocidad de aprendizaje en el uso del sistema. En segundo lugar, se les solicita también responder un cuestionario sobre su percepción acerca de las extensiones realizadas al sistema.

6.2.1. Participantes en la evaluación

Para la evaluación se contactó a tres usuarios expertos del sistema, con diversos grados de conocimiento y uso del mismo. El perfil de cada usuario se puede encontrar en la tabla 6.1.

	U1	U2	U3
Hace cuánto tiempo utiliza la aplicación	3 años	2 años	1 año
Con qué frecuencia utiliza la aplicación semanalmente	1 hora	2 horas	2 horas

Tabla 6.1: Perfil de los usuarios expertos

6.2.2. Tareas de familiarización

Para la ejecución de las tareas, los usuarios son citados a una videoconferencia en la cual se les otorga una cuenta de acceso a la aplicación extendida. En este punto los usuarios deberán compartir su pantalla y realizar la siguiente lista de actividades:

1. Iniciar sesión con la cuenta entregada.
2. Crear un nuevo tablero de producto usando el layout "Procesamiento de datos".
3. Crear un post-it en la sección de concepto.
4. Marcar el post-it como una adición en desarrollo.
5. Crear un post-it en la sección de generación de indicadores.
6. Marcar el último post-it creado como completado.
7. Filtrar los post-its para solo visualizar el as-is. Luego, volver a ver todos.
8. Crear un segundo hito de desarrollo.
9. Crear un tercer hito de desarrollo.
10. En el segundo hito, mover el primer post-it de "en desarrollo" a "en revisión".
11. En el tercer hito, marcar el primer post-it de "en revisión" como completado.
12. Visualizar el as-is del tercer hito, luego el del segundo.
13. Modificar la descripción del proyecto actual.

Esta sesión de evaluación es grabada[2], para su posterior análisis. Al analizar, se extraen dos métricas: número de *clicks* realizados y tiempo requerido para completar cada tarea (sin considerar el tiempo de tipeo).

[2]Con el debido consentimiento de los involucrados.

6.2.3. Descripción del cuestionario utilizado

Luego de la realización de tareas, se entrega a los usuarios un instrumento tipo cuestionario con afirmaciones sobre las cuales el usuario debe indicar su nivel de acuerdo o desacuerdo usando una escala Likert de 5 puntos, donde el 1 significa "Totalmente en desacuerdo" y el 5 significa "Totalmente de acuerdo".

El instrumento utilizado es una versión extendida del cuestionario *SUS* (*System Usability Scale*) traducido al español, el cual fue complementado con afirmaciones orientadas a la medición de utilidad, inspiradas en el cuestionario descrito en el reporte "Questionnaire Measuring the Utility of Knowledge-Based Systems" [13].

Afirmaciones sobre usabilidad

Las afirmaciones que conforman el cuestionario SUS traducido son las siguientes:

1. Creo que me gustaría utilizar este sistema frecuentemente.
2. Encontré el sistema innecesariamente complejo.
3. Creo que el sistema es fácil de usar.
4. Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar este sistema.
5. Encontré que las funciones de este sistema estaban bien integradas.
6. Creo que habían muchas inconsistencias en este sistema.
7. Creo que la mayoría de personas aprenderían a utilizar este sistema muy rápidamente.
8. Encontré que el sistema es muy engorroso de utilizar.
9. Sentí confianza y seguridad en lo que hice al utilizar el sistema.
10. Necesité aprender muchas cosas antes de poder empezar a utilizar el sistema.

Para calcular el rendimiento del sistema en esta escala se deben sumar los puntos asociados a cada afirmación y multiplicar el total por un factor de 2,5. Los puntos de cada afirmación se obtienen de la siguiente forma:

- Para afirmaciones impares se debe restar el valor 1 a la respuesta.
- Para afirmaciones pares se debe obtener la diferencia entre la respuesta y el valor 5.

El resultado obtenido será un número del 1 al 100, y según un estudio estadístico realizado por Dr. Jeff Sauro [17] sobre 500 sistemas evaluados, los deciles se distribuyen aproximadamente según la siguiente lista:

1. Entre 0 y 47 puntos.
2. Entre 48 y 55 puntos.
3. Entre 56 y 60 puntos.
4. Entre 61 y 65 puntos.
5. Entre 66 y 68 puntos.
6. Entre 69 y 71 puntos.
7. Entre 71 y 73 puntos.
8. Entre 74 y 78 puntos.
9. Entre 79 y 81 puntos.
10. Entre 82 y 100 puntos.

Aquí observamos que obteniendo una puntuación de 68 o superior, sabemos que contamos con un sistema de calidad razonable, que se encuentra sobre el promedio.

