Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Nombre del Centro

Trabajo Fin de Grado

APLICACIÓN PARA LA CUANTIFICACIÓN DE TRABAJO PERSONAL DE CADA ESTUDIANTE EN TABLEROS TRELLO

Alumno: Román Martínez, Inmaculada

Tutor: Rivas Santos, Víctor Manuel
Dpto: Departamento de Informática

Junio, 2022

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Victor Manuel Rivas Santos , tutor del Proyecto Fin de Carrera titulado: Aplicación para la cuantificación de trabajo personal de cada estudiante en tableros Trello, que presenta Inmaculada Román Martínez, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2022

El alumno:

ROMAN MARTINEZ INMACULADA - 26509584Z

Firmado digitalmente por ROMAN MARTINEZ INMACULADA - 26509584Z
Fecha: 2022.06.30 12:49:59 +02'00'

Inmaculada Román Martínez

Los tutores:

Firmado por RIVAS SANTOS VICTOR MANUEL - 29798183G el día 30/06/2022 con un certificado emitido por AC FNMT Usuarios

Victor Manuel Rivas Santos

Índice

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Para hacer una breve introducción sobre el proyecto, es importante saber en qué consiste la gestión de proyectos y como ha ido evolucionando a lo largo de los años.

Hoy en día es muy común el uso de software para la administración de proyectos, ya sea en una empresa llevando a cabo proyectos más complejos o de uso personal, ayudándonos a realizar una serie de tareas o actividades siguiendo un orden.

Un software de gestión de proyectos ayuda al equipo de trabajo a seguir unas pautas para desarrollar un proyecto, aumentando así la eficacia y productividad del equipo y ayudando a los miembros del equipo a estar preparados ante cualquier cambio que surja durante el desarrollo del mismo.

Para gestionar un proyecto, podemos dividir la gestión en diferentes pasos.

- Planificar cuáles son las tareas/actividades que nos van a ayudar a cumplir con los objetivos deseados y el período de tiempo para desarrollarlos.
- Organizar a los miembros del equipo para saber qué tienen que hacer, cuándo y cómo hacerlo, de esta forma las tareas se realizan de manera ordenada y garantizando la correcta ejecución.
- Motivar a los miembros del equipo para que estos no pierdan la motivación.
- Controlar que el trabajo que se está realizando cumple con las expectativas y los objetivos del cliente.

En este proyecto lo que hemos realizado es una aplicación servidor que se comunica con la API de Trello para obtener la información necesaria de un tablero y de este modo poder obtener los valores numéricos de la actividad que ha realizado cada miembro del equipo.

Lo primero que se hizo fue conocer y documentar el funcionamiento de la herramienta de Trello y a su vez que hacía la API de Trello para saber que consultas eran necesarias para obtener la información deseada.

A raíz de esto, se comenzó con la definición de las historias de usuario que teníamos que realizar y a su vez con el diseño y desarrollo de las aplicaciones.

Más tarde se realizaron las pruebas de validación para comprobar que las aplicaciones cumplían con los objetivos deseados.

1.1 Motivación

La utilización de herramientas para la gestión de software como Trello, son interesantes tanto para la planificación de proyectos como para visualizar cómo están trabajando los miembros del equipo.

En entornos docentes es imprescindible llevar una cuantificación del trabajo de cada miembro para ver si el equipo trabaja de forma adecuada y ver el trabajo real de cada miembro para así puntuar al alumnado. Esto se complica cuando el número de alumnos se excede, ya que hay mirar con mucho detenimiento el trabajo que ha ido realizando cada uno.

Por esta razón surge la idea de crear una aplicación servidor que mediante el uso de la API de Trello se pueda recopilar, procesar y servir información relacionada con cada miembro del equipo de trabajo. Además, para mostrar dicha información se ha optado por crear una aplicación cliente, la cual va a permitir mostrar la información recopilada del servidor al docente de forma clara y concisa, simplificando así el trabajo.

Mi principal motivación es el aprendizaje de nuevas tecnologías como Angular y Spring WebClient. El seguir aprendiendo es algo positivo en todos los aspectos para el desarrollo de aplicaciones.

1.2 Objetivos

El objetivo principal consiste en el desarrollo de una aplicación que permita cuantificar el trabajo realizado por cada estudiante en los tableros Trello. Para lograr este objetivo, se tienen que tener en cuenta los siguientes objetivos secundarios:

- Realizar una descripción detallada de la herramienta Trello, especialmente de la API que proporciona.
- Elegir de forma justificada una metodología de ingeniería de software que permita detectar la funcionalidad deseada para la aplicación servidor, elegir las herramientas de desarrollo más adecuadas y llevar a cabo la implementación de la aplicación.

- Realizar una pequeña aplicación web, app o de escritorio que permita validar el funcionamiento de la aplicación servidor creada.

1.3 Metodología

La metodología a seguir para la realización de nuestro proyecto es la siguiente:

- Estudio de los objetivos que debe alcanzar la aplicación. La tarea principal para el desarrollo correcto será definir cuáles son las tareas básicas que tiene que realizar el sistema
- Estimación y costes del proyecto. Realizar una estimación del tiempo que se va a tardar en desarrollar el proyecto, y cuál va a ser el coste de este.
- Diseño e implementación de la aplicación. Desarrollar una aplicación servidor que pueda comunicarse con la API de Trello y una aplicación cliente que muestre la información al usuario, recopilada por el servidor.
- Recopilación bibliográfica

2. HERRAMIENTA TRELLO

Trello [1] es una herramienta que fue diseñada y creada para poder gestionar proyectos, organizar y realizar tareas, entre otras actividades. Esta herramienta sigue el modelo de trabajo Kanban, el cual permite organizar el trabajo entre los miembros de un equipo y realizar las tareas mostradas en un tablero en forma de tarjetas.

Algunos aspectos que se pueden destacar de Trello son que es un gestor de proyectos online que facilita que los miembros del tablero puedan estar comunicados en todo momento, es fácil e intuitiva y además ofrece un plan gratuito.

Para saber cómo funciona esta herramienta nos vamos a ayudar de la ilustración 1. Imaginemos que tenemos una pizarra colgada en la pared la cual vamos a dividir en varias listas. Al inicio, todas las tarjetas estarán colocadas en la lista de entrada y, conforme se vayan realizando, se irán pasando de una lita a otra hasta que dicha tarea esté completada.

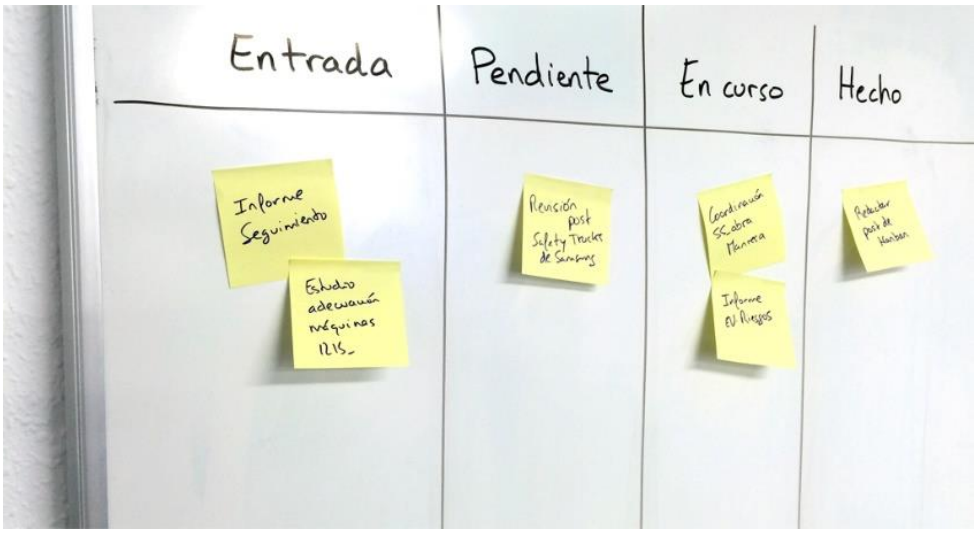

Ilustración 1: Visualización flujo de trabajo

2.1 Historia Trello

La empresa Fog Creek decidió lanzar en el año 2011 la aplicación Trello. Fue un gran descubrimiento para los usuarios ya que les iba a facilitar la gestión de proyectos y tareas de forma online. El primer lanzamiento que la empresa realizó solo estaría disponible para dispositivos iOS, lo que sería un gran inconveniente para muchas personas. Fue por ello y por el gran auge que tuvo, que la empresa decidió lanzar al mercado un año después una versión para dispositivos Android.

Unos años más tarde Trello decide separarse de la empresa Fog Creek y por lo tanto surge Trello Inc. Como la empresa ya contaba con versiones disponibles para iOS y Android, no quedaba más remedio que seguir avanzando en el mercado, y por ello decidieron lanzar la versión para ordenadores, de manera que esto facilitase todavía más el trabajo a los miembros del equipo y que empresas grandes pudieran utilizarla.

