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

DESARROLLO DE UN TABLERO DIGITAL DE APOYO AL LEVANTAMIENTO DE REQUISITOS EN LA ETAPA DE PREVENTA DE PROYECTOS DE SOFTWARE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

JAVIER IGNACIO GÓMEZ PEÑA

PROFESOR GUÍA:
SERGIO OCHOA DELORENZI

MIEMBROS DE LA COMISIÓN:
DANIEL PEROVICH GEROSA
CECILIA BASTARRICA PIÑEYRO
JOSÉ BENGURIA DONOSO

SANTIAGO DE CHILE
2021

Resumen

En el contexto de la preventa de un proyecto de software, se ven involucradas dos partes principales: los representantes del cliente y los del desarrollador. Para que el proceso se lleve a cabo de la buena manera, es importante que ambas partes se encuentren en concordancia respecto a qué es lo que el cliente quiere y qué es lo que va a incluir la propuesta del desarrollador. Durante esta etapa, ambas partes realizan reuniones para reducir la incertidumbre y acotar el alcance del proyecto a cotizar. Esta actividad típicamente se realiza de manera presencial, a veces utilizando un tablero físico, donde los participantes añaden y acuerdan tarjetas (post-its) asociadas a conceptos o funcionalidades que debe incluir el sistema a presupuestar.

Este tablero se vuelve fundamental a la hora de definir el alcance, ya que permite tener una visión global del proyecto y sus límites. Sin embargo, usar un tablero físico tiene diversas limitaciones, por ejemplo, cuando esta actividad se realiza de manera remota, o cuando se necesita acceder al tablero de manera offline (fuera de la reunión). A partir de estos escenarios nace la necesidad de contar con un tablero digital, que sirva de apoyo para la etapa de preventa de proyectos, y que permita a las partes contar con acceso distribuido al contenido del tablero.

Para dar respuesta a esta necesidad, el objetivo de este trabajo de memoria es desarrollar una herramienta que permita a los distintos usuarios (es decir, a los representantes del cliente y del desarrollador) colaborar en tiempo real, a través de un espacio virtual, en donde cada uno es capaz de añadir sus conceptos o funcionalidades del sistema para luego discutirlas con el resto de los involucrados en el proyecto.

La herramienta desarrollada es un tablero digital, dividido en secciones, en las cuales los usuarios pueden añadir tarjetas con información a su gusto, además de moverlas entre las distintas secciones. Debido a que la herramienta permite la colaboración en tiempo real, cuando un usuario realiza un cambio sobre el tablero, éste se propaga instantáneamente al resto de los participantes de la sesión. De esa manera, todos los usuarios ven la misma información en un determinado momento. Por otro lado, los usuarios también son capaces de modificar la información añadida por otro usuario, y así pueden corregir o eliminar la información errónea o no precisa existente en el tablero.

El tablero digital fue evaluado en términos de la correctitud de las operaciones que realiza, así como de la usabilidad y utilidad percibida por los usuarios. Los resultados muestran que las operaciones hechas sobre el tablero están correctamente implementadas, tanto en lo que respecta al uso individual o al uso colaborativo de los tableros. Respecto a la utilidad de la herramienta, ésta fue calificada como útil para apoyar la etapa de definición del alcance de un proyecto. Sin embargo, los evaluadores hicieron algunas observaciones sobre la usabilidad de ésta, que afectan a su utilidad. A pesar de los comentarios, la herramienta es percibida como usable.

A mi familia

Tabla de Contenido

1. Introducción

La preventa de proyectos de software es fundamental para la sustentabilidad de la amplia mayoría de las empresas [1]. Si bien esta etapa representa una oportunidad para obtener nuevos contratos, también representa un riesgo si no se realiza apropiadamente, pues podría llevar a entregar presupuestos por debajo del esfuerzo realmente requerido, lo cual significa pérdidas económicas (y probablemente de prestigio) para el desarrollador.

Desde el punto de vista de la empresa desarrolladora, la "preventa" de proyectos va desde la recepción del requerimiento del potencial cliente, hasta el momento en que este último indica si acepta o no la propuesta de la empresa desarrolladora [2]. Típicamente este proceso involucra tres fases (Fig. 1): 1) *definición del alcance del producto*, 2) *preparación de la propuesta*, y 3) *cotización y negociación de ésta*.

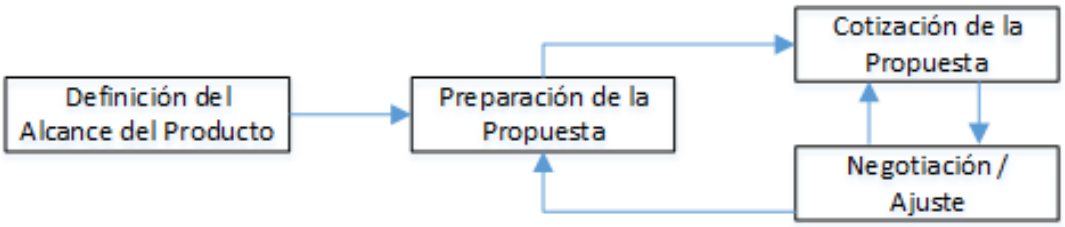

Figura 1. Proceso de preventa de un proyecto de software (obtenido de [3]).

En proyectos a la medida (también conocidos como "llave en mano"), la primera fase de la preventa (definición del alcance del producto) involucra a representantes del cliente y del desarrollador. El objetivo de esta etapa es determinar el propósito y el alcance del producto a desarrollar, y a partir de ello, establecer (entre otras cosas) el plan de proyecto, su costo y duración. Aunque todas las fases son importantes, la primera de ellas es fundamental porque las demás fases utilizan como entrada el resultado de la primera. Este trabajo de memoria aborda solo esta primera fase del proceso de preventa de un proyecto de software a la medida.

Al iniciar esta fase, no siempre hay certeza del objetivo y el alcance del producto a desarrollar. Sin embargo, es necesario conocer los límites del producto de forma temprana, para así poder realizar una buena estimación del proyecto. Es por esto que se vuelve fundamental que el alcance del producto se pueda representar de una manera que ambas partes (cliente y desarrollador) lo puedan entender, validar y ajustar. Dicha representación debe permitir alcanzar acuerdos respecto al objetivo y al alcance del producto; es decir, sus principales requisitos funcionales, no funcionales y restricciones.

1.1 Uso de tableros en la etapa de preventa de proyectos

Para intentar llegar a un acuerdo entre las partes se pueden utilizar múltiples técnicas. Una de ellas es emplear un tablero físico (similar al business canvas [4]) durante una reunión, en donde cada uno de los participantes es capaz de proponer componentes (o funcionalidades) del sistema, especificándolos en un papel adhesivo (o post-it), para luego discutirlos con toda la audiencia y eventualmente incluirlos dentro del alcance del producto, si es que hay acuerdo entre las partes.

Esta dinámica, que se da entre los representantes del cliente y del desarrollador, es llamada product scoping (o dimensionamiento del producto a desarrollar) [5]. Se ha visto que funciona bien cuando los participantes están co-localizados, es decir, reunidos en un mismo espacio físico. Sin embargo, el problema se presenta en situaciones donde no es posible realizar un encuentro presencial, impidiendo así el uso del tablero físico. La Figura 2 muestra una reunión presencial, en la que representantes del cliente y del desarrollador realizan el dimensionamiento de un sistema de procesamiento masivo de datos utilizando un tablero físico.

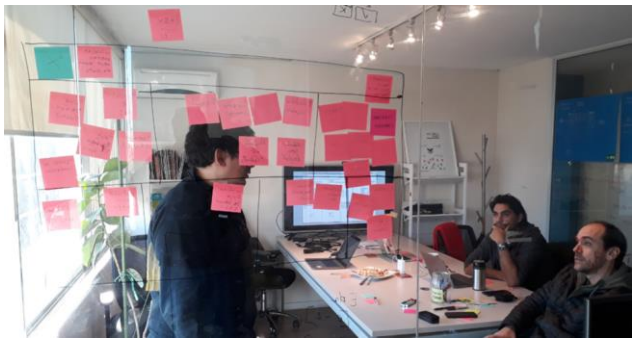

Figura 2. Sesión de scoping de un sistema de procesamiento masivo de datos

El uso de tableros físicos tiene también otras limitaciones, por ejemplo, su persistencia a lo largo del tiempo o el acceso asincrónico al contenido de ellos. Estos son problemas típicos que ya han sido reportados a partir del uso de otros tipos de tableros, como el Kanban o el Business Canvas.

Durante esta actividad de dimensionamiento (scoping), todos los tableros se utilizan de la misma manera, independientemente del tipo de sistema que se pretende desarrollar. Sin embargo, cada tipo de sistema (por ejemplo, un sistema de IoT o un sistema de información) tiene un tablero con un layout particular, y en cada sector del tablero deben indicarse componentes (a través de post-its) particulares también. La Figura 3 muestra el layout de un tablero físico (versión basada en el business canvas [4] con títulos modificados para adecuarse a los sistemas de información) que apoya el scoping de sistemas de información.

Figura 3. Tablero diseñado para abordar el scoping de sistemas de información

El layout de un tablero surge a partir de las arquitecturas de referencia que existan para cada tipo de sistema. Por lo tanto, tipos de sistemas distintos tendrán asociados tableros con un layout diferente. Al momento de iniciar este trabajo de memoria ya existen tableros físicos para realizar dinámicas de product scoping de diversos tipos de sistemas; por ejemplo, sistema de procesamiento de datos, sistemas de información, sistemas IoT y sistemas colaborativos.

1.2. Problema abordado

Si bien estos tableros han demostrado ser útiles en la práctica cuando han sido utilizados de manera presencial (tableros físicos), hasta el momento no existe una versión digital que permita la utilización distribuida y colaborativa de estos, que cuente además con todas las herramientas y características y capacidades necesarias para llevar a cabo esta etapa de manera apropiada. Particularmente, no hay un software colaborativo que permita la actualización distribuida de un tablero, por parte de los representantes del cliente y del desarrollador, durante la etapa de product scoping, en donde además los usuarios tengan la posibilidad de interactuar entre ellos. Además, la herramienta debe ser lo suficientemente flexible como para representar diversos tipos de tablero; inclusive aquellos diseñados de manera ad hoc para una cierta empresa o tipo de sistema. Este es uno de los principales desafíos abordados en este trabajo de memoria.

Otro desafío importante es permitir la participación distribuida, coordinada, y sin limitar a los participantes. Esta capacidad es requerida, no sólo en situaciones donde es difícil realizar reuniones presenciales, sino también como una forma de reducir el tiempo y esfuerzo de acordar y llevar a cabo estas reuniones. Esto haría al proceso de scoping más ágil y flexible.

El desarrollo de un tablero digital que apoye a la actividad de scoping lleva consigo diversos desafíos; por ejemplo, el tablero debe contar con un diseño gráfico a la medida, según el tipo de proyecto a abordar, y debe permitir el trabajo remoto y colaborativo entre las partes (clientes y desarrolladores).

Usando la funcionalidad del software los participantes deben poder especificar y acordar el alcance del producto a desarrollar. Además de esto, el tablero deberá contar con distintas herramientas de interacción y de participación síncrona y asíncrona. Por ejemplo, los participantes podrán sugerir nuevos post-its (o componentes) de manera asíncrona, los cuales podrán ser luego revisados durante las reuniones de trabajo posteriores.

Hoy en día existen diversas plataformas (diseñadas con otros propósitos) que ofrecen tableros o pizarras digitales, las cuales pueden actualizarse de forma colaborativa (por ejemplo, Miro [6], Cardsmith [7], y Mural [8], entre otras). El problema que presentan estas herramientas es que no ofrecen un diseño de tablero lo suficientemente flexible como para abordar el dimensionamiento y la caracterización de cualquier tipo de sistema. Por tanto, se hace necesario desarrollar una herramienta ad hoc para abordar este problema, la cual debe permitir abordar la definición del alcance y objetivos de sistemas de diversos tipos.

1.3. Desafíos técnicos del trabajo realizado

Es importante notar que una de las principales funcionalidades a desarrollar en el tablero digital, es su capacidad para soportar el uso concurrente de dicha plataforma, ya que distintos participantes modificarán el tablero de manera simultánea. Desarrollar un software con esas capacidades conlleva una complejidad no menor, debido a que las modificaciones del tablero se traducen en una actualización a la base de datos del ambiente donde está alojado. Para que los usuarios visualicen los cambios en el tablero en tiempo real, se debió idear un mecanismo para comparar constantemente el tablero local, contra el del servidor. Si existen diferencias, éstas deben propagarse hacia las aplicaciones cliente cuando corresponda.

Otro de los desafíos abordados en este trabajo de título fue el diseño flexible de los tableros, pues es necesario contar con la posibilidad de crear nuevos diseños, no simétricos (no necesariamente cuadrículas), con el fin de separar las distintas secciones en donde irán las anotaciones (post-its) de los usuarios.

Finalmente, otro de los desafíos importantes fue lograr que el proceso de definición del objetivo y alcance de un software a presupuestar (o desarrollar según sea el caso), pueda realizarse de forma

fácil para los participantes. En ese sentido, la usabilidad de la herramienta de apoyo, tanto en términos de simplicidad e intuitividad de sus interfaces, como de facilidad de acceso a los servicios que ésta provee, se volvió vital para que sea percibida como útil en la práctica.

1.4. Objetivos de la memoria

El objetivo general de este trabajo de memoria fue desarrollar una herramienta colaborativa, particularmente un tablero digital, que permita a los representantes del cliente y del desarrollador acordar el objetivo y el alcance de un producto de software a desarrollar. Los objetivos específicos definidos para alcanzar el objetivo general fueron los siguientes:

- Diseñar e implementar un sistema digital donde se permita la creación de tableros y post-its virtuales. El tablero debía ser percibido como un instrumento usable y útil para los participantes en la actividad de scoping de un producto.

- Crear un ambiente colaborativo que permita el acceso concurrente al tablero y a su contenido por parte de los usuarios (representantes del desarrollador y del cliente). El contenido de este debía quedar almacenado en un servidor, para poder ser utilizado a futuro y en distintas sesiones de trabajos. Este ambiente debía ser utilizado simultáneamente con un canal de comunicación oral (fuera del alcance de este trabajo).