Afirmaciones sobre utilidad

Las afirmaciones seleccionadas en referencia a la utilidad percibida del sistema son las siguientes:

1. El sistema me permite completar tareas en menos tiempo.
2. El sistema cumple con mis necesidades.
3. El sistema simplifica el flujo de información.
4. El sistema disminuye el trabajo que debo realizar.
5. El sistema me parece útil.

Para calcular el rendimiento en esta escala se tomará el valor seleccionado en cada respuesta, se le restará 1, y se sumarán. Este valor es multiplicado por un factor de 5 para obtener una puntuación final entre 0 y 100. Se considerará que el sistema cuenta con una utilidad suficiente si obtiene una puntuación superior a 68 puntos (equivalente al de la escala SUS de usabilidad).

6.2.4. Resultados obtenidos

A continuación se muestran los resultados y análisis para tanto las métricas de rendimiento, como las respuestas al cuestionario de usabilidad.

Métricas rendimiento

La primera métrica a analizar es el número de clicks necesarios para cada tarea realizada por los usuarios expertos. En la tabla 6.2 se encuentra el listado de tareas, y para cada una, los clicks requeridos por cada usuario y su promedio.

Tarea	U1	U2	U3	Promedio
Iniciar sesión con la cuenta entregada.	3	3	2	2,7
Crear un nuevo tablero de producto para la herramienta de scoping usando el layout "Procesamiento de datos".	7	6	6	6,3
Crear un post-it en la sección de concepto.	3	4	6	4,3
Marcar el post-it como una adición en desarrollo.	11	10	14	11,7
Crear un post-it en la sección de generación de indicadores.	3	4	6	4
Marcar el último post-it creado como completado.	8	8	8	8
Filtrar los post-its para solo visualizar el as-is. Luego, volver a ver todos.	2	4	2	2,7
Crear un segundo hito de desarrollo.	3	16	4	7,7
Crear un tercer hito de desarrollo.	2	3	3	2,7
En el segundo hito, mover el primer post-it de "en desarrollo" a "en revisión".	7	9	6	7.3
En el tercer hito, marcar el primer post-it de "en revisión" como completado.	8	9	8	8,3
Visualizar el as-is del tercer hito, luego el del segundo.	8	4	6	6
Modificar la descripción del proyecto actual.	5	4	4	4,3

Tabla 6.2: Número de clicks requeridos por cada usuario en cada tarea

En la tabla 6.3 se comparan los valores promedio obtenidos por los usuarios y el valor esperado en base al flujo óptimo dentro del sistema[3]. La diferencia entre estos valores se calcula en forma de porcentaje sobre el valor esperado. Por otro lado, en la tabla 6.4 observamos el tiempo que tomó cada usuario en realizar cada tarea, descontando el tiempo de tipeo[4]. También se incluye el promedio entre usuarios.

A partir de estos resultados podemos observar que las tareas que resultaron menos intuitivas a los usuarios fueron: *8. Crear un segundo hito de desarrollo* y *4. Marcar un post-it como una adición en desarrollo*. Estas tareas son precisamente las que introducen a los usuarios a las dos zonas más relevantes sobre la cual se aplicaron modificaciones: el menú desplegable lateral que contiene el manejo de proyectos e hitos, y la edición de post-its. Dada la relevancia de estas tareas, se solicitó a los usuarios realizarlas en dos instancias (las tareas 6 y 9 consisten en las mismas acciones). Al comparar la diferencia de clicks para crear hitos entre las tareas 8 y 9, vemos esta se reduce de un +119 % a un +7 %, mientras que el tiempo requerido disminuye en un 90 %. De la misma forma, en la modificación de post-its, al comparar las tareas 4 y 6, vemos que la diferencia de clicks disminuye de un +94 % a un 0 % (flujo óptimo)

[3]En algunos casos el valor esperado se considera una fracción como el óptimo promedio, ya que el sistema permite al usuario una descripción opcional, la cual aumentaría el valor óptimo en 1

[4]Ya que la velocidad de ingreso de texto no es relevante para el estudio realizado

Tarea	Esperado	Promedio	Diferencia	Diferencia Porcentual
1	3	2,7	-0,3	-11 %
2	6	6,3	0,3	+6 %
3	3,5	4,3	0,8	+24 %
4	6	11,7	5,7	+94 %
5	3,5	4	0,5	+14 %
6	8	8	0	0 %
7	2	2,7	0,7	+33 %
8	3,5	7,7	4,2	+119 %
9	2,5	2,7	0,2	+7 %
10	6	7,3	1,3	+22 %
11	8	8,3	0,3	+4 %
12	4	6	2	+50 %
13	4	4,3	0,3	+8 %