Fue tan grande la acogida que tuvo la aplicación que un año más tarde se volvió internacional y se tradujo al español, alemán y portugués brasileño.

Después de la gran aceptación que tuvo esta herramienta, la empresa Atlassian decide comprar Trello en el año 2017.

2.1.1 Fog Creek Software

Fog Creek [2] es una empresa de consultoría que basaba su trabajo en asesorar a otras empresas para su correcta administración. Fue fundada por Joel Spolsky y Michael Pryor, La idea de esta empresa surgió porque no encontraban un sitio fijo en el que llevar a cabo su trabajo y fue entonces cuando pensaron en desarrollar herramientas online que permitiesen que personas ubicadas en distintos lugares pudiesen trabajar de forma cooperativa sin tener que estar en un lugar de trabajo fijo. Han desarrollado otras herramientas como StackOverflow, FogBugz y la más reciente, HyperDev.

Ilustración 2: Logo Fog Creek

En la actualidad esta empresa ha decidido cambiar su nombre y apostar por Glitch, Inc [3]con sede en Nueva York. La mayoría de sus trabajadores trabajan de forma remota.

2.1.2 Atlassian

Atlassian [4] es una conocida empresa australiana desarrollada por Mike Cannon-Brookes y Scott Farquhar, que se encarga del desarrollo de herramientas software para la gestión de proyectos. Esta empresa surgió en el año 2002 y su nombre surgió de la mitología griega, más concretamente del Titán Atlas. En 2001 se realiza el lanzamiento al mercado de Jira 1.0, en 2005 consiguen alcanzar los 1000 clientes y en 2009 consiguen abrir su primera oficina en Ámsterdam. Fue en 2017 cuando decidieron que la herramienta de Trello se uniera a su empresa.

Ilustración 3: Logo Atlassian

Esta empresa fue pensada para trabajar con equipos ya que promueve e incita el espíritu ágil, mejorando así el trabajo en equipo y adaptándose a cualquier metodología ágil (Scrum, Kanban, etc.). Hablando del espíritu ágil, creen que es de vital importancia tener en cuenta una serie de valores que al final van a crear un mejor espíritu de trabajo y el trabajo realizado va a ser muy bueno. Entre los distintos valores encontramos:

- Puesta en común de ideas.
- Crear con corazón, pasión y sabiduría para llevar a cabo un trabajo exitoso.
- Satisfacer en todo momento al cliente.
- Que el trabajo en equipo no se vuelva monótono.
- Saber cómo afrontar nuevos cambios, para que el producto vaya mejorando.

La empresa cuenta con un total de 10 productos, entre ellos Jira Software, Trello, StatusPage, etc.

2.2 ¿Qué es el modelo Kanban?

Trello basa su filosofía de trabajo, en el modelo Kanban. El modelo Kanban [5] surgió en la compañía Toyota, para gestionar el trabajo, agilizar y mejorar la fabricación de vehículos. El objetivo era crear mejores vehículos sin incrementar el coste de estos. Este modelo no define roles en el equipo de trabajo, ni herramientas para el desarrollo, ni fija la duración de las etapas.

Principalmente se centra en estudiar el flujo de trabajo del equipo empleando para ello un tablero Kanban. En los tableros Kanban se trabaja a nivel de características del software y siendo cada lista, una fase del flujo de trabajo. El número de listas en los tableros Kanban pueden variar.

Con el fin de obtener el máximo rendimiento del equipo, Kanban fija una serie de principios [6]:

- **Visualizar el flujo de trabajo.** El equipo tiene que ver cómo irá transcurriendo su proyecto desde el inicio hasta su entrega. Para ello se necesita un tablero, el cual estará dividido en listas y cada lista tendrá una serie de tareas a realizar.

- **Limitar el WIP (work in progress),** para evitar atascos en el flujo de trabajo. Por ejemplo, tener un número límite de tareas por lista, de forma que la realización de cada una de ellas se haga consecutiva y no surjan problemas.
- **Gestionar el flujo de trabajo para que sea óptimo,** de manera que no haya pausas muy grandes entre una tarea y otra, evitando retrasos y que el trabajo sea lento.
- **Ideas claras desde el comienzo.** El equipo debe tener claro lo que hay que hacer desde el principio, definiendo un objetivo común y teniendo una mentalidad positiva que les conduzca a tomar buenas decisiones.
- **Implementar un sistema de retroalimentación,** que permita evaluar el efecto de los cambios en el método de trabajo. Por ejemplo, llevar a cabo unas reuniones diarias para saber en qué va a trabajar cada miembro, y en qué trabajó el día anterior. La duración de estas reuniones será entre 10 y 15 minutos.
- **Trabajar de manera conjunta para mejorar el trabajo en equipo**. De esta forma los miembros se entienden mejor, tienen una visión compartida y tienen un trabajo óptimo.

El empleo de este modelo, ofrece también una serie de beneficios:

- **Estímulo del rendimiento.** Realizar estimaciones y analizar el rendimiento de los miembros para así detectar algún fallo durante la realización del proyecto.
- **Organización y colaboración.** Permitiendo que los miembros del equipo, estén comunicados en todo momento, haciendo que realicen un mejor trabajo.
- **Distribución del trabajo.** Con la utilización de los tableros, es más fácil distribuir las tareas y ver cómo va avanzando el trabajo de los miembros.

2.3 Aspectos básicos de Trello

Para entender el funcionamiento de Trello, vamos a describir cuáles son los aspectos básicos [7] y así conocer su funcionalidad.

2.3.1 Tableros

Un tablero representa el proyecto que se va a llevar a cabo, visualizando el flujo de trabajo. Como podemos observar en la ilustración 4, el tablero se va a dividir en listas las cuales hacen referencia a las diferentes etapas por las que tienen que pasar las tarjetas. Y, por último, cada lista tendrá una serie de tarjetas que indicarán la tarea a desarrollar. El proyecto puede ser desarrollado por un individuo o por un equipo.

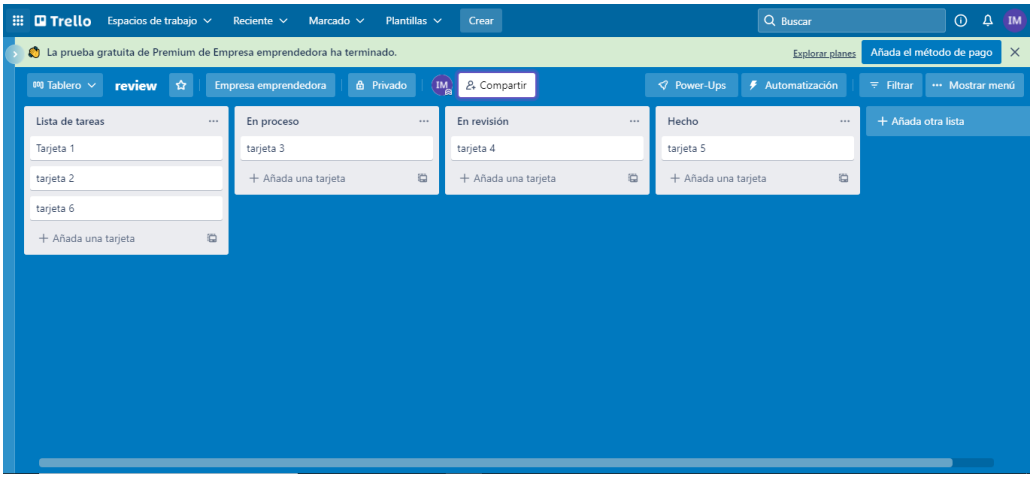

Ilustración 4: Tablero Trello

Esta herramienta permite elegir cuál va a ser la visibilidad del tablero (ilustración 5), entre ellas encontramos:

- **Privado.** El contenido del tablero solo puede ser visualizado por los miembros del tablero.
- **Público.** Cualquier persona puede acceder y visualizar el contenido del tablero. Sólo los miembros pueden editar el tablero.
- **Equipo.** El tablero tiene que añadirse a un espacio de trabajo, y entonces pueden acceder a él los miembros de la organización.
- **Espacio de trabajo.** El tablero puede visualizarse y editarse por los miembros del espacio de trabajo de la empresa emprendedora.