1.5. Estructura del documento

En el presente documento de memoria se presenta el trabajo realizado dividido en distintos capítulos. En el primer capítulo se presentó la introducción al problema y la necesidad de crear una solución como la propuesta. Además, se presentaron los objetivos a alcanzar durante la realización de este trabajo. En el capítulo 2 se revisa el marco teórico, donde se realiza una revisión de las herramientas existentes en el mercado, y las tecnologías que podrían ser útiles para el desarrollo de una herramienta como la propuesta.

En el capítulo 3 se presenta el proceso al cual se pretende apoyar con la herramienta propuesta en este trabajo, junto con los requisitos necesarios para el desarrollo de ésta. En el siguiente capítulo se presenta el diseño del sistema, se analiza la arquitectura de la solución, los modelos de datos utilizados, y se detalla la API implementada en el backend, junto con sus endpoints.

En el capítulo 5 se describe la implementación del sistema, donde se muestran las distintas interfaces con las que interactúa un usuario. Además, se detallan las acciones que este puede realizar sobre el tablero digital. El capítulo 6 describe la evaluación de la solución por parte de usuarios reales, y los resultados obtenidos. Esta evaluación considera tanto la correctitud de la solución, como su usabilidad y utilidad. Finalmente, el capítulo 7 presenta las conclusiones obtenidas a partir del desarrollo realizado, y se proponen tareas a futuro para mejorar la herramienta desarrollada.

2. Marco teórico

A continuación, se presenta una revisión de las herramientas más conocidas para la creación de tableros digitales. Luego, se presentan las tecnologías para manejo interactivo de interfaces de usuario, las cuales podrían utilizarse para crear el tablero digital. Finalmente, se revisan las tecnologías a usar para el desarrollo del servidor de la aplicación, y se explica la preferencia en la utilización de éstas.

2.1. Herramientas para la creación de tableros

En esta sección se describen brevemente las soluciones existentes para crear tableros digitales. Además, se analizan las limitaciones que cada una posee para apoyar la actividad de scoping. Para evaluar las herramientas, se crearon tableros y se los usó en la práctica, como una forma de determinar las capacidades y limitaciones de dichas herramientas para apoyar la actividad de scoping.

2.1.1. Miro

Tal como se muestra en la Figura 4, la herramienta Miro [6] permite la creación de tableros y la edición colaborativa sobre los mismos. Sin embargo, no permite la generación del tablero a partir de un diseño personalizado. Sólo se pueden utilizar los diseños preestablecidos por dicha herramienta, o bien, la creación ‘a mano’ de un diseño. Tampoco es posible cargar un diseño desde un archivo JSON u otro similar. Desde el punto de vista de la usabilidad, las secciones del tablero no expanden su tamaño cuando una cantidad grande de post-its son colocados dentro de ellas.

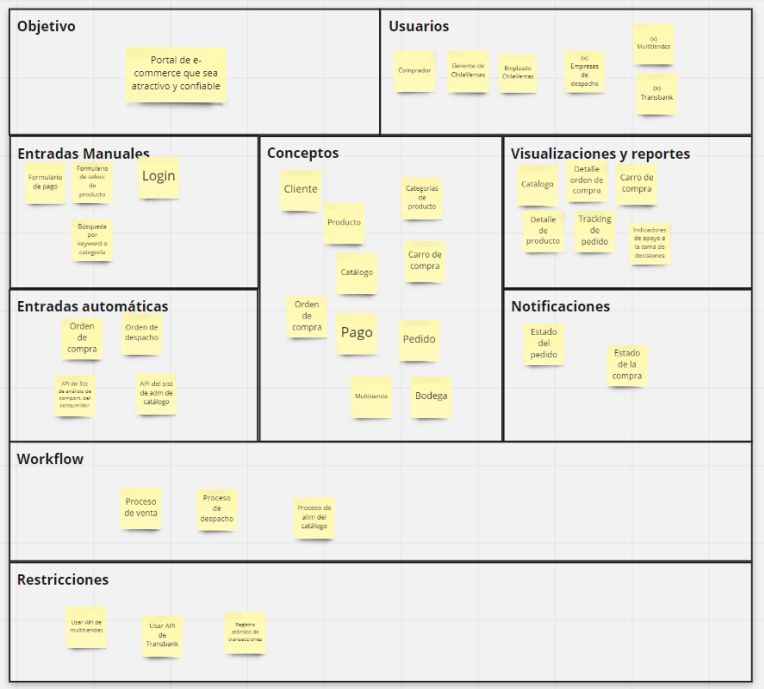

Figura 4. Ejemplo de tablero implementado con la herramienta Miro.

2.1.2. Cardsmith

La herramienta Cardsmith [7] permite la edición de tableros digitales (Figura 5), además del uso concurrente por parte de varios usuarios. También permite agregar contenido de distintos tipos (texto, imágenes, URLs, etc.) a las tarjetas. Sin embargo, sólo permite crear tableros simétricos (es decir, 'matriciales' de a x b secciones). Esto impide abordar un requisito clave de la solución a implementar, es decir, que permita generar tableros con distintos diseños (layouts). A esto último se le agrega el hecho de que no existe la posibilidad de cargar diseños desde un archivo JSON o similar.

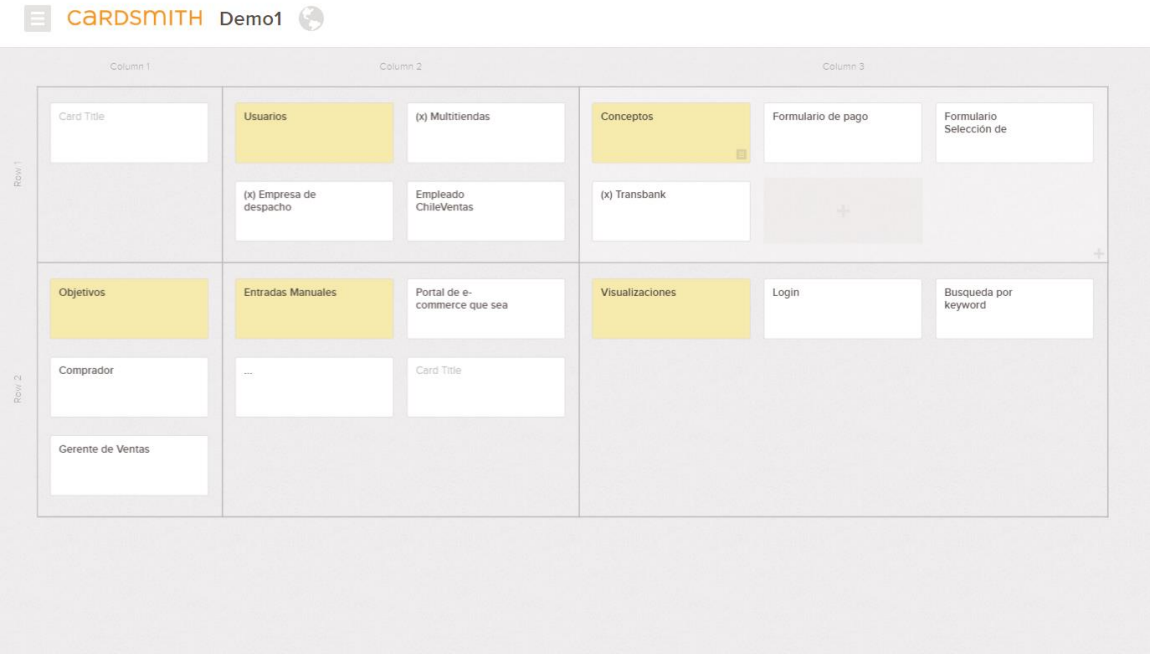

Figura 5. Ejemplo de tablero 2 x 3 implementado con la herramienta Cardsmith.

2.1.3. Mural

En la Figura 6 se muestra un tablero implementado con la herramienta Mural [8]. Si bien esta herramienta permite el uso concurrente de varios usuarios y la edición de las tarjetas del tablero, carece de características importantes para abordar la solución considerada. A pesar de que se permite crear tableros no simétricos, estos se generan a partir de diseños preestablecidos por la plataforma, por lo que no es posible personalizarlos en caso que se desee un diseño distinto.

A partir de esto se desprende que no existe la posibilidad de cargar diseños desde un archivo JSON o similar. La herramienta tampoco permite la expansión de las secciones de un tablero a medida que se agregan post-its, siendo necesario utilizar tarjetas cada vez más pequeñas (y menos visibles), y hacer uso de la función de zoom. Mural tampoco permite agregar información adicional a las tarjetas.

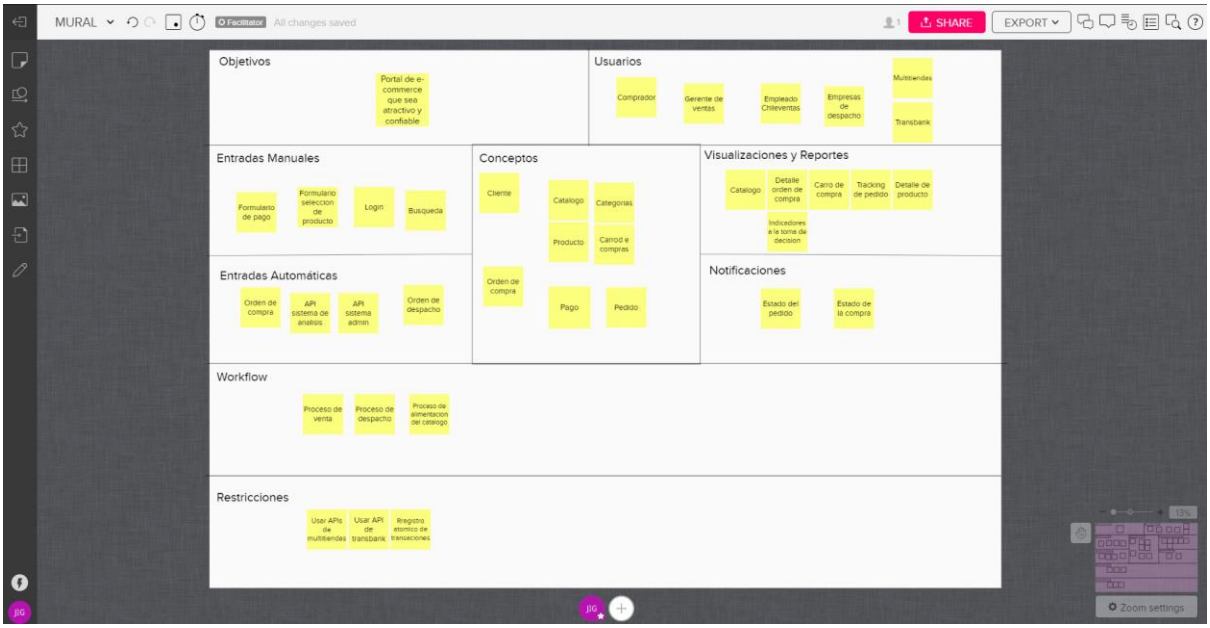

Figura 6. Ejemplo de tablero implementado con la herramienta Mural.

2.1.4 Trello

La herramienta Trello (Figura 7) permite crear tableros donde los usuarios pueden interactuar entre ellos en tiempo real, por medio de servicios integrados a la herramienta o mediante el uso de plugins. El problema que presenta esta herramienta es que sólo permite un estilo de diseño (sólo columnas). Esto dificulta agrupar visualmente tarjetas dentro de una misma categoría.

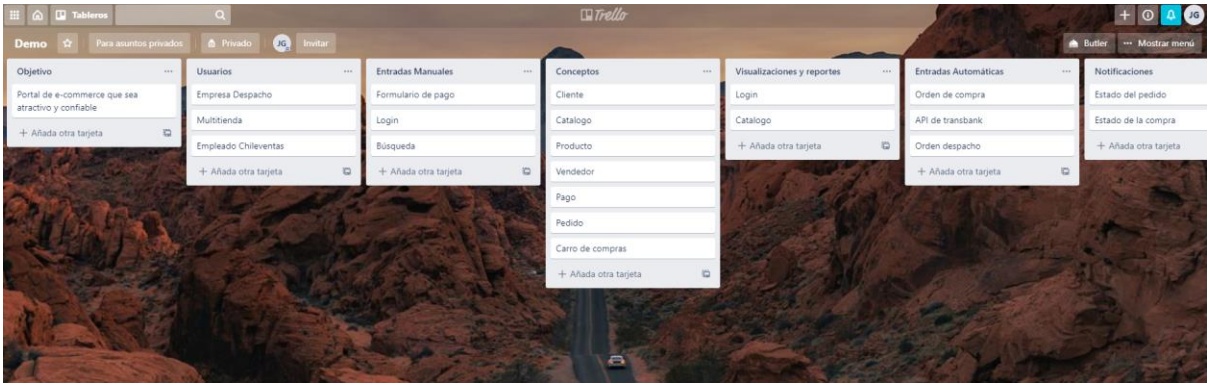

Figura 7. Ejemplo de tablero utilizando Trello

2.1.5 Comparación de herramientas

Tal como se revisó en la sección anterior, ninguna de estas cuatro herramientas cumple con el requisito de poder cargar diseños preestablecidos a partir de un archivo externo. Esto es fundamental en el caso de que se desee utilizar un diseño que no está disponible dentro de los

preestablecidos por la aplicación. Los resultados del análisis anterior se pueden ver, en forma resumida, en la Tabla 1.

Tabla 1. Funcionalidades en las herramientas revisadas

Funcionalidad \ Herramienta	Miro	Cardsmith	Mural	Trello
Permite cargar diseños personalizados a partir de archivos o estructuras externas.				
Permite generar múltiples secciones dentro del tablero y asignar tarjetas a este mismo.	✓	✓	✓	✓
Permite agregar, quitar, editar y reordenar tarjetas en las distintas secciones del tablero	✓	✓	✓	✓
Permite la transferencia de tarjetas entre las secciones que componen el tablero	✓	✓	✓	✓
Permite el acceso concurrente de usuarios al contenido de los tableros	✓	✓	✓	✓
Permite la edición del contenido del tablero, tanto la modificación del texto de las tarjetas como la eliminación de estas	✓	✓	✓	✓
Permite guardar los cambios realizados	✓	✓	✓	✓
Permite agregar información extra a las tarjetas (título, descripción, url, estado, etc.)		✓		✓

2.2. Tecnologías para el manejo interactivo de interfaces de usuario

Para la implementación de un tablero interactivo es necesario contar con librerías que permitan al usuario agregar, eliminar, editar y arrastrar tarjetas (post-its) en una interfaz web. Para esto, se listan a continuación las librerías que se consideraron para realizar un primer acercamiento a la solución, junto a algunas observaciones.