Tabla 6.3: Cálculos sobre la métrica de clicks

Tarea	U1	U2	U3	Promedio
Iniciar sesión con la cuenta entregada.	3	3,1	3,3	3,1
Crear un nuevo tablero de producto para la herramienta de scoping usando el layout "Procesamiento de datos".	13,4	9,4	4,4	9,1
Crear un post-it en la sección de concepto.	8,5	5,7	7,8	7,3
Marcar el post-it como una adición en desarrollo.	27,1	32,5	32,3	30,6
Crear un post-it en la sección de generación de indicadores.	3,2	9,2	5,2	5,9
Marcar el último post-it creado como completado.	9,7	9,1	5,8	8,2
Filtrar los post-its para solo visualizar el as-is. Luego, volver a ver todos.	7,3	12,2	5,2	8,2
Crear un segundo hito de desarrollo.	17,5	63,9	14,6	32
Crear un tercer hito de desarrollo.	2,7	3,6	3,6	3,3
En el segundo hito, mover el primer post-it de "en desarrollo" a "en revisión".	15,5	18,3	9,2	14,3
En el tercer hito, marcar el primer post-it de "en revisión" como completado.	10,6	9,4	9,6	9,8
Visualizar el as-is del tercer hito, luego el del segundo.	15,5	8	7,1	10,2
Modificar la descripción del proyecto actual.	11,6	10,3	11,4	11,1

Tabla 6.4: Segundos requerido por cada usuario en cada tarea

y el tiempo requerido disminuye en un 73 %.

Con los resultados del análisis realizado, podemos afirmar que el descubrimiento y entendimiento iniciales de las nuevas funcionalidades podría ser mejorado, ya que el sistema no es totalmente intuitivo para el usuario, pero también, podemos afirmar que al realizar la tarea una vez, este ya es capaz de mejorar su rendimiento, demostrando que la forma de uso del

sistema puede ser aprendida de manera rápida.

Resultados cuestionario

A continuación se presentan los puntajes calculados en base a las respuestas de cada usuario, según cada ítem de usabilidad y utilidad, junto al promedio por afirmación:

Afirmación	U1	U2	U3	Promedio
1	4	2	4	3,3
2	4	2	4	3,3
3	3	1	4	2,7
4	3	4	4	3,7
5	4	1	4	3
6	4	0	4	2,7
7	4	4	4	4
8	4	3	4	3,7
9	4	4	4	4
10	1	4	4	3

Tabla 6.5: Puntuación asignada a cada afirmación de usabilidad

Afirmación	U1	U2	U3	Promedio
1	2	4	4	3,3
2	3	3	3	3
3	3	4	4	3,7
4	3	4	4	3,7
5	4	4	4	4

Tabla 6.6: Puntuación asignada a cada afirmación de utilidad

Con estos resultados podemos obtener el puntaje global de la evaluación, sumando los promedios y multiplicando por los factores correspondientes. En el caso de la usabilidad, obtenemos una suma de $33,3$, obteniendo una puntuación final de $83,3$, ubicando la aplicación en el décimo decil, es decir, en el $10\,\%$ de aplicaciones con mejor rendimiento en la escala SUS. También, podemos observar que las peores puntuaciones provienen de los items 3 y 6, referentes a la facilidad de uso del sistema, y su consistencia; mientras que las mejores provienen de los items 7 y 9, referentes a la velocidad de aprendizaje y la confianza que provee la aplicación.

Por el lado de la utilidad, la suma de valores es $17,7$, lo que significa una puntuación final de $88,3$, incluso mayor a la de usabilidad; ahora bien, este instrumento no ha sido validado considerando únicamente las preguntas seleccionadas, por lo que no sabemos con exactitud la distribución de puntajes en deciles. Dicho esto, la puntuación es suficientemente alta como para visualizar que el sistema es útil para sus usuarios. La peor puntuación proviene del item

3, referente a la simplificación del flujo de información; y la mejor puntuación, se obtuvo en el item 5, que engloba la opinión del usuario respecto a la utilidad que le da el sistema en general.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Resumen del trabajo realizado

El trabajo realizado consistió en la adición de la estructura digital necesaria para que usuarios sean capaces de visualizar y manejar el as-is de un producto definido en el sistema de Tablero Digital. Esta extensión comprende lo necesario para que usuarios del sistema puedan aplicar metodologías ágiles sin la necesidad de depender de aplicaciones externas al tablero donde realizó el scoping de su producto. Observando los resultados de la evaluación realizada, esta implementación resultó útil para los usuarios, y siguió los estándares de usabilidad planificados.

Dicho esto, esta extensión consiste en un MVP que permite el manejo de los avances realizados en un producto dentro del sistema de Tablero Digital. Aún quedan varias líneas que permitirían desarrollar el concepto que fue introducido en esta memoria, lo cual es discutido en mayor detalle en la sección 7.2.

7.1.1. Aspectos a mejorar

Al hablar de los aspectos a mejorar dentro de la extensión realizada podemos comenzar por los resultados de la evaluación de usabilidad, donde observamos que se puede mejorar la intuitividad del sistema, objetivo que se cumple acercando la implementación del tablero a las implementaciones de otros sistemas, para así permitir al usuario seguir trabajando de la misma forma en que ya lo hace.