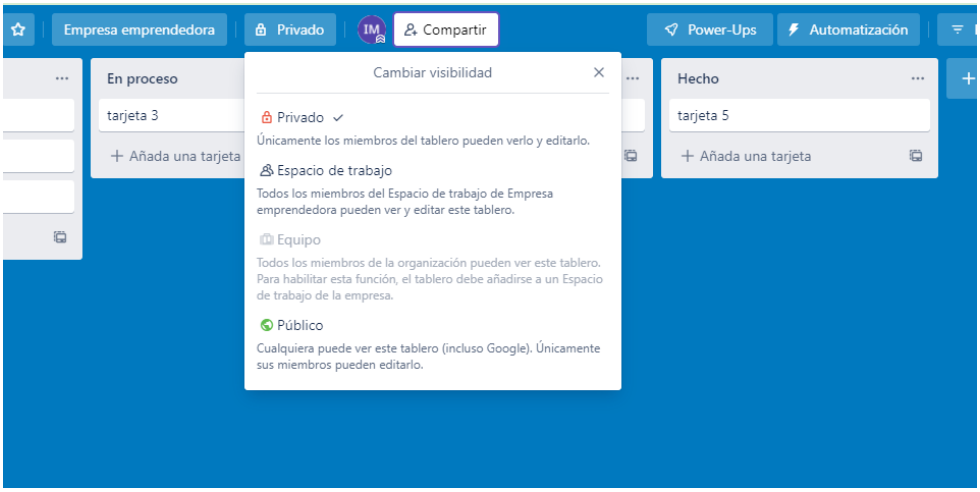

Ilustración 5: Visibilidad de un tablero

2.3.2 Listas

Las listas contendrán las tareas que tienen que llevarse a cabo. Son una forma diferente de crear un flujo de trabajo y realizar el seguimiento de cada una de las tareas. Las listas más utilizadas son las siguientes:

- **Backlog.** Es la primera lista del tablero, en la cual el equipo planea todo lo que se va a realizar a lo largo del proyecto, colocando las tarjetas en dicha lista. Cada tarea de dicha lista es asignada a un miembro y este tiene que realizar la tarea dentro del tiempo de entrega previsto.
- **Doing.** Cuando la tarjeta se mueve a dicha lista, indica que un miembro está trabajando en ella. Esta es una manera de tener al equipo comunicado.
- **Review.** Cuando se está finalizando una tarea, se mueve dicha tarjeta a esta lista para que alguien se encargue de comprobar si está bien ejecutada.
- **Done.** Si la tarea ha sido revisada correctamente y se ha dado el visto bueno, se mueve a esta lista para indicar que la tarea ha sido finalizada.

2.3.3 Tarjetas

Las tarjetas hacen referencia a la tarea que hay que realizar. Estas pueden crearse y asignarse a uno o más miembros del equipo y además pueden personalizarse para contener cualquier información que nos sea útil. En una tarjeta podemos encontrar:

Título de la tarjeta

Lo primero que encontramos para crear una tarjeta es añadir un título a esta, como podemos ver en la ilustración 6.

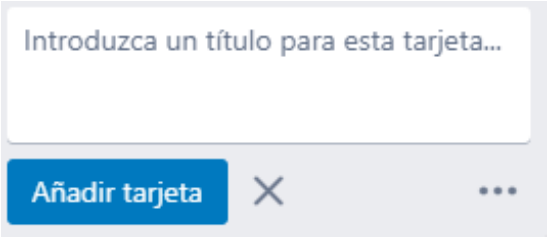

Ilustración 6: Elementos de una tarjeta: Añadir título

Descripción de la tarjeta

En este apartado se puede añadir una descripción más detallada sobre la tarea a realizar (ilustración 7). Por ejemplo, se pueden detallar los pasos a realizar para poder ejecutar la tarea.

Ilustración 7: Elementos de una tarjeta: Añadir descripción

Comentarios y actividad

En el apartado de actividad, los miembros del equipo pueden añadir comentarios en la tarea para así facilitar la comunicación entre ellos (ver ilustración 8).

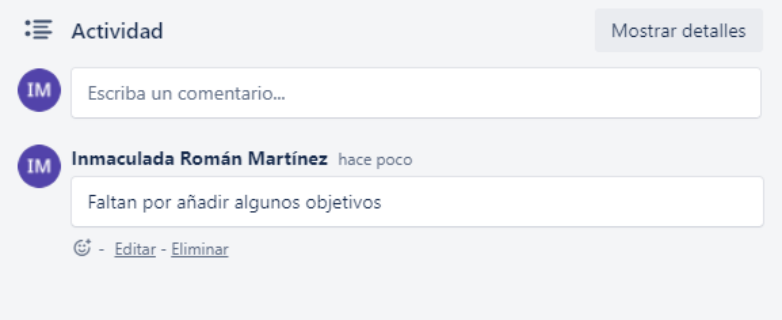

Ilustración 8: Elementos de una tarjeta: Añadir comentarios

Añadir a la tarjeta

En este apartado podemos añadir cualquier tipo de información a la tarjeta, entre ellas encontramos:

- Añadir más de un miembro para llevar a cabo la ejecución de la tarea.
- Añadir etiquetas (colores), para distinguir las tarjetas.
- Crear subtareas.
- Añadir checklist a las tarjetas que tengan subtareas para así llevar un control de que se cumplen todos los pasos.
- Añadir una fecha de vencimiento a las tarjetas para asegurarse que tienen que ser entregadas antes de la fecha propuesta.
- Adjuntar varios archivos desde el ordenador o desde la nube.
- Añadir una portada para llamar la atención de los usuarios.

Power-Ups

Los Power-Ups son funciones o acciones adicionales que se pueden añadir a los tableros o a las tarjetas. En las ilustraciones 9 y 10 encontramos algunos Power-Ups. Algunos de los más utilizados son:

- Votar por sus tarjetas favoritas.
- Crear un calendario para ver todas las fechas de entrega de las tareas del tablero.
- Añadir enlaces de Google Drive y adjuntar archivos.
- Mandar tarjetas de Trello directamente a canales de Slack o recibir notificaciones.
- Añadir Hangouts para realizar video llamadas entre los miembros del equipo.

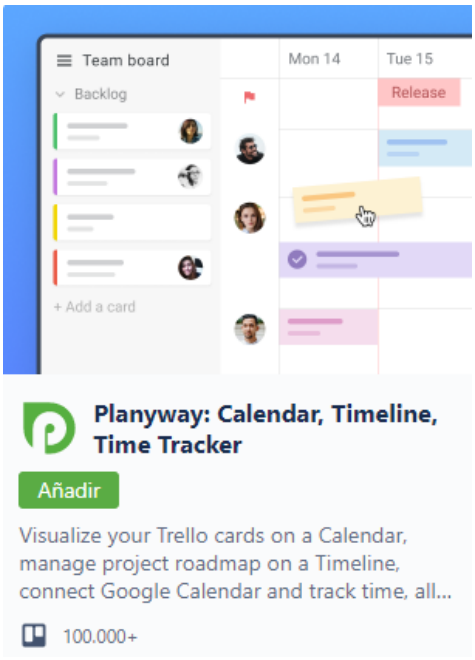

Ilustración 9: Power-Up Google Calendar

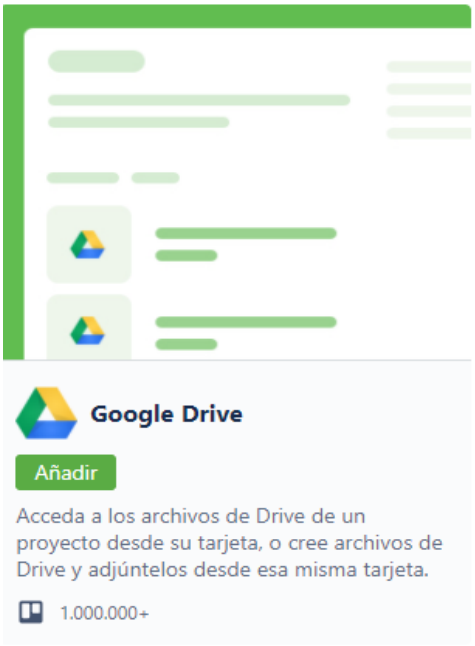

Ilustración 10: Power-Up Google Drive

Butler

Butler es una forma de automatizar el trabajo, de forma que se crean unas reglas para que no se olvide la realización de ninguna tarea. Algunas de las reglas pueden ser:

- Marcar la fecha de vencimiento automáticamente.
- Añadir una etiqueta.

- Eliminar a miembros de un equipo.

Acciones

Las acciones que se pueden realizar sobre una tarjeta (Ilustración 11) son:

- Mover una tarjeta de una lista a otra y añadir la posición en que se desea que se encuentre la tarjeta.
- Copiar una tarjeta en la lista deseada, pudiendo modificar la posición en la que se desea que se encuentre.
- Convertir una tarjeta en una plantilla.
- Realizar un seguimiento sobre la tarjeta.
- Archivar la tarjeta.
- Enviar al tablero.
- Compartir la tarjeta.
- Eliminar la tarjeta.