2.2.1. jQueryUI Draggable

Se tomó en cuenta la librería jQueryUI Draggable [9], la cual facilita la implementación de la funcionalidad de arrastrar y soltar las tarjetas. Se descartó utilizar esta librería en la construcción de un prototipo, puesto que era muy complejo detectar cuándo una tarjeta era soltada en una determinada sección del tablero.

2.2.2 DraggableJS

La librería DraggableJS [10] también fue considerada para la implementación del prototipo de la solución, ya que maneja contenedores (que en este caso corresponden a cada sección del tablero), y permite arrastrar y soltar elementos dentro de estos. Sin embargo, esta librería fue descartada debido a la complejidad que significaba integrarla a los múltiples contenedores que tendría el tablero (que serían creados dinámicamente), y a los datos que debían ser cargados desde un archivo externo para este prototipo.

2.2.3 GridstackJS

La librería GridstackJS [11] se tomó en cuenta debido a que tenía la posibilidad de manejar distintos contenedores, y trasladar elementos entre ellos. Esta librería está hecha para manejar tarjetas de texto, por lo que fue la elegida para implementar un primer prototipo del tablero digital (Figura 8) debido a su simplicidad para integrarla a los múltiples contenedores del tablero.

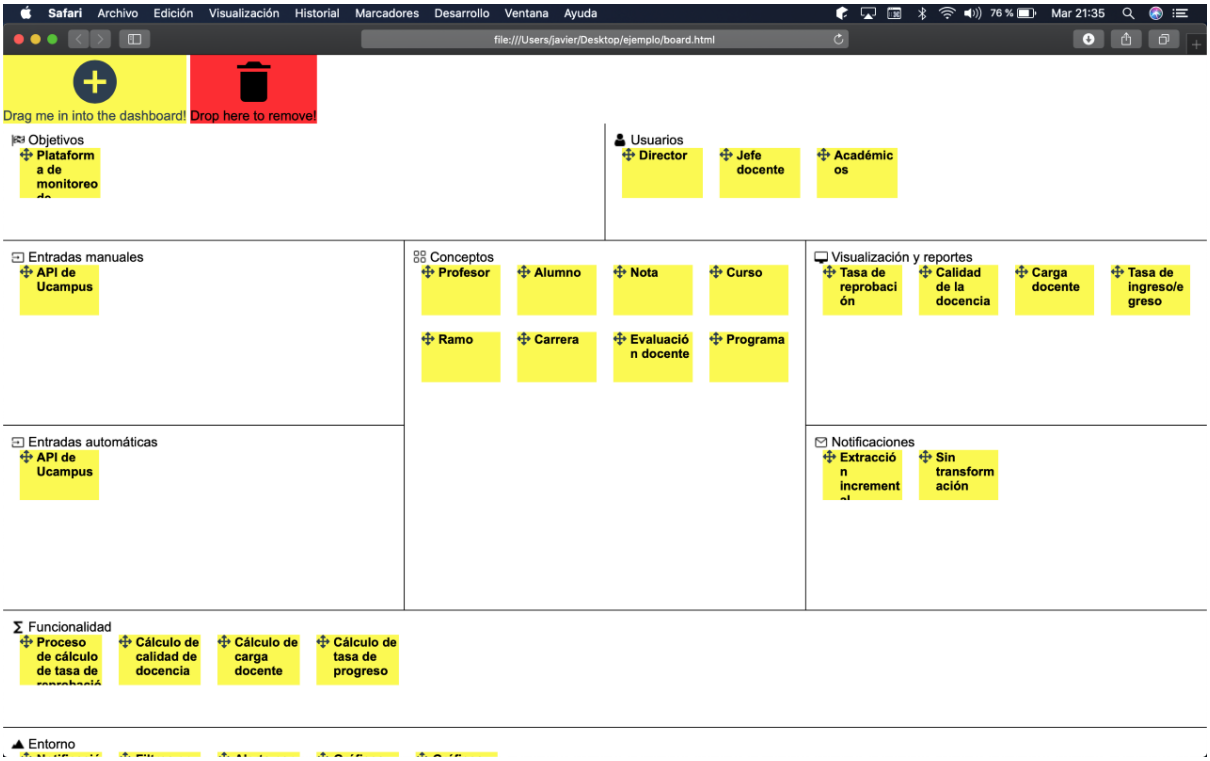

Figura 8. Prototipo de tablero digital utilizando GridstackJS.

En este prototipo se logró implementar las funcionalidades de agregar, mover, editar y eliminar tarjetas. Además, los datos y el diseño del tablero se cargan desde un archivo externo en formato JSON. Sin embargo, este primer acercamiento estaba implementado en Javascript Vanilla, por lo que se realizó un segundo prototipo implementado en Angular 10 [12]. Esta reimplementación se hizo principalmente por un tema de vigencia tecnológica, y para mejorar la mantenibilidad y capacidad de evolución de la aplicación a construir. Para esto, se consideraron dos librerías más para utilizar drag-and-drop.

2.2.4 Material Drag and Drop

Se consideró también utilizar la librería llamada Material Drag and Drop [9], puesto que se integra fácilmente con Angular 10. Ésta tiene los mecanismos para generar la disposición de las secciones a partir de un archivo JSON, y en el caso del análisis de esta herramienta, el memorista integró el módulo para ver su capacidad de manejo de post-its. Finalmente, la librería se descartó, puesto que cuando el ancho de la totalidad de las tarjetas de una sección del tablero supera el ancho del contenedor, la librería dejaba de funcionar correctamente; particularmente, esto se ponía en evidencia al momento de arrastrar una tarjeta para reordenarlas.

2.2.5 Dragula

Se analizó también la librería Dragula (ng2-dragula) [14], como una alternativa a la librería Material Drag and Drop [13]. Debido a que el prototipo anterior estaba pensado para ser desarrollado en Angular 10, se optó por la librería ng2-dragula, un wrapper para Angular de la librería Dragula. Esta librería soluciona el problema presentado cuando el ancho de las tarjetas supera al del contenedor. Así, se implementa el segundo prototipo del tablero digital, el cual se muestra en la Figura 9.

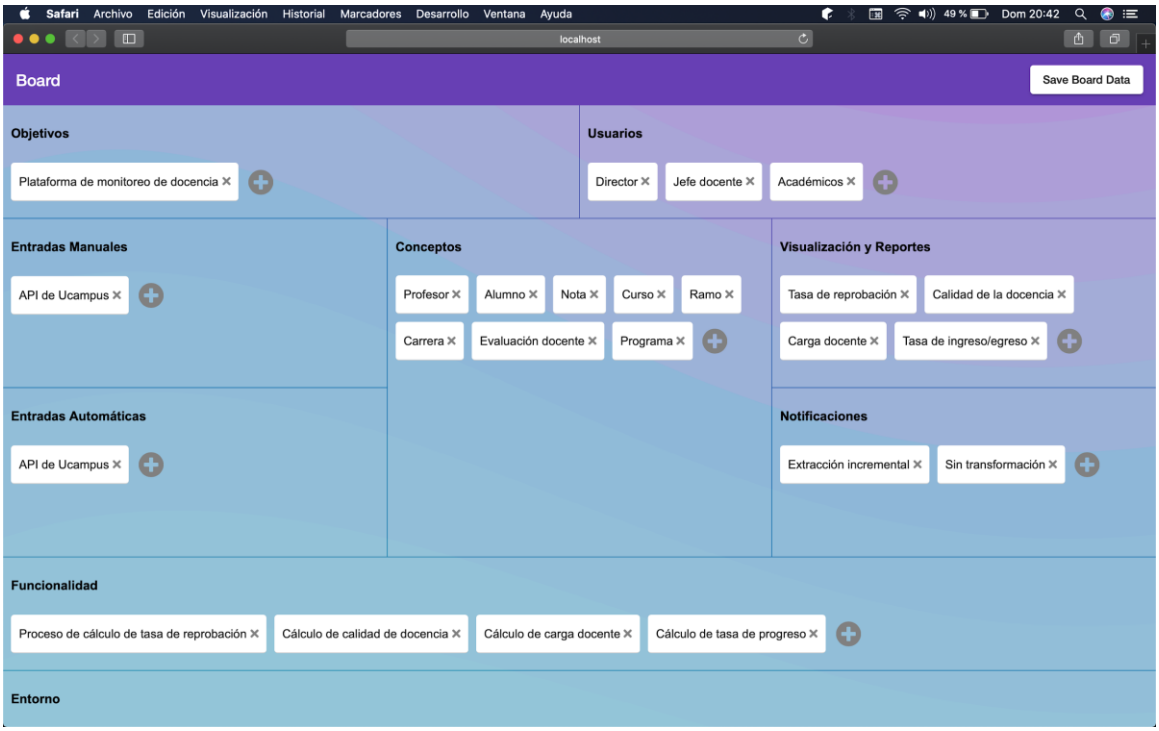

Figura 9. Segundo prototipo de tablero digital usando ng2-dragula y Angular 10.

En este segundo prototipo se mantiene la funcionalidad del primer prototipo; es decir, la carga del layout y de los datos del tablero desde un archivo o estructura externa (revisada más adelante en este documento), y la posibilidad de editar tarjetas. En este caso, eso se hizo utilizando el

framework Angular 10. Esta tecnología es más reciente, por lo tanto, tiene mayor soporte y desarrollo por parte de sus promotores.

Finalmente, la tecnología escogida para desarrollar el front-end del trabajo fue la librería Dragula debido a los favorables resultados obtenidos en el prototipo desarrollado.

2.3 Tecnología para el servidor

Para desarrollar el servidor, se escogió Express.js [15] como framework sobre la tecnología NodeJS [16]. Además de esto, se les suman a ambas tecnologías el uso de una base de datos, donde se escogió MongoDB [17] como base de datos no relacional. La preferencia sobre estas tecnologías se debe a la velocidad y simpleza que tienen éstas, además de que son frecuentemente utilizadas en conjunto, debido a que componen el stack MEAN (MongoDB Express Angular Node) [18]. Esto representa una ventaja, pues existe una amplia documentación respecto a su uso, la cual es mantenida de forma frecuente.

3. Concepción de la solución

A continuación, se describe el proceso de scoping de un proyecto, y su relevancia para este trabajo. Además, se presentan los principales requisitos de la solución, y los usuarios objetivos a los que está destinada la herramienta.

3.1. Descripción del proceso a apoyar

En la fase de definición de alcance (o product scoping) en la preventa de un proyecto de software, usualmente se utilizan tableros físicos o digitales para especificar y organizar información relevante para el cliente o el desarrollador.

En esta fase, tanto el cliente como los desarrolladores pueden agregar información en forma de tarjetas al tablero, en distintas secciones (debidamente etiquetadas) que refieren distintos componentes del proyecto, dependiendo del tipo de este último. Luego de que es ingresada toda la información, se obtiene un tablero con tarjetas organizadas de forma que se pueden ver fácilmente las tarjetas que pertenecen a una sección determinada.
El rol que cumple este trabajo es apoyar la fase de scoping, es decir la alimentación del tablero con post-its y su posterior acuerdo respecto a la relevancia o ubicación de estos. Con esto se busca facilitar el levantamiento de requerimientos del proyecto que se va a presupuestar o desarrollar.

3.2. Requisitos de la solución

A continuación, se presentan los principales requisitos funcionales y no funcionales del sistema.

- El sistema debe permitir la generación de tableros a partir de un layout descrito en archivos o estructuras externas. Esto permitirá generar múltiples diseños de tablero para abordar el scoping de diversos tipos de sistema. Esta definición de nuevos diseños se debe poder realizar tan solo cambiando los datos que describen la disposición (layout) del board.

- El sistema debe permitir la generación de múltiples secciones dentro del tablero. Esto permite que las tarjetas (post-its) puedan ser asignadas a éstas, y dispuestas en distintos contenedores para indicar distintas categorías de ellas.

- El sistema debe permitir agregar, quitar, editar y reordenar tarjetas en las distintas secciones del tablero. De esta manera, los usuarios serán capaces de agregar y clasificar las ideas que agreguen al tablero, según estimen conveniente.

- El sistema debe permitir la transferencia de tarjetas entre las secciones que componen el tablero. Así, una tarjeta puede cambiar de sección o categoría.

- El sistema debe permitir el acceso concurrente de usuarios al contenido de los tableros. Esto permitirá que todos los usuarios que se encuentren conectados al tablero simultáneamente puedan previsualizar el contenido en tiempo real.

- El sistema debe permitir la edición del contenido del tablero, tanto la modificación del texto de las tarjetas, como la eliminación de éstas. Esto permitirá que se puedan corregir errores o cambiar información de las tarjetas.

- El sistema debe permitir guardar los cambios realizados. Esto permitirá que los usuarios puedan volver a visitar el tablero, y logren visualizar el contenido que añadieron previamente.

Respecto a los requisitos no funcionales del producto, los más importantes son los siguientes:

- El sistema debe ser fácil de usar, y tener una interfaz de usuario intuitiva. Esto permitirá que la actividad de scoping pueda realizarse de manera ágil, y facilitará la participación de los involucrados.

- El sistema debe ser consistente al procesar y propagar las acciones de cada usuario, pues el acceso y edición concurrente al tablero podría producir condiciones de carrera que deben ser manejadas.

3.3. Usuarios objetivos

Los usuarios objetivos a los que está dirigido este trabajo son los integrantes del equipo del cliente y del desarrollador, los cuales participan en la fase de scoping de un proyecto de software. En la aplicación todos los usuarios tienen las mismas capacidades y permisos. Cabe destacar que usualmente un proceso de scoping, ejecutado de forma remota, es acompañado por una videoconferencia entre los participantes, en donde se coordina y acuerda la información que es ingresada al tablero.

4. Diseño del sistema

A continuación se presenta la arquitectura del sistema, el modelo de datos utilizado, la funcionalidad de soporte para diversos layouts de tablero y el backend del sistema.