Por otro lado, se pueden implementar tests para ejecutar la evaluación de corretitud de manera automática, y así asegurar que futuros desarrollos no rompan las funcionalidades previas, especialmente en el contexto del tablero digital donde trabajan diversos equipos de desarrollo.

7.2. Trabajo futuro

Como continuación a la extensión descrita en esta memoria, se pueden realizar diversas adiciones que ayudarán a los usuarios a utilizar el tablero digital como principal y única herramienta para el manejo del avance del desarrollo de un producto. Entre ellos se incluyen:

- *Configuración de estados*: Permitir a los usuarios definir los estados que seguirán los post-its, además de los flujos permitidos entre dichos estados.
- *Visualización separada estilo tablero Kanban*: Integrar una visualización separada con columnas por cada estado, donde los post-its pueden ser desplazado para modificar su estado.
- *Visualización separada de backlog*: Integrar una visualización separada que permita observar todos los post-its y sus estados para cada hito de desarrollo.
- *Campos personalizables*: Permitir a los usuarios definir pares llave/atributo para asociar a sus post-its.
- *Distinción entre post-it y ticket*: Permitir al usuario generar múltiples tickets a partir de un post-it para utilizar en las visualizaciones del proceso Kanban. De este modo se permitiría al usuario aumentar la granularidad en sus actividades Scrum sin aumentar la complejidad del tablero, manteniendo la visualización del tablero entendible por el cliente.

Por otro lado, se considera que para continuar el desarrollo del tablero de manera correcta y eficiente, es necesario revisar la implementación existente para normalizar su estructura. Particularmente, al trabajar en el desarrollo de la extensión se observó la necesidad de las siguientes mejoras en el área front-end:

- *Definición de colores*: Normalizar el uso de color definiendo una cantidad de variables reducidas para todo el sistema.
- *Generación de librería de componentes*: Crear componentes reutilizables para reducir la duplicación de código y mejorar la separación de lógica en el sistema.

Bibliografía

[1] Inc. Amazon.com. Aws. https://aws.amazon.com/, 2006. Accessed: 2022-07-12.

[2] Atlassian. Jira software. https://www.atlassian.com/software/jira, 2002. Accessed: 2022-07-12.

[3] Atlassian. Trello. https://trello.com/, 2011. Accessed: 2022-07-12.

[4] Automattic. Socket.io. https://socket.io/, 2021. Accessed: 2022-07-12.

[5] Oracle Corporation. Mysql. https://www.mysql.com/, 1995. Accessed: 2022-07-12.

[6] Encode. Django rest framework. https://github.com/encode/django-rest-framework, 2017. Accessed: 2022-07-12.

[7] Django Software Foundation. django. https://www.djangoproject.com/, 2005. Accessed: 2022-07-12.

[8] OpenJS Foundation. nodejs. https://nodejs.org/, 2009. Accessed: 2022-07-12.

[9] Python Software Foundation. python. https://www.python.org/, 1991. Accessed: 2022-07-12.

[10] Javier Gómez. Desarrollo de un tablero digital de apoyo al levantamiento de requisitos en la etapa de preventa de proyectos de software. In *Memoria de Ingeniería Civil en Computación. Departamento de Ciencias de la Computación, Universidad de Chile*, 2021.

[11] et al Hampton Catlin. Sass. https://sass-lang.com/, 2006. Accessed: 2022-07-12.

[12] Shore Labs. kanban tool. https://kanbantool.com/, 2009. Accessed: 2022-07-12.

[13] Sharon L. Riedel Ann P. Trent Leonard Adelman, James Gualtieri. In *Questionnaire Measuring the Utility of Knowledge-Based Systems*, pages A6–A9, 1996.

[14] Inc. Notion Labs. Notion. https://www.notion.so/, 2016. Accessed: 2022-07-12.

[15] Brendan Eich of Netscape. Javascript. https://developer.mozilla.org/en-US/docs/Web/JavaScript, 1995. Accessed: 2022-07-12.

[16] et al Pooya Parsa. Bootstrapvue. https://github.com/bootstrap-vue, 2016. Accessed: 2022-07-12.

[17] Dr. Jeff Sauro. Measuring usability with the system usability scale (sus). https://measuringu.com/sus/, 2011. Accessed: 2022-07-12.

[18] et al TJ Holowaychuk, StrongLoop. Express. https://expressjs.com/, 2017. Accessed: 2022-07-12.

[19] Tomas Vera, Daniel Perovich, and Sergio F. Ochoa. An instrument to define the product scope at preselling time. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 604–608, 2021.

[20] Evan You. Vue.js. https://vuejs.org/, 2014. Accessed: 2022-07-12.