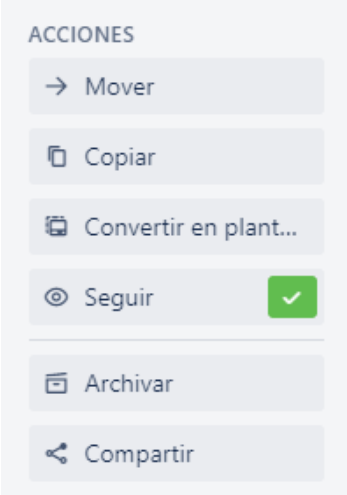

Ilustración 11: Acciones sobre una tarjeta

2.3.4 Menú

En el lateral derecho del tablero podemos encontrar un menú (ver ilustración 12), el cual permite cambiar la configuración del tablero. Algunas de las acciones que se pueden realizar son:

- Añadir o eliminar a miembros al equipo.
- Mostar y editar la información del tablero.

- Cambiar el fondo del tablero, para personalizarlo.
- Buscar alguna tarjeta en concreto.
- Habilitar los Power-Ups.
- Añadir Butler.

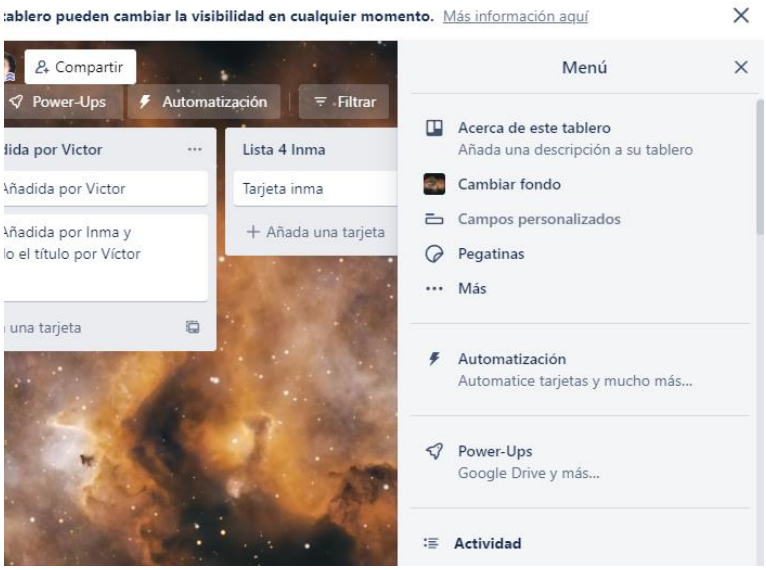

Ilustración 12: Menú de un tablero

2.4 Integración con otras aplicaciones

Trello permite que el trabajo realizado en aplicaciones externas pueda ser controlado y supervisado por el mismo. Esto es posible gracias a la utilización de los Powers-ups que Trello facilita. De esta manera, se pueden personalizar los flujos de trabajo, ver la actividad de cada miembro, ver el rendimiento, utilizar herramientas como Google Drive o Dropbox y enviar correos electrónicos a tableros o tarjetas.

Además, como se ha mencionado anteriormente, se pueden añadir distintos Powers-ups a las tarjetas para destacar información relevante.

2.4.1 Dropbox

Dropbox es un canal de comunicación donde fácilmente los usuarios pueden adjuntar archivos o carpetas. La idea de esta integración, es que el equipo permanezca siempre comunicado, pudiendo ver con una vista previa el trabajo realizado por otro miembro en ese instante. Algunas cosas que se pueden hacer con este power-up son:

- Ver el contenido de la tarjeta con las vistas en miniatura.

- Visualizar documentos que se tienen en Dropbox sin salir de Trello.
- Crear una tarjeta desde un documento en Dropbox.

2.4.2 Gmail

La idea de integrar Gmail básicamente consiste en mantener comunicados en todo momento la bandeja de entrada del correo con los tableros. Instalando este power-up, los miembros pueden convertir los correos en tarjetas de Trello, estando dentro de Gmail. Desde la bandeja de entrada se puede:

- Seleccionar un tablero o una lista para añadir una tarjeta.
- Modificar la fecha de vencimiento.

2.4.3 Slack

Añadiendo el Power-up Slack de Trello se va a conseguir que ambas aplicaciones estén vinculadas y de esta forma conseguir que todas las ideas queden claras y puedan ser realizadas exitosamente. Nos va a permitir:

- Añadir recordatorios a las tarjetas.
- Compartir información con los demás miembros.
- Configurar alertas cuando se realice alguna actividad.

2.5 Automatización

Trello ofrece un sistema de automatización integrado que permite automatizar las acciones para que el trabajo que se va realizando vaya organizado y no se produzcan errores. Con la automatización se pueden crear:

- **Reglas.** Consiste en definir una acción que se ejecutará automáticamente. Por ejemplo, marcar una tarea como completada, cuando se haya marcado la fecha de vencimiento.
- **Botones.** Se pueden crear botones que permitan desencadenar una serie de acciones haciendo click en una tarjeta o en un tablero. Por ejemplo, se puede crear un botón en el tablero que permita obtener cuáles son todas las tareas que hay atrasadas en él.

- **Comandos de fecha de vencimiento y de calendario.** Son avisos, que se pueden configurar para que se ejecuten de forma diaria, semanal, etc, permitiendo así que el trabajo sea exitoso. Por ejemplo, que todos los días se ordenen las tarjetas de una lista por fecha de vencimiento.

2.6 Precios

La herramienta Trello cuenta con dos versiones que se pueden elegir para un mejor desarrollo.

Versión gratuita

- Tiene un límite de 10 tableros por equipo.
- No hay límite para los tableros personales, las listas y las tarjetas.
- La capacidad de almacenamiento por archivo adjunto es de 10MB.

Versión de pago

- Tableros por equipo ilimitados, al igual que las tarjetas y las listas.
- La capacidad de almacenamiento por archivo adjunto es de 250MB.
- Asistencia por parte del equipo Trello.

2.7 Ventajas de Trello

Ahora que ya tenemos una primera idea de cómo funciona de Trello, vamos a mencionar algunos de los aspectos positivos de esta herramienta.

- Aplicación simple e intuitiva.
- Puede ser editada y compartida en tiempo real por los miembros del equipo.
- Mantiene al equipo siempre conectado y trabajando de forma cooperativa.
- Tiene un sistema de notificaciones que informa cuando se realiza algún cambio en el tablero.
- Dispone de una función de etiquetado de colores para destacar y organizar visualmente las tareas.
- Tiene un motor de búsquedas bastante rápido.
- Permite personalizar la apariencia de los tableros.
- Tiene la opción de poder compartir los tableros.

- Los tableros pueden ser públicos o privados.
- La conexión es segura.
- Hay una confidencialidad de los datos con copias de seguridad cifradas.
- Una tarea puede ser asignada a más de un miembro del equipo.
- Es una herramienta que continuamente se está actualizando.

2.8 Desventajas de Trello

Al igual que hemos comentado los aspectos positivos, como toda herramienta, también tiene algunos aspectos negativos, los cuales son los siguientes:

- No es útil cuando no se utilizan tableros Kanban.
- No es muy útil cuando existe un gran número de tareas.
- Ineficiente para proyectos grandes.
- No se pueden mostrar dependencias entre tareas.
- No cuenta con la generación de informes, ni se puede hacer un seguimiento del tiempo y ni de los gastos.

2.9 Alternativas a Trello

2.9.1 Jira Software

Jira [8] es una herramienta de desarrollo de software que permite la gestión de tareas. Fue creada especialmente para empresas que trabajan con proyecto grandes. Esta herramienta es muy buena para empresas que quieren trabajar con cualquier metodología ágil, ya sea Kanban, Scrum, etc.

Ilustración 13: Logo Jira Software

Algunas de las diferencias [9] que encontramos con Trello son:

- Jira se puede alojar en servidores y en la nube, mientras que Trello solo en la nube.

- Jira se encarga de la gestión de tareas, realiza un seguimiento del tiempo, gestiona los recursos, además de elaborar informes y gestionar documentos; Trello sin embargo no dispone del seguimiento del tiempo, ni la realización de informes

Precios

Esta herramienta ofrece distintos planes [10], los cuales vamos a mencionar a continuación.

El plan gratuito (ilustración 14), que permite hasta 10 usuarios.

Ilustración 14: Comparativa precios Jira

El paquete Estándar (ilustración 15) permite de 1 hasta 100 usuarios, y solo cuesta 7’18€ (7’50$) al mes, ofreciendo además 7 días de prueba gratuita.

Ilustración 15: Comparativa precios Jira

Y, por último, ofrece un paquete Premium (ilustración 16), el cual cuesta unos 12’88€ (14’50$) al mes por usuario y permite de 1 a 100 usuarios.

Ofrece almacenamiento ilimitado, soporte las 24 horas del día con respuesta inmediata, archivar los proyectos más antiguos y reforzar la seguridad de la dirección IP, entre otros.