4.1. Arquitectura de la solución

La estructura de la solución adhiere a un modelo de tres capas (three-tier architecture [19]); particularmente, divide al sistema en las tres capas típicas de un sistema Web: presentación, lógica de negocio, y datos. En la Figura 10 se muestra cómo interactúan las tres capas de la solución. Para la implementación de esta arquitectura se escogió el stack tecnológico MEAN (MongoDB, ExpressJS, AngularJS, NodeJS) [18].

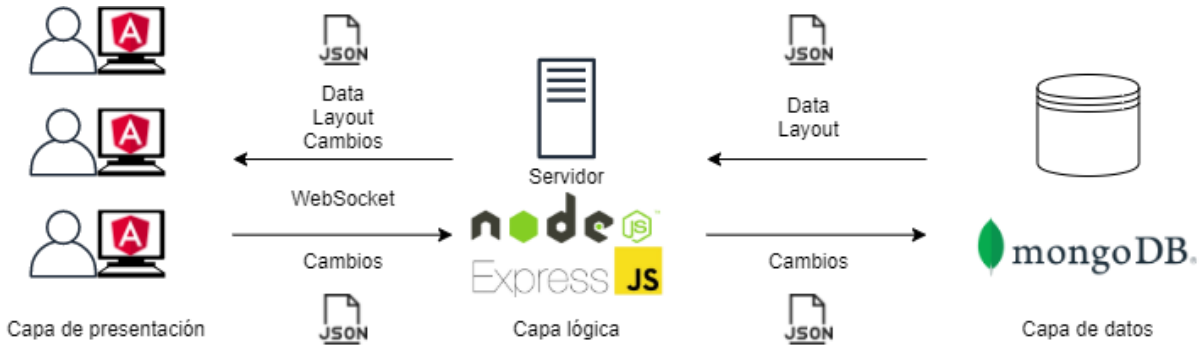

Figura 10. Diagrama de arquitectura de la solución y sus tecnologías asociadas

4.1.1 Capa de presentación

En cada una de las sesiones de trabajo, la capa de presentación recibe el diseño (“layout”) del tablero y los datos que éste contiene. Esto lo entrega la capa lógica de la herramienta (Node.js + Express.js) a través de un websocket. Además, la capa de presentación recibe los cambios que han sido reportados al servidor, por parte de otros usuarios. Cuando un usuario decide realizar un cambio en el tablero, este cambio se envía a la capa lógica, y luego es enviado al resto de los usuarios. Este es el mecanismo a través del cual cada usuario recibe los cambios realizados por otros participantes.

La tecnología que se encarga de la capa de presentación es Angular 10, la cual permite también detectar los cambios que realiza el cliente, y los envía al backend para que estos sean propagados al resto de los usuarios que están colaborando en el tablero.

4.1.2 Capa lógica

La capa lógica por su parte obtiene los diseños de los tableros, y su contenido desde la capa de datos, que corresponde a una base de datos mongoDB. Luego, envía estos datos a la capa de presentación para que sean visibles para todos los usuarios. A medida que los usuarios realizan cambios, la capa lógica recibe tales cambios en formato JSON, los cuales son utilizados para

actualizar la información del tablero en la capa de datos, y también para comunicarlos a los distintos usuarios que estén utilizando el tablero.

En la capa lógica, Node.js [16] se encarga de lo que corresponde al backend, junto a Express.js [15]. El servidor que implementa esta capa recibe los cambios realizados por los usuarios en el tablero, y los comunica al resto de los participantes. Esto último, con el fin de que todos los clientes (instancias del tablero digital central) mantengan una versión actualizada de los datos. Además, la capa lógica expone una API REST [20], mediante la cual se permite agregar nuevos layouts a la base de datos y eliminar existentes. Además, esto permite manejar la inicialización de nuevos tableros.

4.1.3 Capa de datos

La capa de datos, que se compone de una base de datos mongoDB, almacena los diseños de los tableros y también el contenido de estos. Esta capa envía estos datos a la capa lógica. Al momento de recibir una actualización, la capa de datos actualiza los documentos almacenados que correspondan, y los deja disponibles para una futura consulta.

Para la capa de datos se propone utilizar MongoDB [17], debido a que se deben almacenar tanto los diseños de los tableros, como su contenido. Estos elementos estarán representados en archivos JSON. De esta manera, se hace eficiente trabajar con esta base de datos, pues corresponde a una base de datos NoSQL, la cual almacena estructuras BSON (Binary JSON) [18]. Como su nombre lo indica, éstas están basadas en las estructuras JSON.

4.1.4 Comportamiento del sistema

El flujo habitual del sistema consiste en primer lugar, de una petición desde la capa de presentación a la API que expone la capa lógica, para así obtener los tableros disponibles para un usuario, o para crear un nuevo tablero.

Una vez que se selecciona un tablero, se abre un websocket entre el navegador del usuario y el servidor; usando ese websocket se envía el contenido del tablero. Luego, cada vez que el usuario realice un cambio en el tablero (por ejemplo, crear, borrar, mover o editar una tarjeta), se envía hacia la capa lógica un archivo JSON con un resumen de éste. El servidor actualiza el tablero en la base de datos, y envía el resumen de los cambios al resto de los usuarios conectados al tablero, los cuales actualizan su copia local.

4.2. Modelo de datos

En la capa de datos se almacenan dos tipos de documentos. Los primeros representan los diseños (layouts) de los tableros, y los segundos representan el contenido de cada tablero. Además, los cambios realizados en un tablero también son representados por una estructura de datos.

4.2.1 Documento de layout

La estructura de los documentos que definen los layouts corresponden a un objeto JSON, que cuenta con los siguientes campos:

- *categories*: Consta de un arreglo de objetos, los cuales definen cada una de las secciones o categorías del tablero. Dentro de estos objetos se considera un identificador de la sección ("id"), una etiqueta (visible para el usuario) "label", los límites de cada sección (campos que comienzan por "grid") y la altura mínima que tendrá ésta ("min-height").
- *_id*: Corresponde al identificador del layout dentro de la base de datos. Este campo es fundamental al momento de registrar el layout correspondiente en el objeto de un tablero.
- *name*: Nombre del layout. Éste será mostrado al usuario al momento de elegir un determinado layout.
- *grid*: Consiste en un objeto con un campo que indica la cantidad de columnas que tendrá la grilla del layout. La sintaxis va de acuerdo con el parámetro *grid-template-columns* [22] perteneciente al sistema CSS Grid [23].

A continuación se muestra un extracto del mismo:

```
{
    "categories": [
      {
        "id": String,
        "label": String,
        "grid-column-start": String,
        "grid-column-end": String,
        "grid-row-start": String,
        "grid-row-end": String,
        "min-height": String
      },
      ...
    ],
    "_id": String,
    "name": String,
    "grid": {
      "grid-template-columns": String
    },
  }
```

4.2.2 Documento de tablero

En el caso del documento que representa a los tableros, éste contiene los siguientes atributos:

- *users*: Consta de un arreglo de strings, el cual define los usuarios que tienen acceso al tablero en cuestión.

- *_id*: Corresponde al identificador del tablero en la base de datos. Este campo es fundamental para el acceso al tablero.
- *name*: Indica el nombre del tablero. Este valor se muestra a los usuarios y es modificable desde la interfaz de usuario.
- *layout*: Indica el identificador del layout correspondiente para el tablero. Se obtiene desde la estructura de layout revisada anteriormente.
- *data*: Corresponde a la estructura que guarda el contenido del tablero. Ésta consta de un campo *cards*, que corresponde a un diccionario, en donde se almacenan las tarjetas del tablero. La llave corresponde al identificador de cada tarjeta y su valor es un objeto que contiene el texto o título de la tarjeta (campo *text*) y la descripción correspondiente a ella (campo *description*).
- *sections*: La estructura de un tablero también contiene un campo *sections*, que consta de un diccionario en donde la llave es el identificador de cada sección del tablero (proveniente de la estructura de layout), y el valor es un arreglo de strings, en donde cada cadena corresponde al identificador de una tarjeta.

A continuación se muestra un extracto de la estructura que representa a los tableros:

```
{
    "users": [
      String
    ],
    "_id": String,
    "name": String,
    "layout": String,
    "data": {
      "cards": {
        <String>: {
          "text": String,
          "description": String
        },
        ...
      },
      "sections": {
        <String>: [
          String
        ],
       ...
      }
    },
  }
```

4.2.3 Cambios en el tablero

Adicionalmente, en el sistema interactúan otras estructuras, las cuales no son almacenadas en la base de datos, por lo tanto, no constituyen un documento. Se trata de las estructuras que representan los cambios, las que son intercambiadas entre el navegador del usuario y el servidor. Éstas son

llamadas “acciones” o *actions*. A continuación se revisan cada una de las distintas estructuras posibles para las acciones:

Mover una tarjeta. Al mover una tarjeta en el tablero, se envía al servidor una estructura representando los cambios. Ésta consta de un campo *type* que indica el tipo de acción que se ha realizado. Los campos de *sourceSection* y *targetSection* indican el identificador de las secciones de origen y de destino respectivamente. El campo *sourceIndex* indica el índice de la tarjeta en la sección de origen. El campo *targetIndex* a su vez, indica el índice de la tarjeta en la sección de destino. A continuación se muestra la estructura de la información intercambiada para notificar una acción de este tipo.

```
{
  type: 'MOVE',
  sourceSection: String,
  sourceIndex: Integer,
  targetSection: String,
  targetIndex: Integer,
}
```

Agregar una tarjeta nueva. Al crear una nueva tarjeta, la estructura enviada al servidor consta de un campo *type* indicando la acción realizada, un campo *section* que refiere al identificador de la sección donde se ha añadido una tarjeta, y un campo *card* que contiene el identificador y contenido de la tarjeta recién creada. A continuación se muestra la estructura de la información que comunica una acción de este tipo.

```
{
  type: 'CREATE',
  section: String,
  card: {
    id: String,
    text: String,
    description: ''
  }
}
```

Eliminar una tarjeta. Al eliminar una tarjeta, se envía una estructura con los parámetros *type*, *section* e *id*. *type* indica el tipo de acción, *section* corresponde al identificador de la sección desde donde se eliminó la tarjeta e *id* refiere al identificador de la tarjeta eliminada. A continuación se muestra la estructura usada para comunicar una acción de este tipo.

```
{
  type: 'DELETE',
  section: String,
```

```
  id: String
}
```

Editar una tarjeta. Cuando un usuario edita el contenido de una tarjeta, se envía una estructura con los campos *type*, *cardId*, *field* y *data*. El primero hace referencia al tipo de acción realizada. El campo *cardId* es el identificador de la tarjeta editada, y *field* indica el campo de la tarjeta que fue editado (título o descripción). Finalmente, el campo *data* es el nuevo valor del campo editado. Similar a los casos anteriores, a continuación se muestra la estructura usada para informar este tipo de acción.

```
{
  type: 'UPDATE_CARD',
  cardId: String,
  field: String,
  data: String
}
```

Añadir un usuario al tablero. Para otorgarle acceso a un usuario a un tablero, se debe añadir a la lista de usuarios registrados. Para esto, el navegador envía una estructura con dos campos. El campo *type* indica el tipo de acción y el campo *email* corresponde a la dirección de correo electrónico del usuario añadido. La estructura usada es la siguiente:

```
{
  type: 'ADD_USER',
  email: String
}
```

Cambiar el nombre del tablero. Al editar el nombre del tablero, se envía una estructura con los campos *type* y *value*. El primero indica el tipo de acción y el segundo el nuevo valor para el nombre del tablero. La estructura usada es la siguiente:

```
{
  type: 'NAME',
  value: String,
}
```

Todas las estructuras de datos revisadas anteriormente son transferidas mediante websockets, desde el navegador del usuario hasta el servidor. Luego, el servidor realiza los cambios correspondientes frente a la base de datos, y envía la acción al resto de usuarios conectados al tablero. Esto se hace con el fin de que actualicen su versión local del tablero y se mantenga la consistencia de la información (contenido del mismo).

4.3. Soporte para diversos layouts del tablero

En la sección 3.2 del documento se habló del requisito de soportar distintos diseños o layouts para los tableros. Esto es posible gracias a los documentos de layout presentados en la sección 4.2.1.

Estos documentos incluyen la información que define el diseño del tablero, la cual se define al momento de su creación, cuando se selecciona el diseño deseado. Sin embargo, la variedad de layouts disponibles es limitada, y está sujeta a los diseños existentes en la base de datos.

Para cumplir el objetivo a cabalidad, se debe poder añadir nuevos tipos de diseño. Es por esto que existe un endpoint en la API que expone el servidor, en donde el administrador del sistema puede añadir y eliminar diseños de tablero a su gusto, mediante un request POST al endpoint correspondiente. De esta manera, el o los usuarios con acceso a la API pueden crear diseños personalizados para utilizar con sus tableros. Esta funcionalidad puede quedar también disponible para cualquier usuario, si es que se desea dejar expuesto el endpoint; o sea, visible de manera pública.

4.4 API backend

Tal como se mencionó anteriormente, el servidor expone una API mediante la cual se permite agregar o eliminar diseños de tablero. Además, la API permite la inicialización de nuevos tableros, por ejemplo, entregando una lista de los distintos layouts disponibles, o listando los tableros disponibles para un usuario determinado. A continuación se detalla cada uno de los endpoints expuestos por la API.

4.4.1 /layouts

Ruta: https://scoping.dcc.uchile.cl/api/layouts

- Método GET:
  Lista todos los layouts disponibles en la base de datos.

  Respuesta de ejemplo:


```
[
  {
    "categories": [
      {
        "id": "32cab424-6ba7-4fb2-986a-bdc3bff3b4bc",
        "label": "Observaciones",
        "grid-column-start": "2",
        "grid-column-end": "3",
        "grid-row-start": "2",
        "grid-row-end": "4",
        "min-height": "400px"
```

```
      },
    ],
    "_id": "493a994f-a1fb-4f6f-8616-30b3093853fd",
    "name": "Layout 1",
    "grid": {
      "grid-template-columns": "repeat(6, 1fr)"
    },
    "__v": 0
  },
  ...
]
```

- Método POST:
  Añade a la base de datos un objeto de layout.