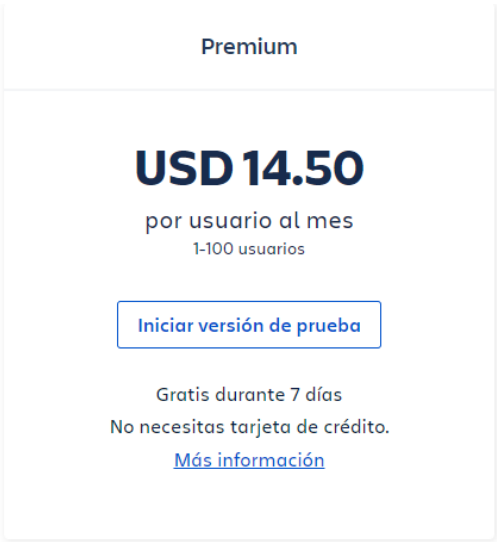

Ilustración 16: Comparativa precios Jira

2.9.2 Asana

Es una aplicación centrada en administrar y gestionar las tareas a llevar a cabo en un proyecto, permitiendo que los equipos compartan la información y planificación de las tareas para cumplir con los objetivos [11]. Esta aplicación utiliza métodos similares a Trello, entre ellos la utilización de un tablero Kanban para visualizar el flujo de trabajo.

Ilustración 17: Logo Asana

A diferencia de Trello, se introducen nuevas características o mejoras que hacen a este software más funcional.

- Se introduce un sistema de gestión de dependencias entre tareas.
- Se añade una línea de tiempo, para conocer los plazos de entrega de cada tarea.
- También se introduce un calendario para evitar conflictos entre los miembros y la realización de las tareas.

Para empresas pequeñas con proyectos sencillos es recomendable utilizar Trello, pero para empresas complejas donde varios equipos trabajan en proyectos complejos, es mejor utilizar Asana.

Precios

Esta herramienta dispone de tres paquetes con diferentes precios [12], dependiendo de los usuario o empresas que vayan a hacer uso de ella y las prestaciones que quieran.

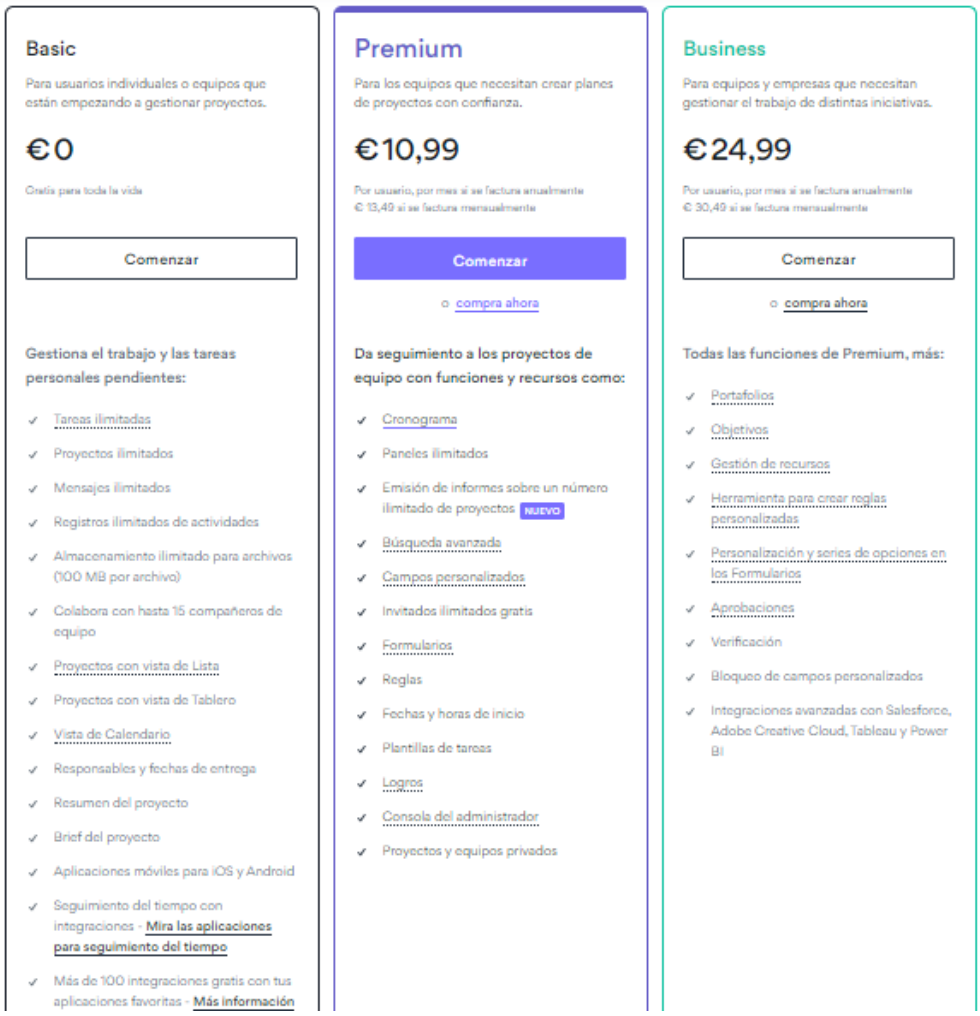

Ilustración 18: Comparativa precios Asana

2.9.3 KanbanFlow

KanbanFlow [13] es un software de gestión de proyectos que facilita la colaboración en equipo, la visualización de trabajo y un seguimiento de tiempo. A priori es un tablero Kanban que ofrece las mismas prestaciones que la aplicación de Trello.

Ilustración 19: Logo KanbanFlow

Pero KanbanFlow introduce las siguientes mejoras:

- Añadir a las listas un número límite de tarjetas que pueden asignarse.
- Buscar tarjetas por color, etiqueta, usuario y fecha de vencimiento.
- Se puede realizar un seguimiento del progreso de las actividades.
- Permite la realización de informes.
- Se puede realizar un seguimiento del tiempo, de las horas y de los gastos.

Precios

A la hora de elegir, esta herramienta nos ofrece dos paquetes diferentes:

KanbanFlow ofrece una versión gratuita que es bastante completa y una versión Premium que cuesta alrededor de 4’79€ al mes.

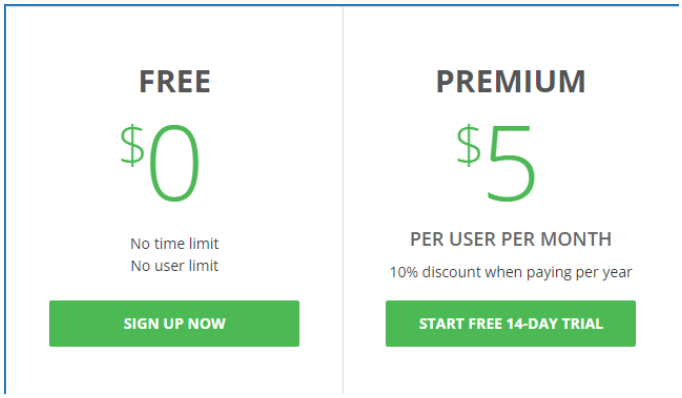

Ilustración 20: Comparativa precios KanbanFlow

La diferencia entre ambos paquetes es que la versión gratuita está más limitada en la realización de análisis e informes, no pueden realizarse la integración de otras herramientas y no se puede llevar un control de seguridad, entre otras.

2.9.4 Basecamp

Como las herramientas anteriores, Basecamp es una herramienta pensada para que los equipos trabajen con ella y sea útil para la colaboración y la gestión del flujo de trabajo.

Ilustración 21: Logo Basecamp

La idea de esta herramienta es dividir todo el trabajo que tiene que realizar un equipo en proyectos bien organizados, de manera que mejore el trabajo y los resultados sean más eficaces.

Algunas de las herramientas [14] que se pueden encontrar dentro de cada proyecto son:

- Tablero de mensajes, donde se puede ir actualizando el progreso, destacar algunas ideas, etc.
- Compartir archivos con los miembros del equipo.
- Asignar las tareas a los miembros del equipo.
- Establecer las fechas de entrega.
- Planificar los calendarios.
- Generar informes.

Precios

Esta herramienta ofrece dos paquetes diferentes:

Un primer paquete limitado para uso personal, estudiantes o autónomos, pero totalmente gratuito (ilustración 22).

Ilustración 22: Comparativa precios Basecamp

Y otro paquete más completo (ver ilustración 23) para empresas, que cuesta alrededor de 94’75€ al mes.

Ilustración 23:Comparativa precios Basecamp

3. ESPECIFICACIÓN DE LA API DE TRELLO

La API de Trello permite tener acceso a los datos y funciones que se van a describir a continuación e integrarlos en otras aplicaciones. Para ello, la API se encuentra conectada a un servidor con una base de datos, el cual nos da la oportunidad de realizar cualquier petición http (POST, PUT, GET, DELETE) teniendo en cuenta que podemos realizar aquellas solicitudes que ya aparezcan definidas en la documentación oficial de la API.

Para el desarrollo del proyecto, nos vamos a centrar en la realización de las peticiones GET, ya que sólo tenemos que cuantificar la información obtenida.

3.1 Información sobre la API de Trello

3.1.1 Autenticación y autorización

Para hacer uso de la API de Trello, tenemos dos formas.

I. La primera forma es acceder a la API sin tener que autenticarnos. De esta forma sólo se pueden realizar consultas GET para obtener la información, pero solo se puede hacer si los tableros son públicos.
II. La segunda forma de acceder es autenticándonos. Para ello, Trello utiliza una clave y un token, que es específico para cada usuario. Para obtener dicha clave hay que acceder a la siguiente url: ‘https://trello.com/app-key’, aceptar los términos y de forma automática se genera nuestra clave (ver ilustración 24 y 25). Cuando la clave API es generada, aparece la palabra ‘token’ en la cual al pulsarla se generará. Aquí podemos realizar cualquier petición http.

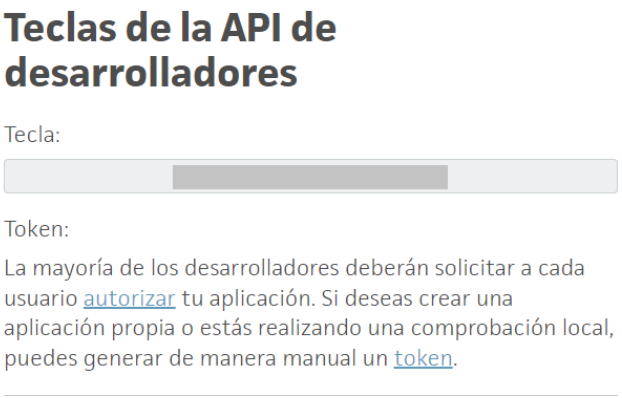

Ilustración 24: Obtener API Key de Trello

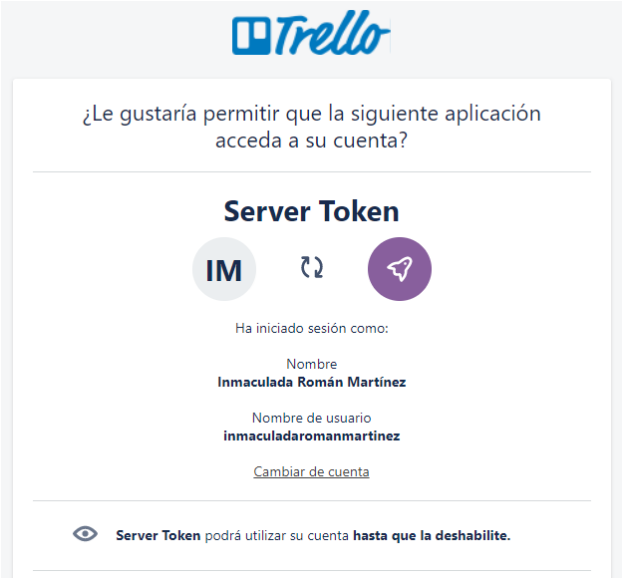

Ilustración 25: Generar token para acceder a Trello

3.1.2 Hacer llamadas a la API

A continuación, vamos a ver las diferentes llamadas que se pueden hacer a Trello, sin entrar mucho en detalle, ya que todo se encuentra bien explicado en la documentación de la API [15].

La dirección url se va a dividir en dos partes

I. La primera es la parte fija de la url, la cual se va a dirigir a la API de Trello → 'https://api.trello.com/1/'
II. Y la segunda parte de la url va a ir variando, dependiendo si queremos ver los tableros, las acciones o las tarjetas, entre otras. Por ejemplo → 'https://api.trello.com/1/actions/{id}

Ha de tenerse en cuenta que, si el tablero es privado o la petición a realizar es del tipo PUT, DELETE o POST será necesario pasar la clave y el token en la url.

3.1.3 Tableros

Algunas de las propiedades que tiene un tablero son:

- Identificador del tablero: id.
- Nombre del tablero: name.
- Descripción: desc.
- Tablero cerrado: closed.
- Identificador del miembro que creó el tablero: idMemberCreator.
- Identificador de la organización a la que pertenece: idOrganization.
- Url del tablero: url.

Como hemos mencionado anteriormente, para obtener información relacionada con un tablero, depende de si el usuario está o no autenticado. Si el usuario no está autenticado:

- **GET** → https://api.trello.com/1/boards/{idTablero} Con esta petición, el usuario le está pidiendo a la API, que le devuelva la información perteneciente a dicho tablero. En caso de que el tablero no sea público, al realizar la petición dará un error de autenticación.

Si el usuario ya está autenticado, igualmente tiene que pedirle a la API que le devuelva en una lista todos los tableros que le pertenecen. Para ello, tiene que realizar:

- **GET** →https://api.trello.com/1/members/me/boards?key={yourKey}&token={yourToken} El usuario solicita que con su correspondiente clave y token, la api le devuelva todos los tableros.

Una vez obtenida la información del tablero, ya se puede realizar cualquier petición http relacionada con el tablero. Para las peticiones POST, PUT y DELETE es necesario añadir a la llamada el key y token si el usuario no está autenticado en la aplicación.

Algunos de los métodos más usados son:

- Crear un tablero: **POST**→ https://api.trello.com/1/boards/?name={name} &key={APIKey}&token={APIToken}.
- Actualizar un tablero; es necesario pasar en la url la clave y el token del miembro que pertenece al tablero, en caso de no estar autenticado: **PUT** → https://api.trello.com/1/boards/{idBoard}?key={APIKey}&token={APIToken}.
- Eliminar un tablero, indicando la clave y el token del miembro, en caso de no estar autenticado en la aplicación: **DELETE** → https://api.trello.com/1/boards/{idBoard}?key={APIKey}&token={APIToken}.
- Obtener la información de un campo en concreto, como por ejemplo el nombre, la fecha de la última actividad, entre otros: **GET** → https://api.trello.com/1/boards/{idBoard}/{field}.
- Obtener las acciones que se han realizado en un tablero: **GET** → https://api.trello.com/1/boards/{idBoard}/actions.
- Obtener la información específica de una tarjeta indicando el identificador de esta: **GET** → https://api.trello.com/1/boards/{idBoard}/cards/{idCard}.
- Obtener información de todas las tarjetas que pertenecen a un tablero: **GET** → https://api.trello.com/1/boards/{idBoard}/cards.
- Crear una lista en un tablero: **POST** → https://api.trello.com/1/boards/{idBoard}/lists?name={name}.
- Obtener las listas que han sido definidas en un tablero: **GET** → https://api.trello.com/1/boards/{idBoard}/lists.
- Obtener información de una lista en concreto del tablero: **GET** → https://api.trello.com/1/boards/{idBoard}/lists/{idList}.
- Obtener información sobre las membresías a la que pertenecen los usuarios: **GET** → https://api.trello.com/1/boards/{idBoard}/memberships.
- Crear una etiqueta en un tablero: **POST** → https://api.trello.com/1/boards/{idBoard}/lists/labels?name={name}.
- Obtener los miembros que pertenecen a un tablero: **GET** → https://api.trello.com/1/boards/{idBoard}/members.
- Invitar a un miembro, por email: **PUT** → https://api.trello.com/1/boards/{idBoard}/members?email={email}.

- Añadir un miembro al tablero, indicando el tipo de miembro (admin, normal, observer): **POST** → https://api.trello.com/1/boards/{idBoard}/members/{idMember}?type={type}.
- Eliminar a un miembro del tablero: **DELETE** → https://api.trello.com/1/boards/{idBoard}/members/{idMember}.

3.1.4 Listas

La API de Trello devuelve al usuario las listas de un tablero. Algunos de los campos que podemos encontrar en las listas son:

- Nombre de la lista: name.
- Lista cerrada: closed.
- Id del tablero al que pertenece: idBoard.
- Posición: pos.

Los métodos más destacados son:

- Obtener información de una lista en concreto: **GET** → https://api.trello.com/1/lists/{idList}.
- Actualizar una lista en concreto: **PUT** → https://api.trello.com/1/lists/{idList}.
- Crear una nueva lista en un tablero: **POST** → https://api.trello.com/1/lists?name={name}&idBoard={idBoard}.
- Listar las tarjetas que pertenecen a una lista en concreto: **GET** → https://api.trello.com/1/lists/{idList}/cards.
- Archivar todas las tarjetas en una lista: **POST** → https://api.trello.com /1/lists/{idList}/archiveAllCards.
- Mover todas las tarjetas a una lista: **POST** → https://api.trello.com /1/lists/{idList}/moveAllCards?idBoard={idBoard}&idList={idList a la que se mueve}.
- Archivar o desarchivar una lista: **PUT** → https://api.trello.com /1/lists/{idList}/closed.
- Mover una lista a otro tablero: **PUT** → https://api.trello.com /1/lists/{idList}/idBoard?value={idBoard}.