  Parámetros:
  - 'name': String. Define el nombre del layout.
  - 'grid': Objeto. Define la cantidad de columnas del layouts.
  - 'categories': Objeto. Define cada una de las secciones del tablero (nombre, id y tamaño)

  Petición de ejemplo:

```
{
    "categories": [
      {
        "id": "goals",
        "label": "Objetivos",
        "grid-column-start": "1",
        "grid-column-end": "4",
        "grid-row-start": "1",
        "grid-row-end": "2",
        "min-height": "100px"
      },
    ],
    "name": "Layout 1",
    "grid": {
      "grid-template-columns": "repeat(6, 1fr)"
    }
  }
```

  Respuesta:

  200:

```
{
  "status": "Layout saved"
```

```
}
```

Error: Responde el error generado

4.4.2 /layouts/{id}

Ruta: https://scoping.dcc.uchile.cl/api/layouts/{id}

- Método GET:
  Responde el objeto Layout correspondiente al parámetro ‘id’ entregado en la ruta del endpoint.

  Respuesta de ejemplo:

```
{
    "categories": [
      {
        "id": "2c373103-e754-4e19-a2cd-f5f4079cb3a7",
        "label": "Notas",
        "grid-column-start": "1",
        "grid-column-end": "3",
        "grid-row-start": "2",
        "grid-row-end": "4",
        "min-height": "200px"
      },
    ],
    "_id": "2e84f782-33e7-4223-8420-28d8a114085c",
    "name": "Layout 1",
    "grid": {
      "grid-template-columns": "repeat(6, 1fr)"
    },
    "__v": 0
  }
```

- Método DELETE:
  Elimina el layout correspondiente al parámetro ‘id’ entregado en la ruta.

  Respuesta:

  200:

```
{
  "status": "Layout deleted"
}
```

Error: Responde el error generado

4.4.3 /boards

Ruta: https://scoping.dcc.uchile.cl/api/boards

- Método POST:
  Añade a la base de datos un tablero.

  Parámetros:

  - ‘name’: String. Define el nombre del tablero.
  - ‘layout’: String. Define el layout que se debe utilizar en conjunto con el tablero. Corresponde al ObjectId del layout en la base de datos.
  - ‘users’: Array. Lista los usuarios que tienen acceso al tablero.
  - ‘data’: Objeto. Contiene dos objetos ‘cards’ y ‘sections’ que contienen objetos de cada tarjeta y las secciones con tarjetas, respectivamente.

  Petición de ejemplo:

```
[
  {
    "users": [
      "user@domain.com"
    ],
    "name": "Test Board",
    "layout": "5fda9dd5f0a77316ea32633d",
    "data": {
      "cards": {
        "b92cb31d-33c9-17c1-fe2d-fea767a94f2f": {
          "text": "This is a card",
          "description": "Very important card"
        },
      },
      "sections": {
        "goals": [
          "b92cb31d-33c9-17c1-fe2d-fea767a94f2f",
        ],
      }
    },
  },
]
```

  Respuesta:

  200:

```
{
  "status": "Board saved"
}
```

Error: Responde el error generado

4.4.4 /boards/{id}

Ruta: https://scoping.dcc.uchile.cl/api/boards/{id}

- Método GET:
  Responde el objeto Board correspondiente al parámetro ‘id’ entregado en la ruta del endpoint.

  Respuesta de ejemplo:

```
{
   "users": [
     "user@domain.com",
     ...
   ],
   "_id": "5fda9de7f0a77316ea32633e",
   "name": "Test Board",
   "layout": "5fda9dd5f0a77316ea32633d",
   "data": {
     "cards": {
       "b92cb31d-33c9-17c1-fe2d-fea767a94f2f": {
         "text": "This is a card",
         "description": "Very important card"
       },
     },
     "sections": {
       "goals": [
         "b92cb31d-33c9-17c1-fe2d-fea767a94f2f",
       ],
     }
   },
  "__v": 0
 }
```

- Método DELETE:
  Elimina el tablero correspondiente al parámetro ‘id’ entregado en la ruta.

  Respuesta:

  200:

```
{
  "status": "Board deleted"
}
```

Error: Responde el error generado

4.4.5 /boards/user/{user}

Ruta: https://scoping.dcc.uchile.cl/api/boards/user/{user}

- Método GET:
  Responde un arreglo de todos los tableros a los cuales tiene acceso el usuario indicado en el parámetro {user} en la ruta.

  Respuesta de ejemplo:

```
[
  {
    "users": [
      "user@domain.com",
      ...
    ],
    "_id": "5fda9de7f0a77316ea32633e",
    "name": "Test Board",
    "layout": "5fda9dd5f0a77316ea32633d",
    "data": {
      "cards": {
        "b92cb31d-33c9-17c1-fe2d-fea767a94f2f": {
          "text": "This is a card",
          "description": "Very important card"
        },
      },
      "sections": {
        "goals": [
          "b92cb31d-33c9-17c1-fe2d-fea767a94f2f",
        ],
      }
    },
   "__v": 0
  },
  ...
]
```

Los endpoints detallados anteriormente son los que constituyen la API del servidor, mediante la cual se pueden agregar y eliminar diseños, además de permitir que el frontend cree nuevos tableros en la base de datos cuando el usuario decide hacerlo a través de la interfaz.

5. Implementación de la solución

Como se mencionó en el capítulo anterior, la solución se implementó usando una arquitectura de tres capas, separando las capas de presentación, lógica de negocio y datos. En este capítulo se detallan las vistas del usuario frente a la aplicación (capa de presentación).

5.1 Landing Page

La primera vista que se presenta al usuario es una pantalla de bienvenida, junto con un botón de inicio de sesión. En este punto, el usuario debe iniciar sesión con sus credenciales de una cuenta Google [24] (Fig. 11). El uso de Google como proveedor de identidad fue una decisión de diseño tomada meramente por simpleza. Es posible integrar otras plataformas como proveedores de identidad.

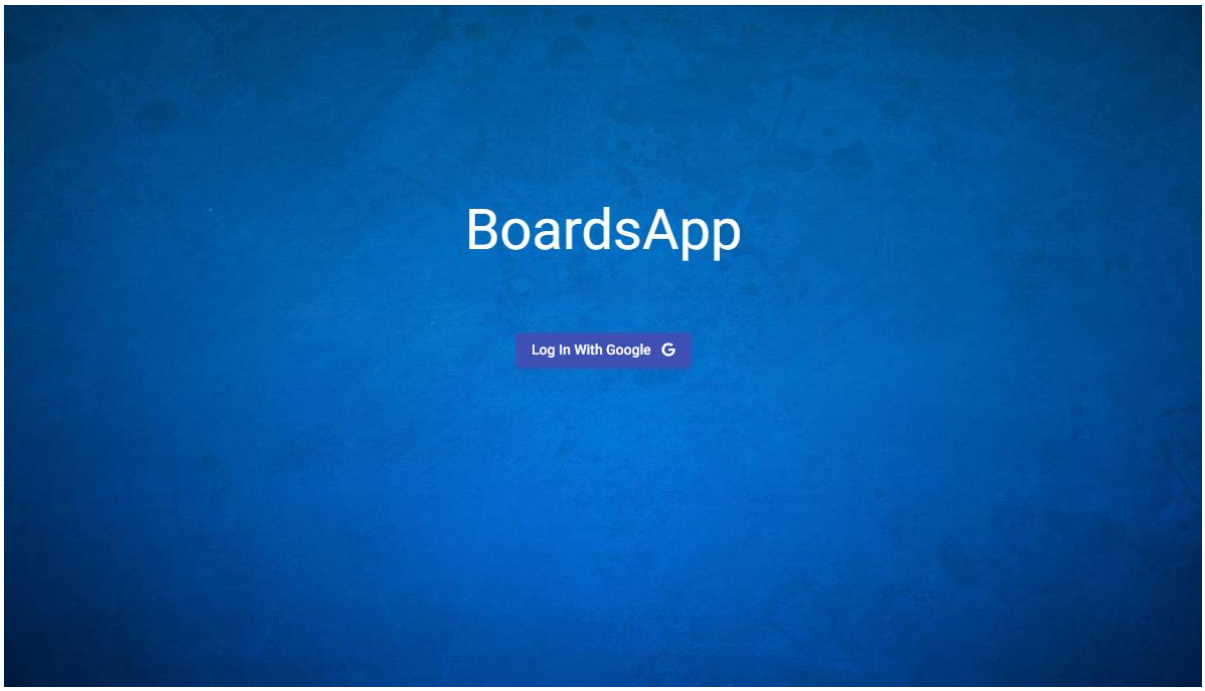

Figura 11. Vista inicial del sistema

Una vez que el usuario clickea el botón de inicio de sesión, se abre una ventana pop-up para que ingrese sus credenciales (Fig 12).

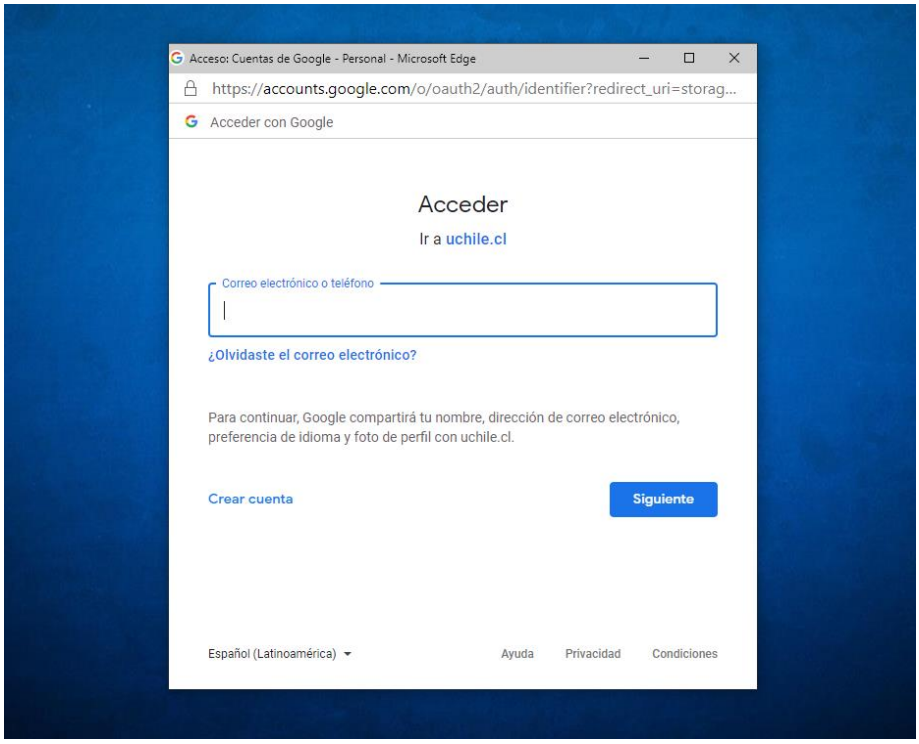

Figura 12. Ventana pop-up para inicio de sesión.

Luego de que el usuario inicie sesión con sus credenciales de Google, éste es dirigido a un menú donde se muestran dos selectores, tal como se ve en la Figura 13. En el primero, el usuario es capaz de seleccionar un tablero ya existente, para luego ser redirigido a él. El segundo selector contiene los layouts disponibles, en donde el usuario puede seleccionar el de su preferencia, y luego clickear el botón 'New Board'. Al hacer esta acción, se crea un nuevo tablero vacío con el layout seleccionado anteriormente. Luego, se redirige al usuario hacia dicho tablero.

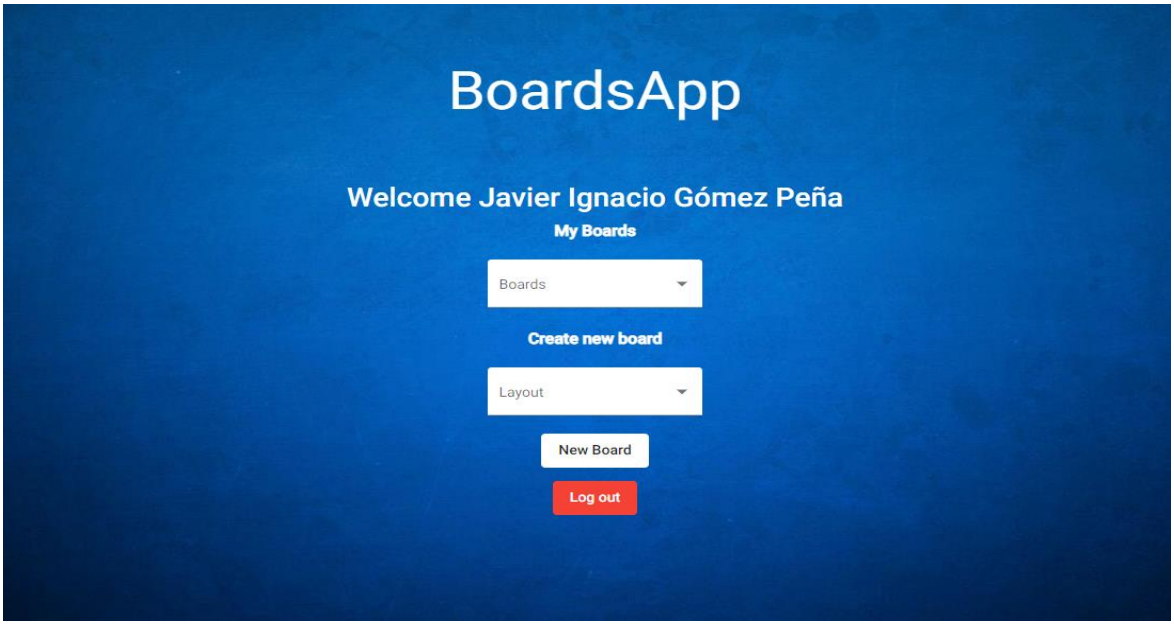

Figura 13. Vista del menú principal del sistema.

5.2 Vista del tablero

Una vez que el usuario es dirigido hacia un tablero (ya sea, uno nuevo o uno existente), se le presenta la vista global del mismo (Figura 14). Allí se muestran las distintas secciones del layout, junto a las tarjetas que el tablero contiene (si es que no está vacío). Además, se muestra el título del tablero, la cantidad de usuarios activos y un botón para invitar a nuevos usuarios.