- Actualizar un campo de una lista (name, pos, etc): **PUT**→ 'https://api.trello.com /1/lists/{idList}/{field}.
- Obtener información de las acciones realizadas sobre una lista: **GET** → 'https://api.trello.com /1/lists/{idList}/actions.
- Obtener información sobre el tablero en el que se encuentra la lista: **GET** → 'https://api.trello.com /1/lists/{idList}/board.
- Obtener las tarjetas que se encuentran en la lista: **GET** → 'https://api.trello.com /1/lists/{idList}/cards.

3.1.5 Tarjetas

Las tarjetas son las tareas que hay que realizar en un tablero. Algunos de los campos de una tarjeta son:

- Identificador de la tarjeta: id.
- Nombre de la tarjeta: name.
- Descripción: desc.
- Posición de la tarjeta en la lista: pos.
- Fecha de la última actividad: dateLastActivity.
- Tarjeta archivada: closed.
- Id del tablero al que pertenece: idBoard.
- Id de la lista en la que se encuentra: idList.

Algunas de las peticiones más destacadas son:

- Crear una nueva tarjeta: **POST** → https://api.trello.com /1/cards?idList={idList}.
- Obtener información de una tarjeta a partir de su identificador: **GET** → https://api.trello.com /1/cards/{idCard}.
- Actualizar una tarjeta a partir de su identificador: **PUT** → https://api.trello.com /1/cards/{idCard}.
- Eliminar una tarjeta a través de su identificador: **DELETE** → https://api.trello.com /1/cards/{idCard}.
- Obtener información de algún campo de una tarjeta (id, closed, desc, etc): **GET** → https://api.trello.com /1/cards/{idCard}/{field}.

- Obtener las acciones realizadas en una tarjeta: **GET** → https://api.trello.com /1/cards/{idCard}/actions.
- Obtener información del tablero al que pertenece la tarjeta: **GET** → https://api.trello.com /1/cards/{idCard}/board.
- Crear una lista de verificación en una tarjeta: **POST** → https://api.trello.com /1/cards/{idCard}/checklists.
- Obtener información de la lista de verificación de una tarjeta: **GET** → https://api.trello.com /1/cards/{idCard}/checklits.
- Obtener información de la lista a la que pertenece la tarjeta: **GET** → https://api.trello.com /1/cards/{idCard}/list.
- Obtener los miembros que tienen que realizar la tarjeta: **GET** → https://api.trello.com /1/cards/{idCard}/members.
- Actualizar un comentario de una tarjeta: **PUT** → https://api.trello.com /1/cards/{idCard}/actions/{idAction}/comments?text={text}.
- Eliminar el comentario de una tarjeta: **DELETE** → https://api.trello.com /1/cards/{idCard}/actions/{idAction}/comments.
- Añadir un comentario a una tarjeta: **POST** → https://api.trello.com /1/cards/{idCard}/actions/{idAction}/comments?text={text}.
- Añadir un miembro a una tarjeta: **POST** → https://api.trello.com /1/cards/{idCard}/{idMember}.
- Eliminar un miembro de una tarjeta: **DELETE** → https://api.trello.com /1/cards/{idCard}/idMembers/{idMember}.

3.1.6 Acciones

Las acciones son útiles para obtener toda la actividad que se ha realizado en el tablero. Los campos que componen las acciones son:

- Identificador de la acción: id.
- Tipo de acción: type.
- Identificador del miembro que realizó la acción: idMemberCreator.
- Información perteneciente a la acción: data.
- Fecha de cuando ocurrió la acción: date.

Algunas de las peticiones más destacados son:

- Obtener información de una acción concreta: **GET** → https://api.trello.com /1/actions/idAction.
- Actualizar una acción, por ejemplo, actualizar un comentario: **PUT** → https://api.trello.com /1/actions/{idAction}/text?value={value}.
- Eliminar una acción indicando su identificador: **DELETE** → https://api.trello.com /1/actions/{idAction}.
- Obtener información de algún campo específico de la acción, por ejemplo, idMemberCreator, data, type, etc: **GET** → https://api.trello.com /1/actions/{field}.
- Obtener información del tablero al que pertenece la acción: **GET** → https://api.trello.com /1/actions/{idAction}/board.
- Obtener información de la tarjeta perteneciente a la acción: **GET** → https://api.trello.com /1/actions/{idAction}/card.
- Obtener información de la lista a la que pertenece la acción: **GET** → https://api.trello.com /1/actions/{idAction}/list.
- Obtener información del miembro que creó la acción: **GET** → https://api.trello.com /1/actions/{idAction}/memberCreator.
- Obtener información del miembro que realizó la acción, no el miembro creador: **GET** → https://api.trello.com /1/actions/{idAction}/member.
- Obtener información de la organización de una acción: **GET** → https://api.trello.com /1/actions/{idAction}/organization.
- Crear un emoji para una acción: **POST** → https://api.trello.com /1/actions/{idAction}/reactions.
- Eliminar un emoji de una acción: **DELETE** → https://api.trello.com /1/actions/{idAction}/reactions/{id}.

3.1.7 Miembros

Los miembros son aquellos usuarios que pertenecen a un tablero, de los cuales podemos averiguar la actividad que realizan en él.

Algunos campos que podemos encontrar en un miembro son:

- Identificador: id.
- Si la actividad en el tablero está bloqueada: activityBlocked.

- Nombre completo: fullName.
- Iniciales: initials.
- Nombre usuario: username.
- Estado: status.
- Tipo de miembro: normal, admin, observer.
- Identificador de la organización a la que pertenece: idOrganizations.
- Tableros a los que pertenece: idBoards.

Algunas de las solicitudes que podemos destacar son:

- Obtener información de un miembro: **GET** → https://api.trello.com/1/members/{idMember}.
- Actualizar un miembro indicando su identificador: **PUT** → https://api.trello.com/1/members/{idMember}.
- Obtener información de un campo específico de un miembro: **GET** → https://api.trello.com/1/members/{idMember}/{field}.
- Obtener las acciones que ha realizado un miembro: **GET** → https://api.trello.com/1/members/{idMember}/actions.
- Obtener los tableros a los que pertenece el miembro: **GET** → https://api.trello.com/1/members/{idMember}/boards.
- Obtener los tableros a los que un miembro ha sido invitado: **GET** → https://api.trello.com/1/members/{idMember}/boardsInvited.
- Obtener las tarjetas en el que está el miembro: **GET** → https://api.trello.com/1/members/{idMember}/cards.
- Recibir notificaciones de un miembro cuando realice algún cambio: **GET** → https://api.trello.com/1/members/{idMember}/notifications.
- Obtener las organizaciones a las que pertenece un miembro: **GET** → https://api.trello.com/1/members/{idMember}/organizations.

4. TECNOLOGÍAS UTILIZADAS

Para la realización del proyecto, se debe de tener claro de qué partes se compone. Por un lado, encontramos el Backend, más concretamente el servidor, que será el encargado de realizar las peticiones a la API y almacenar la información. Por otro lado, tenemos el Frontend que será el encargado de solicitar al backend la información almacenada y mostrar al usuario dicha información.

4.1 Back-End

El backend [16] es el que se encarga básicamente de que una página web funcione. Contiene todas las acciones necesarias para que la página funcione, las cuales no son visibles al usuario. Algunas de las acciones que tiene son la comunicación con una base de datos, la comunicación con una API, etc.

Para el desarrollo del backend se puede emplear varios lenguajes de programación. Como podemos observar en la ilustración 26, algunos de los lenguajes más conocidos son Java, C, Python, PHP, etc.

Ilustración 26: Lenguajes de programación del Backend

4.1.1 Tecnologías

En este apartado, se van a mencionar cuáles son las tecnologías que se han utilizado para la implementación del back-end. Entre ellas encontramos:

Spring WebClient

Spring WebClient es una interfaz que permite consumir servicios web de manera asíncrona y sin bloqueo mediante solicitudes http [17]. Forma parte de la biblioteca de Spring Webflux y es conocido como la alternativa a RestTemplate [18]. Es necesario para poder consumir una API.

Ilustración 27: Icono Spring WebClient

Esta herramienta de cliente nos permite realizar diferentes consultas http, entre ellas encontramos:

- **Get:** permite obtener información sobre un objeto específico.
- **Put:** permite modificar o actualizar información de un objeto.
- **Post:** permite crear un nuevo objeto.
- **Delete:** permite eliminar un objeto.

Para poder hacer uso de WebClient, tenemos que crear un objeto de este tipo y añadirle la URI de la API a la cual tiene que comunicarse para poder obtener la información.

Una vez realizada la llamada correspondiente, es importante saber que el resultado de la llamada se devuelve en un objeto JSON y que puede ser de tipo Mono o Flux, pertenecientes a la biblioteca Reactor [19], la cual permite que el paso de mensajes se realice de forma efectiva.