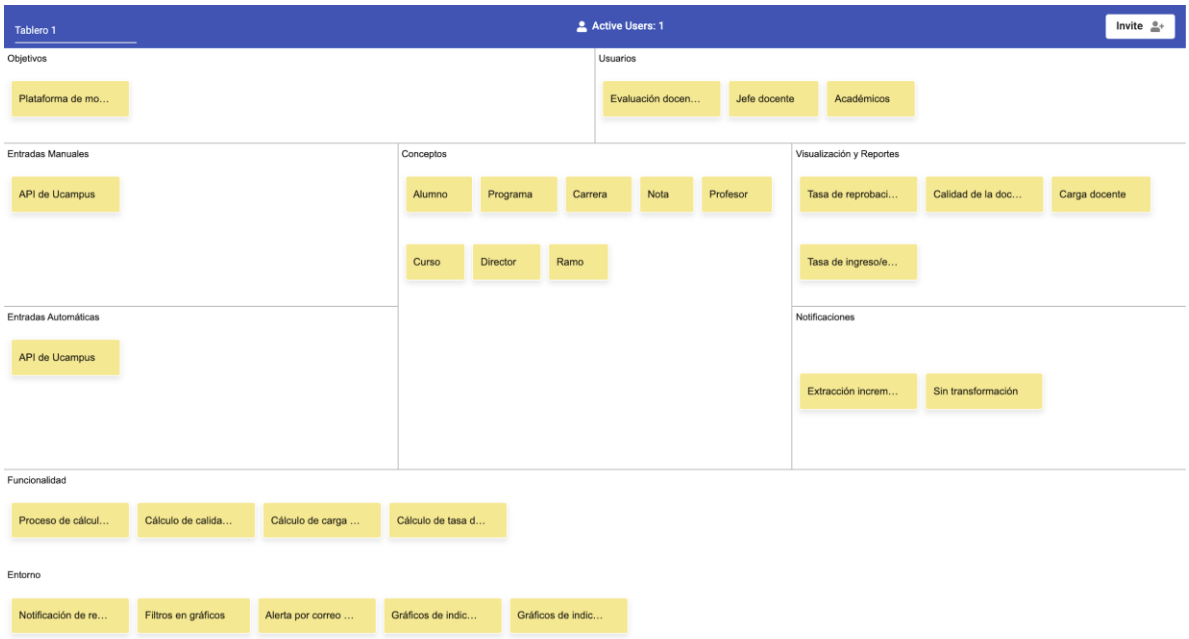

Figura 14. Vista global de un tablero y su contenido.

En este punto, un usuario puede crear, mover, editar y eliminar tarjetas. También se le permite modificar el título del tablero, e invitar a nuevos usuarios a colaborar en la instanciación del mismo.

Para agregar una tarjeta a una nueva sección, el usuario debe clickear el ícono con el símbolo ‘+’, el cual se muestra en cada sección cuando el usuario posiciona el cursor sobre una de ellas.

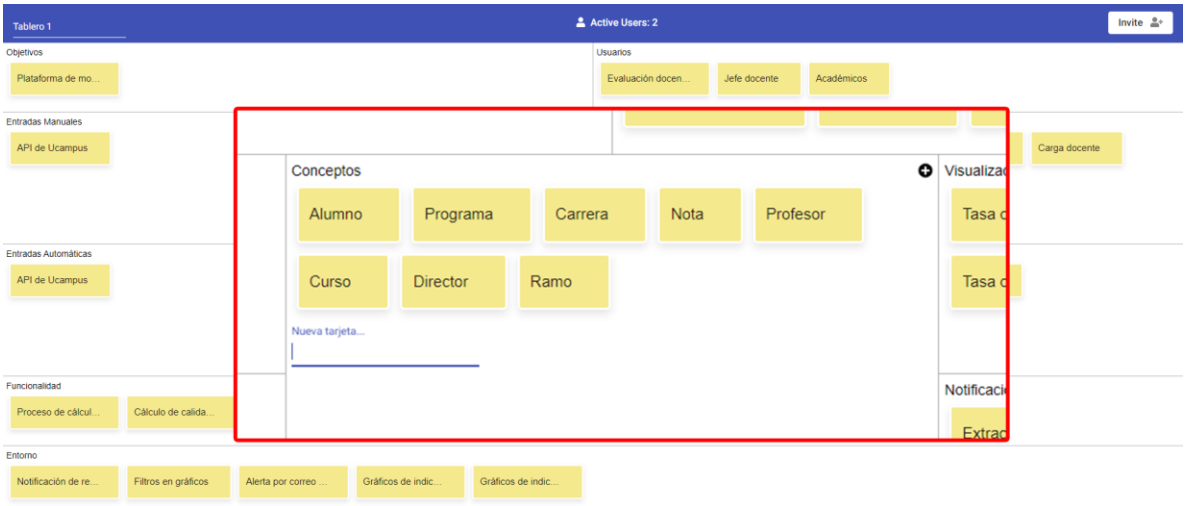

Figura 15. Botón y entrada de texto para añadir nuevas tarjetas

Como se muestra en la figura 15, al poner el cursor sobre una sección (en este caso, 'Conceptos') se muestra un botón con el símbolo '+' en la esquina superior derecha de la sección. Al clickear este botón, se muestra una entrada de texto con la etiqueta 'Nueva tarjeta'. El usuario puede escribir el título que desee para la nueva tarjeta, y al apretar la tecla 'Enter', se añade una tarjeta nueva con el texto escrito.

5.2.1 Mover una tarjeta

Para mover una tarjeta, un usuario debe posicionar el cursor sobre ella y clickear, que simula la operación de "pinchar la tarjeta". Al realizar esto, se podrá arrastrar la tarjeta, al mismo tiempo que se muestra una vista previa de la ubicación temporal en que se encontraría la misma, si el usuario la soltara en ese momento, tal como se muestra en la Figura 16.

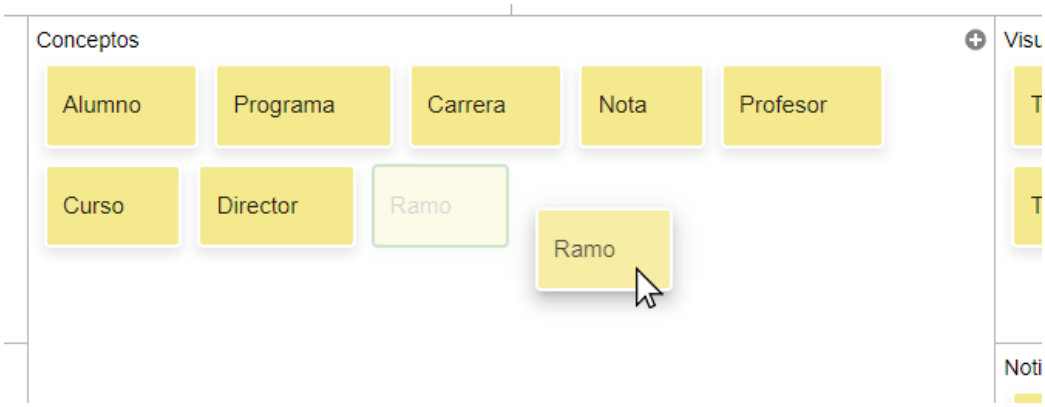

Figura 16. Tarjeta siendo arrastrada por un usuario.

Luego, el usuario puede mover la tarjeta a su gusto a través del tablero. A medida que la tarjeta se mueva, se va a mover la vista previa de la misma, como se muestra en la Figura 17.

Una vez que la vista previa de la tarjeta se encuentre en el lugar deseado por el usuario, éste puede soltarla, instanciándose así la vista previa.

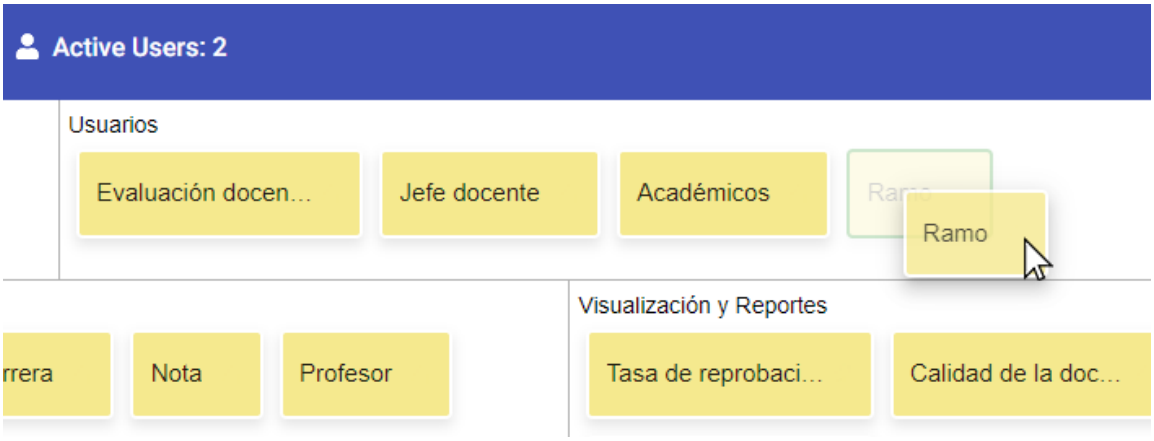

Figura 17. Tarjeta arrastrada a una sección distinta.

5.2.2 Tarjetas - vista detallada

Cuando el usuario posiciona el cursor sobre una tarjeta, se muestra momentáneamente un ícono de edición sobre la misma, tal como se muestra en la Figura 19.

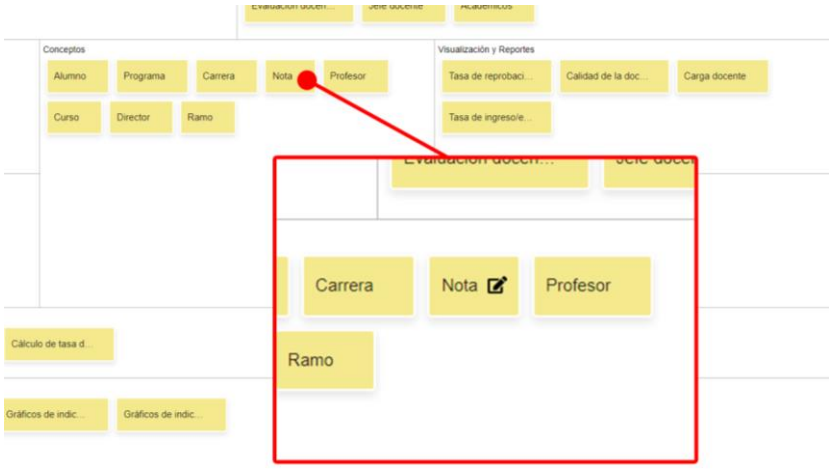

Figura 19. Ícono de edición de una tarjeta.

Al clickear este ícono, se muestra la vista detallada de la tarjeta, donde es posible cambiar el título de ella (que se muestra en la vista global del tablero), cambiar su descripción o eliminar la tarjeta.

Figura 20. Vista detallada de una tarjeta.

5.2.3 Usuarios activos

En la Figura 21 se muestra el listado de usuarios activos en el tablero actual, el cual se despliega al clickear el botón que indica “Active Users”. Allí se puede apreciar que cada usuario está vinculado a un color en específico, el cual se muestra como contorno del ícono de cada usuario, ubicado al lado derecho de la dirección de correo electrónico del mismo.

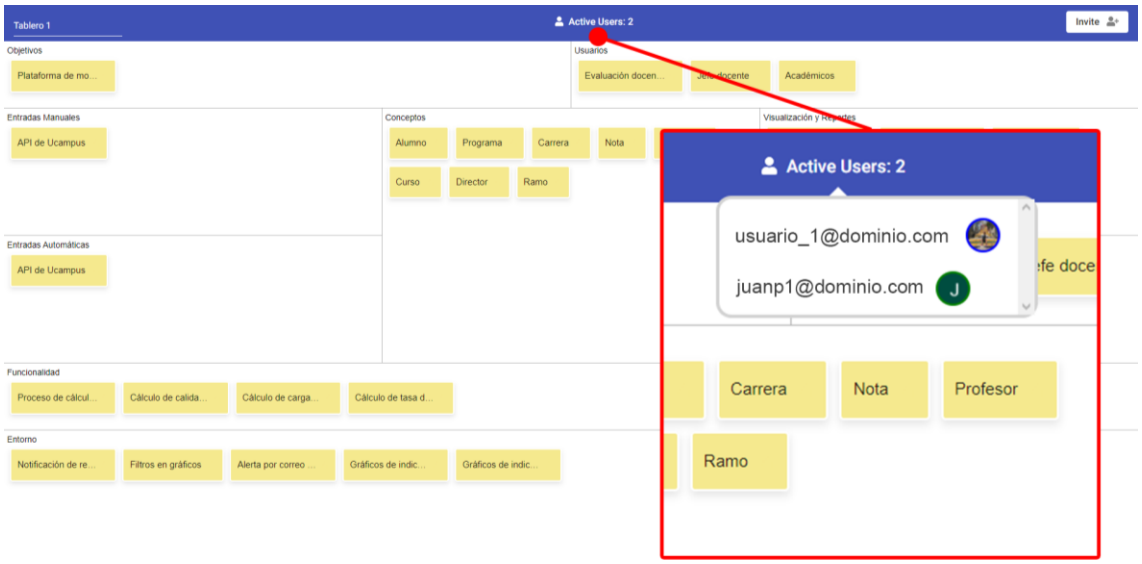

Figura 21. Listado de usuarios activos en un tablero.

El color asignado a cada usuario se apreciará cuando éste bloquee una tarjeta; por ejemplo, para editarla. De esta manera, otros usuarios podrán ver fácilmente quién ha bloqueado la tarjeta, sin necesidad de abrir la vista detallada de la misma.

5.2.4 Tarjeta bloqueada

Cuando una tarjeta es modificada, ésta se bloquea frente a cualquier acción de otros usuarios. Al producirse esto, a la tarjeta se le añade un borde de color, coincidiendo con el correspondiente al usuario que ha bloqueado la tarjeta. De esta manera, es fácil reconocer quién ha bloqueado una tarjeta, si se considera la información obtenida desde el listado de usuarios activos.

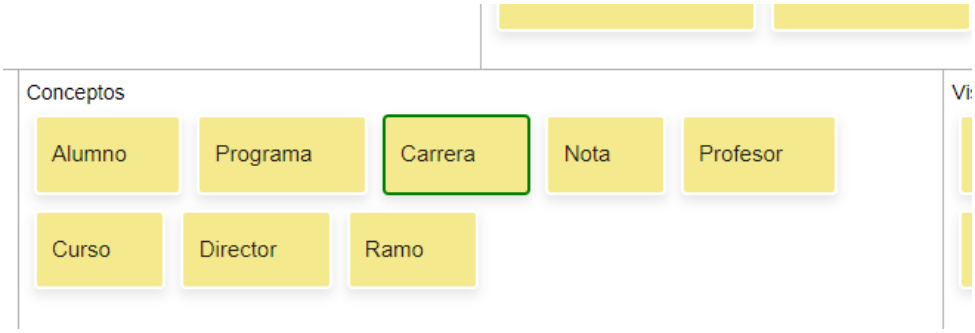

Figura 22. Tarjeta ‘Carrera’ bloqueada por usuario con color verde.

De todas formas, se permite abrir la vista detallada de la tarjeta, permitiendo leer la descripción, a pesar de estar bloqueada (Figura 23).

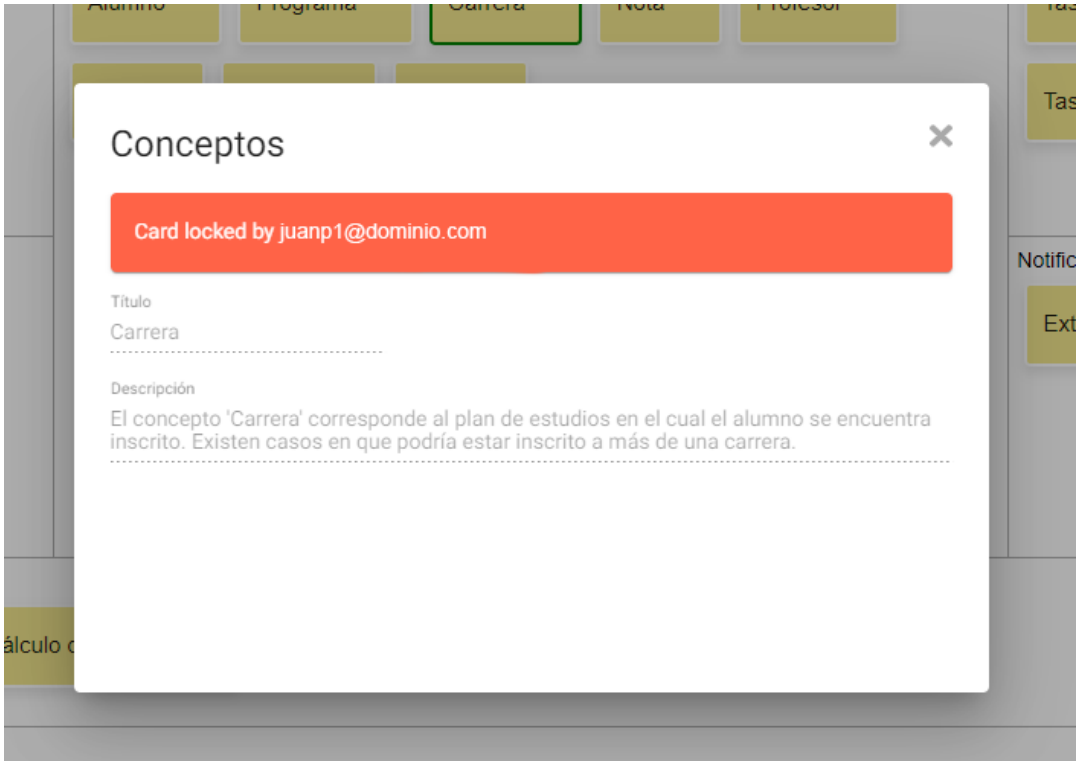

Figura 23. Vista detallada de una tarjeta bloqueada.

5.2.5 Título del tablero

En la esquina superior izquierda del tablero se muestra el título de este. Este parámetro puede ser cambiado por cualquiera de los usuarios; para ello, el usuario debe clickear en el campo de entrada de texto, donde se muestra el título. Una vez que el usuario ha ingresado el nuevo título, basta con clickear en cualquier parte del tablero para que el cambio se propague a los demás usuarios.

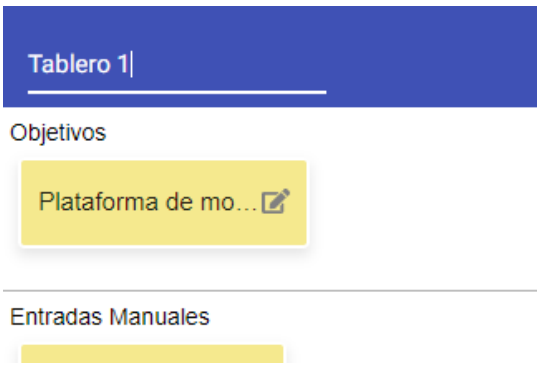

Figura 24. Edición del título del tablero.

5.2.6 Añadir nuevos usuarios al tablero

En la esquina superior derecha se muestra un botón con la etiqueta ‘Invite’. Al clickearlo, se despliega una entrada de texto donde el usuario puede ingresar la dirección de correo de un nuevo usuario, para así permitirle el acceso al tablero. Luego de escribir la dirección, se debe presionar el botón con el símbolo ‘+’ para enviar el cambio al servidor.

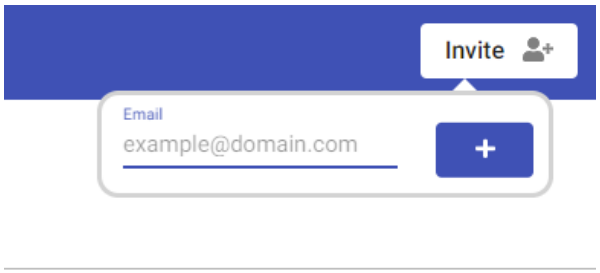

Figura 25. Botón de invitar nuevos usuarios al tablero.

6. Evaluación de la solución

En este capítulo se describen los procesos de evaluación de la herramienta desarrollada, así como los resultados obtenidos en dichos procesos. A continuación se presenta la evaluación de la correctitud de las operaciones que realiza la herramienta (test funcional), y luego se reporta la evaluación de la usabilidad y utilidad de la misma.

6.1. Evaluación de la correctitud de la solución

Para evaluar la correctitud de la solución, se idearon diversos casos de prueba, los cuales buscan determinar las capacidades de la herramienta para realizar operaciones que son mandatorias. Estos casos fueron ejecutados por usuarios reales, quienes posteriormente evaluaron la correctitud de las operaciones realizadas.

6.1.1. Descripción de los casos de prueba

1. Título: *Creación de nuevo tablero.*
   Descripción: Crear un nuevo tablero. Una vez creado, el usuario debe mantener el acceso al tablero después de salir y volver a entrar al sistema.

   Precondiciones:
   1. Poseer cuenta Google.
   2. Iniciar sesión.

   Pasos:
   1. Seleccionar un Layout en el menú ‘Create new board’.
   2. Presionar el botón ‘New Board’.
   3. Cerrar el navegador.
   4. Entrar al sistema nuevamente.
   5. Iniciar sesión.
   6. Seleccionar el menú desplegable ‘My Boards’.

   Resultado esperado: El usuario debe ver el nuevo tablero ‘New Board’ en el menú desplegable.

2. Título: *Creación de nueva tarjeta.*
   Descripción: Crear una nueva tarjeta. Una vez creada, el contenido debe mantenerse, aunque el usuario salga del sistema y luego vuelva a entrar.

   Precondiciones:
   1. Poseer cuenta Google.
   2. Iniciar sesión.
   3. Contar con un tablero creado.

Pasos:

1. Entrar a un tablero previamente existente.
2. Crear una nueva tarjeta.
3. Escribir contenido.
4. Salir del sistema.
5. Volver a entrar al tablero.

Resultado esperado: El usuario debe ver la tarjeta creada con el contenido ingresado.

3. Título: *Cambio de título*
   Descripción: Cambiar el título de un tablero. El título debe mantenerse al reiniciar la sesión.

   Precondiciones:
   1. Poseer cuenta Google.
   2. Iniciar sesión.
   3. Contar con un tablero creado.

   Pasos:
   1. Entrar a un tablero previamente existente.
   2. Seleccionar el título del tablero.
   3. Escribir un nuevo título.
   4. Salir del sistema.
   5. Volver a entrar al tablero.

   Resultado esperado: El usuario debe ver el nuevo título en el tablero.

4. Título: *Mover una tarjeta.*
   Descripción: Mover una tarjeta de una sección a otra en el tablero.

   Precondiciones:
   1. Poseer cuenta Google.
   2. Iniciar sesión.
   3. Contar con un tablero creado.
   4. Crear una tarjeta en el tablero.

   Pasos:
   1. Entrar a un tablero previamente existente.
   2. Arrastrar una tarjeta existente de una sección a otra distinta.
   3. Salir del sistema.
   4. Volver a entrar al tablero.

Resultado esperado: El usuario debe ver la tarjeta en la última sección en donde fue soltada.

5. Título: *Editar una tarjeta.*
   Descripción: Editar el contenido de una tarjeta (post-it).
   Precondiciones:
   1. Poseer cuenta Google.
   2. Iniciar sesión.
   3. Contar con un tablero creado.
   4. Crear una tarjeta en el tablero.

   Pasos:
   1. Entrar a un tablero previamente existente.
   2. Abrir la vista detallada de una tarjeta.
   3. Modificar el título de la tarjeta.
   4. Cerrar la vista detallada de la tarjeta.
   5. Salir del sistema.
   6. Volver a entrar al tablero.

   Resultado esperado: El usuario debe ver la tarjeta con el último contenido (título) ingresado.

6. Título: *Consistencia de la información.*
   Descripción: Editar un tablero (agregar, mover y borrar tarjetas) en una sesión, y el contenido de éste debe ser consistente en todas las sesiones abiertas.

   Precondiciones:
   1. Poseer cuenta Google.
   2. Abrir el sistema en dos ventanas distintas.
   3. Iniciar sesión.
   4. Contar con un tablero creado.

   Pasos:
   1. Entrar a un tablero previamente existente.
   2. Crear una tarjeta en el tablero.
   3. Mover la tarjeta.
   4. Crear una tarjeta en una sección distinta.
   5. Mover la última tarjeta creada.
   6. Eliminar la primera tarjeta.
   7. Eliminar la segunda tarjeta.

Resultado esperado: El contenido del tablero debe ser consistente en todo momento entre las dos o más ventanas del sistema que muestran dicho contenido.

6.1.2. Participantes en la evaluación

Para la evaluación funcional, participaron tres ingenieros dedicados al desarrollo de proyectos de software. Estos ingenieros contaban con 2 años de experiencia promedio en el uso de tableros digitales de distinto tipo. Se ejecutó el plan de pruebas anteriormente descrito.

6.1.3. Resultados obtenidos

La tabla 2 muestra los resultados obtenidos en las pruebas de correctitud realizadas, considerando los distintos participantes antes descritos. Todos los usuarios ejecutaron con éxito las pruebas funcionales de la herramienta.

Tabla 2. Resultado de la ejecución del plan de pruebas funcional.

Caso de prueba	Usuario 1	Usuario 2	Usuario 3
1	Éxito	Éxito	Éxito
2	Éxito	Éxito	Éxito
3	Éxito	Éxito	Éxito
4	Éxito	Éxito	Éxito
5	Éxito	Éxito	Éxito
6	Éxito	Éxito	Éxito

6.2. Evaluación de la usabilidad y utilidad de la solución

Si bien las pruebas funcionales son algo esencial que se debe tener en cuenta, también lo son las pruebas de usabilidad y utilidad de la herramienta. De esta manera se determina si el proyecto desarrollado es útil para el usuario final o no.

A continuación se presentan los participantes del proceso de evaluación, la dinámica realizada, el instrumento utilizado y los resultados obtenidos en términos de la usabilidad y utilidad percibida por los usuarios.

6.2.1. Participantes en la evaluación

Para evaluar la usabilidad de la herramienta se realizaron dos sesiones con distintos usuarios. En el proceso participaron en total cuatro personas, las cuales se dedicaban a la preventa de proyectos de software. Estas personas asumieron el rol de representantes del desarrollador en cada una de las sesiones. Cada uno de estos usuarios contaba con más de 8 años de experiencia en el desarrollo de software, y entre 2 y 4 años de experiencia en el uso de tableros para la realización de scoping de proyectos.

6.2.2. Descripción del proceso realizado

El proceso consistió en dos sesiones por separado, en las cuales participaron 2 usuarios con rol de desarrollador en cada una. En éstas también participaba un cliente, el cual cumplía el rol de stakeholder de un producto ficticio. Los desarrolladores debieron interrogar al cliente para determinar el alcance del proyecto que el stakeholder solicitaba. Para este proceso los usuarios se apoyaron en el uso de la herramienta. Luego de la sesión de dimensionamiento del proyecto usando el tablero digital, los desarrolladores completaron una evaluación que considera tanto la usabilidad como la utilidad de la herramienta. En la siguiente sección se describe dicho instrumento.

6.2.3. Descripción del instrumento utilizado

Para la evaluación de la usabilidad de la herramienta se utilizó la escala SUS (System Usability Scale), o Escala de Usabilidad de Sistemas, en español [25, 26]. Ésta provee un instrumento confiable y rápido de aplicar, el cual sirve para medir usabilidad de una aplicación Web. Dicho instrumento consiste en un cuestionario de 10 ítems (afirmaciones que evalúan la usabilidad del sistema), donde el usuario indica su nivel de acuerdo o desacuerdo con ellas, usando una escala Likert de 1 a 5 (siendo 1 totalmente en desacuerdo, y 5 totalmente de acuerdo). Así los usuarios evalúan cómo ellos perciben los diferentes aspectos del sistema, luego de utilizarlo durante el proceso de evaluación. Finalmente, se calcula el promedio de las respuestas de los participantes, y con ello, una calificación final que define el nivel de usabilidad de la aplicación Web.

El cuestionario SUS utilizado es la versión original, sólo que los ítems fueron traducidos al castellano. Los ítems evaluados fueron los siguientes:

1) Creo que me gustaría usar este sistema frecuentemente.
2) Encuentro al 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) Las funciones de este sistema están bien integradas.
6) Creo que el sistema es muy inconsistente.
7) Imagino que la mayoría de la gente aprendería a usar este sistema en forma muy rápida.
8) Encuentro que el sistema es muy engorroso de usar.
9) Me siento seguro/a al usar este sistema.
10) Necesité aprender muchas cosas antes de aprender a usar este sistema.

En este proceso de evaluación, el instrumento anterior fue extendido para incluir ítems que evalúan la utilidad del sistema según la percepción del propio usuario. Esto incluye el nivel de suficiencia de la funcionalidad entregada por la herramienta, y también los servicios potencialmente faltantes para llevar a cabo el proceso de scoping de mejor manera. Los ítems agregados fueron los siguientes:

11) Los servicios ofrecidos por la herramienta son suficientes como para determinar cómodamente la solicitud del cliente y su alcance.
12) Hay servicios que deben ser agregados a la herramienta para realizar esta labor de manera más apropiada.
13) Hay servicios que, aunque no son mandatorios, podrían ser agregados a la herramienta para hacerla más efectiva.
14) *Comentario abierto*: Indicar los servicios que a su juicio deben ser agregados a la herramienta, y también aquellos que usted considera como deseable de agregar.

Para el cálculo del nivel de usabilidad se suman los resultados entregados por los usuarios, donde las respuestas se computarán de la siguiente manera:

- Los ítems impares (1, 3, 5, 7 y 9) tomarán el valor de la respuesta y se les restará 1.
- Los ítems pares (2, 4, 6, 8 y 10) tomarán el valor de 5, menos el valor de la respuesta.

Una vez obtenida la suma total, a ese resultado se lo multiplica por 2,5, así la fórmula de cálculo queda como se muestra a continuación:

Nivel de Usabilidad: ((r(1)-1) + (5-r(2)) + (r(3)-1) + (5-r(4)) + (r(5)-1) + (5-r(6)) + (r(7)-1) + (5-r(8)) + (r(9)-1) + (5-r(10))) * 2,5

El valor final obtenido respecto a la usabilidad del sistema se evalúa en los siguientes tracks:

- Si no llega a 25 puntos, entonces el escenario es “lo peor imaginable”.
- Si va desde 25 a 38, entonces la usabilidad es considerada “pobre”.
- Si va desde 39 a 52, entonces la usabilidad es considerada “ok” (esto ya es como un mínimo aceptable).
- Si va desde 53 a 73, entonces la usabilidad es considerada “buena”.
- Si va desde 74 a 85, entonces la usabilidad es considerada “excelente”.
- Si va desde 86 a 100, entonces es “lo mejor posible” (aspiración de toda aplicación Web).

Respecto al cómputo de la utilidad percibida por el usuario (ítems 11 a 13), estos se computan de la siguiente manera:

Utilidad percibida: 0,5 * (r(11)-1) + 0,35 * (5-r(12)) + 0,15 * (5-(r(13))

La evaluación del ítem 11 pesa un 50% del total de la utilidad, ya que ahí se determina si la herramienta llega a ser un MVP (Minimum Viable Product). El ítem 12 representa funcionalidad

que es necesaria, pero no fundamental; por esa razón, dicho ítem pesa un 35%. Finalmente, el ítem 13 que representa a la funcionalidad deseable, pesa un 15%.

6.2.4. Resultados obtenidos

En la siguiente tabla se muestran los resultados recopilados en la evaluación de la usabilidad, de acuerdo con la percepción de los usuarios.

Tabla 3. Resultados de la evaluación de usabilidad de la herramienta.

Item / Usuario	**Item 1**	**Item 2**	**Item 3**	**Item 4**	**Item 5**	**Item 6**	**Item 7**	**Item 8**	**Item 9**	**Item 10**	**Total**
Usuario 1	3	1	5	2	3	2	5	2	4	1	80
Usuario 2	3	2	4	1	3	2	5	1	4	1	80
Usuario 3	5	1	5	1	4	1	4	1	4	3	87.5
Usuario 4	5	1	5	1	5	1	5	1	4	1	97.5
Promedio	4	1.25	4.75	1.25	3.75	1.5	4.75	1.25	4	1.5	86.25

Acorde a los parámetros para medir este resultado, el puntaje final obtenido de 86.25 puntos corresponde a la categoría “lo mejor posible”. De esto se desprende que, si bien la herramienta es percibida por los usuarios como muy buena, existen algunos aspectos en donde se podrían realizar algunas mejoras menores no fundamentales para la usabilidad herramienta.

En la tabla 4 se presentan los resultados de la evaluación respecto a la utilidad de la herramienta. Es importante recordar que los resultados mostrados en la tabla 4 en la columna “Total” fueron calculados con la función de *utilidad percibida*, descrita en la sección 6.2.3 de este documento.

Tabla 4. Resultados de la evaluación de utilidad de la herramienta

Item / Usuarios	Item 1	Item 2	Item 3	Total
Usuario 1	4	3	2	2.65
Usuario 2	2	5	5	0.5
Usuario 3	4	3	3	2.5
Usuario 4	4	3	5	2.2
Promedio	3.5	3.5	3.75	1.96

Si bien la usabilidad percibida por los usuarios fue muy alta, al momento de evaluar la utilidad se obtuvo un puntaje de 1.96, de un máximo de 4. Esto indica que la herramienta es lo suficientemente útil para los usuarios, pero sería de su agrado que se incorporen nuevas funcionalidades que amplíen su utilidad.

Se les pidió a los usuarios indicar qué características o funcionalidad faltaba en la herramienta para aumentar la utilidad de ésta. Uno de los aspectos que más se repitió en la retroalimentación entregada por los usuarios, es la capacidad de colorear las tarjetas del tablero a gusto, con el fin de que permitir resaltar algunos contenidos (tarjetas) del tablero que sean más importantes que otros. También indicaron que se podría mejorar la forma de visualización del contenido de las tarjetas, ya que en ocasiones no se mostraba el título de una tarjeta por completo, y se debía acceder a la vista detallada de la misma.

A pesar de que estos aspectos a mejorar son de gran importancia, también indicaron que la herramienta les pareció muy cómoda y simple. Esto les permitió enfocarse en la interacción con los stakeholders, y no desviarse explorando opciones y configuraciones que podrían haber existido en el tablero. También indicaron que les agradó que las tarjetas cuentan con una vista detallada, donde es posible agregar información complementaria. Algunos participantes realizaron una comparación con otras herramientas existentes en el mercado (que ellos venían utilizando), e indicaron que “la herramienta presentada es menos engorrosa para el trabajo”, refiriéndose al proceso de dimensionamiento de proyectos que ellos acostumbran a realizar.

7. Conclusiones y trabajo a futuro

Al comienzo de este documento se habló del proceso de preventa de un proyecto de software, y de cómo éste debe realizarse de manera remota en ciertas ocasiones. Dado que en su versión presencial el tablero cuenta con un espacio visual compartido (desplegado en pizarras o tableros impresos), para la versión digital se debía brindar una herramienta que reemplace al tablero físico, pero que mantenga todos aquellos aspectos que son importantes para apoyar la actividad de dimensionamiento de proyectos durante la preventa. Para esto se estableció como objetivo el desarrollo de un tablero digital que permita la colaboración simultánea de distintos usuarios, con el fin de apoyar la definición conjunta del alcance de un proyecto a estimar.

Con ese objetivo en mente, se implementó el tablero digital, en el cual se les permite a distintos usuarios ingresar información en forma de tarjetas. Estas tarjetas son compartidas por los usuarios entre sí, de forma que todos ven el mismo contenido al mismo tiempo. Además, ellos son capaces de añadir, modificar, mover y eliminar tarjetas, y dichos cambios son propagados en tiempo real a otros usuarios. De esta manera, los usuarios pueden participar de manera remota en la definición del alcance de un proyecto de software.

La herramienta se evaluó mediante dos sesiones de trabajo con usuarios reales; empleados de empresas de desarrollo de software a la medida. Estos usuarios interactuaron con un stakeholder con el fin de realizar el scoping (definición del alcance) del proyecto utilizando el tablero digital. En total participaron cuatro ingenieros de software, los cuales realizaron una evaluación de la herramienta posterior al ejercicio. De dicha evaluación se obtuvo como resultado que los usuarios percibieron una muy buena usabilidad de la herramienta, mientras que también destacaron algunos aspectos a mejorar para que ésta se volviera más útil.

A partir de los resultados de las pruebas realizadas, se puede concluir que este trabajo logra cumplir con los objetivos (general y específicos) planteados inicialmente. El objetivo general hace referencia al desarrollo de una herramienta colaborativa que permite acordar el alcance de un proyecto a desarrolladores y clientes; lo cual está claramente cumplido. Por otra parte, los usuarios percibieron una alta usabilidad del tablero digital, aunque realizaron algunas observaciones sobre la utilidad de la herramienta, dejando algunos aspectos como posibles puntos de mejora. Del mismo modo, se logró crear un ambiente colaborativo para usuarios concurrentes, dejando el contenido almacenado en un servidor para futuras sesiones de trabajo.

De este trabajo, el estudiante rescata el acercamiento con la industria, donde toma fundamental importancia la concordancia entre partes al momento de estimar o desarrollar un proyecto de software. La capacidad de adaptar mecanismos entre clientes y desarrolladores a ambientes no favorables, como la emergencia sanitaria vivida al momento de este desarrollo, se vuelve esencial para la industria del software, y de TI en general. Sin herramientas como la desarrollada se dificultaría en gran medida la instancia de alineamiento entre las partes para poder dar un buen comienzo al proyecto a desarrollar.

A pesar de que los objetivos de este trabajo fueron cumplidos, a continuación se listan posibles mejoras a realizar a futuro, con el fin de ampliar la funcionalidad de la herramienta:

- Implementar un historial de cambios en el tablero.
- Implementar la funcionalidad para deshacer cambios en el tablero.
- Añadir una ‘papelera de reciclaje’ para las tarjetas eliminadas, con posibilidad de ser restauradas.
- Permitir cambiar el color de las tarjetas del tablero.
- Permitir personalizar el tamaño de las tarjetas.
- Añadir nuevos campos a las tarjetas, como por ejemplo imágenes.
- Añadir soporte para distintos tipos de usuarios, con distintos roles.
- Permitir votaciones entre usuarios con respecto a una tarjeta en específico.
- Añadir a la interfaz de usuario soporte para distintos idiomas.

Si bien estas posibles mejoras no son fundamentales para el funcionamiento de la herramienta, existiría un incremento en la utilidad de la misma, permitiendo que los usuarios realicen el scoping de proyectos de software de una manera más cómoda.

Bibliografía

[1] C. Müller, M. Koch, S. Adam. Elicitation of information needs in precontract requirements engineering. International Workshop on Creativity in Requirements Engineering (CreaRE) at REFSQ 2015. CEUR Workshops, vol. 1342, 77-82, 2015.

[2] P. Savolainen, J.J. Ahonen, I. Richardson. 2015. When Did Your Project Start? - The Software Supplier's Perspective. Journal of Systems and Software, vol. 104, C, 32-40, 2015.

[3] T. Vera, S.F. Ochoa, D. Perovich. Requirements Engineering at Pre-contract stage: Exploring the Processes and Practices Used in Small and Medium-Sized Software Enterprises. Accepted to be presented at the 36th ACM/SIGAPP Symposium On Applied Computing (SAC'21), Gwangju, South Korea, March 22-26, 2021.

[4] Osterwalder, Alexander, and Yves Pigneur. Business Model Generation: A Handbook For Visionaries, Game Changers, And Challengers. Wiley, 2010.

[5] T. Vera, D. Perovich, S.F. Ochoa. An Instrument to Define the Product Scope at Pre-selling Time. Accepted to be presented at the IEEE 24th International Conference on Computer Supported Cooperative Work in Design (IEEE CSCWD'21). Dalian, China, May 5-7, 2021.

[6] Miro. [En línea] <https://miro.com> [Consulta: 28/02/2021]

[7] Cardsmith. [En línea] <https://cardsmith.co> [Consulta: 28/02/2021]

[8] Mural. [En línea] <https://www.mural.co> [Consulta: 28/02/2021]

[9] jQueryUI Draggable. [En línea] <https://jqueryui.com/draggable> [Consulta: 28/02/2021]

[10] DraggableJS. [En línea] <https://shopify.github.io/draggable> [Consulta: 28/02/2021]

[11] GridstackJS. [En línea] <https://gridstackjs.com> [Consulta: 28/02/2021]

[12] Angular framework. [En línea] <https://angular.io> [Consulta: 28/02/2021]

[13] Material Drag and Drop. [En línea] <https://material.angular.io/cdk/drag-drop/overview> [Consulta: 28/02/2021]

[14] Dragula. [En línea] <https://valor-software.com/ng2-dragula> [Consulta: 28/02/2021]

[15] Express.js. [En línea] <https://expressjs.com/es/> [Consulta: 28/02/2021]

[16] Node.js. [En línea] <https://nodejs.org/es/> [Consulta: 28/02/2021]

[17] MongoDB. [En línea] <https://www.mongodb.com/es> [Consulta: 28/02/2021]

[18] MEAN stack. IBM [En línea] <https://www.ibm.com/cloud/learn/mean-stack-explained> [Consulta: 28/02/2021]

[19] Three tier architecture. IBM [En línea] <https://www.ibm.com/support/knowledgecenter/en/SSEQTP_8.5.5/com.ibm.websphere.base.iseries.doc/ae/covr_3-tier.html> [Consulta: 28/02/2021]

[20] REST API [En línea] <https://www.ibm.com/cloud/learn/rest-apis> [Consulta: 28/02/2021]

[21] BSON. MongoDB [En línea] <https://www.mongodb.com/json-and-bson> [Consulta: 28/02/2021]

[22] Grid-template-columns [En línea] <https://developer.mozilla.org/es/docs/Web/CSS/grid-template-columns> [Consulta: 28/02/2021]

[23] CSS Grid [En línea] <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout> [Consulta: 28/02/2021]

[24] Google [En línea] <https://www.google.com> [Consulta: 28/02/2021]

[25] Brooke, John. SUS: A quick and dirty usability scale. Book Chapter, in: Usability Evaluation in Industry, pp. 189-197. 1995.

[26] Brooke, John. SUS: A Retrospective. Journal of Usability Studies, Vol. 8, Issue 2, 29-40. 2013.