- **Mono:** significa que la llamada realizada contiene 0 o 1 elemento.
- **Flux:** cuando la llamada realizada contiene de 0 a N elementos.

4.1.2 Herramientas

Para utilizar las tecnologías mencionadas anteriormente es importante conocer el entorno de desarrollo sobre que el vamos a trabajar. Para poder desarrollar un proyecto basado en Spring Boot y haciendo uso de Spring WebClient vamos a utilizar:

Spring Tools Suite

Spring Tools Suite [20] es una herramienta de desarrollo utilizada para el desarrollo de proyectos Web con Java y basados en Spring. Esta herramienta viene bajo la licencia de Eclipse y además es de código abierto.

Postman

Para saber si la comunicación del servidor con la API de Trello es correcta, vamos a emplear Postman. Esta es una herramienta [21] utilizada para realizar peticiones http a APIS. Es una forma sencilla de realizar test, para ver si la comunicación es correcta. Existen aplicaciones nativas para todos los sistemas operativos, aunque para Linux existe la versión beta.

4.2 Front-End

El front-end [22] es la aplicación cliente, con la cual interaccionan los usuarios. Consiste en el diseño del sitio web, el cual permite que el código se ejecute en el navegador aplicando estilos, colores, etc, de forma que la página sea atractiva y llamativa para el usuario. Esto se consigue con la utilización de CSS y HTML. Para el desarrollo de la aplicación cliente se pueden utilizar distintos lenguajes de programación (ilustración 28), como Angular, Vue, etc.

Ilustración 28: Lenguajes front-end

4.2.1 Tecnologías

En nuestro caso para el desarrollo del front-end la tecnología utilizada ha sido Angular, la cual vamos a detallar a continuación.

Angular

Angular [23] es un framework utilizado para la creación de aplicaciones web de una sola página, basadas en TypeScript. Es de código abierto y se adapta completamente a las necesidades del desarrollador.

Ilustración 29: Icono Angular

Angular fue desarrollado por Google, y su primera versión fue lanzada en el año 2012. Este framework sigue la arquitectura Modelo-Vista-Controlador (MVC), la cual consiste en:

- **Modelo:** es el que recibe la información del controlador.
- **Vista:** consiste en la representación de la información.
- **Controlador:** es el encargado de la comunicación con el modelo.

En pocas palabras, Angular establece las rutas que se encargarán de la comunicación con el servidor y, además, permite la aplicación de estilos y formatos para cambiar la vista del navegador. Se ha elegido dicha tecnología, por el aprendizaje que conlleva, ya que anteriormente no lo había visto y por el gran uso que tiene actualmente.

Este framework es comúnmente utilizado por las diferentes ventajas que ofrece, entre ellas:

- Ofrece plantillas para que el código sea más limpio y no sea repetitivo.
- Es un framework escalable y modular.
- Además de ejecutarse en navegadores, es compatible con dispositivos móviles.

- Separa el código de la interfaz de usuario (front-end) y el de la lógica de negocio (black-end).
- Tiene un enlace bidireccional de datos, permitiendo enlazar JavaScript y HTML.
- Tiene sus propias directivas HTML.

Un aspecto importante a conocer de Angular es que este está compuesto por componentes y servicios. Los componentes son elementos que forman el proyecto, es decir, para que una aplicación funcione está compuesta por 3 componentes.

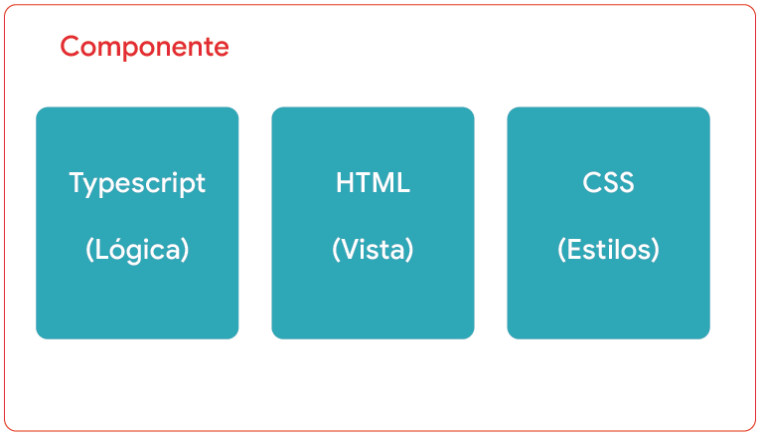

Ilustración 30: Componentes de Angular

- **App.component.ts:** es el archivo que contiene la lógica de la aplicación, incluyendo una clase con sus respectivas propiedades en lenguaje TypeScript.
- **App.component.html:** es el archivo que se va a visualizar al usuario, con código en HTML.
- **App.component.css:** Archivo en el que se incluyen los estilos que tendrá la aplicación en formato CSS.

Para poder hacer uso de Angular necesitamos:

Angular Cli

Angular Cli es una herramienta de interfaz de línea de comandos [24] que se utiliza para desarrollar aplicaciones de Angular directamente desde la línea de comandos. Es una herramienta nos va a ayudar a la instalación, generación y depuración de código de cualquier aplicación con Angular. Para la instalación de esta herramienta es necesario la instalación del paquete Node.js [25].

Angular Cli nos va a permitir hacer todo lo mencionado anteriormente mediante comandos, los cuales vamos a identificar a continuación:

- **Instalar Angular Cli** → npm install –g @angular/cli
- **Crear proyecto** → ng new nuevo-proyecto
- **Dirigirse a la carpeta de trabajo** → cd nuevo-proyecto
- **Ejecutar la aplicación** → ng serve

4.2.2 Herramientas

Las herramientas empleadas para la utilización de las tecnologías mencionadas anteriormente son:

Visual Studio Code

Es un editor de código abierto y gratuito que fue desarrollado por Microsoft [26]. Es una herramienta bastante útil que facilita trabajar con múltiples lenguajes de programación y además está disponible para todos los sistemas operativos. La instalación de esta herramienta es rápida y fácil de utilizar.

GitLab

Gitlab [27] es una herramienta que fue desarrollada en 2011 y además es un repositorio de Git que permite ver cómo van avanzando las tareas en un proyecto. Git es un sistema de control de versiones que permite gestionar que cambios se han realizado en cada archivo. Esta herramienta se utiliza tanto para el servidor como para el cliente.

Ilustración 31: Logo GitLab

Beneficios de utilizar esta aplicación:

- Es una aplicación que se encuentra en la nube.
- Es una herramienta fácil de configurar.
- Facilita el trabajo de los miembros.
- Crea un flujo de trabajo optimizado.

Bootstrap

Bootstrap [28] es un framework CSS utilizado para la creación de aplicaciones web en el cliente. Utiliza CSS y JavaScript para diseñar los elementos de una página HTML. Básicamente, lo que busca es que las páginas diseñadas se puedan adaptar a cualquier dispositivo.

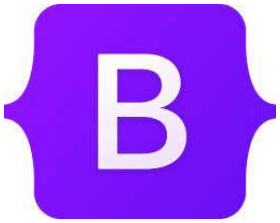

Ilustración 32: Logo Bootstrap

4.3 Lenguajes de Programación

4.3.1 Java

Java [29] es un lenguaje de programación comúnmente utilizado hoy en día para el desarrollo de aplicaciones y cuya descarga es gratuita. Permite que los programas puedan ser distribuidos en cualquier plataforma, debido a la creación de una máquina virtual la cual crea un puente entre la aplicación y el ordenador.

Ilustración 33: Logo Java

Entre las características de este lenguaje, encontramos:

- Lenguaje fácil de aprender, en comparación con otros lenguajes como C.
- Es un lenguaje orientado a objetos.
- Ofrece bibliotecas para que los programas puedan ser distribuidos y, a su vez, es independiente, pudiendo ejecutarse en cualquier tipo de hardware.
- Permite la ejecución de varias tareas a la vez.

4.3.2 TypeScript

TypeScript [30] es un lenguaje de programación que se creó a un nivel superior de JavaScript. Fue desarrollado por Microsoft para poder escribir código más sencillo y con menos errores. Además, es el lenguaje de programación para poder desarrollar proyectos con Angular.

Ilustración 34: Logo TypeScript

4.4 Metodología de trabajo

Como ya se ha mencionado anteriormente, para el desarrollo del proyecto nos hemos decantado por la utilización del modelo Kanban, más concretamente el tablero Kanban. Para aplicar correctamente dicha metodología, se pueden definir algunas fases como podemos ver en la ilustración 35, las cuales tienen que verse reflejadas en la planificación inicial.

Ilustración 35: Visualización del flujo de trabajo

5. DEFINICIÓN

El primer paso para aplicar la metodología Kanban es definir aquellos objetivos que que tiene que alcanzar la aplicación. Por ello, lo primero que hay que hacer es definir en la lista Backlog del tablero cuáles son las tareas a realizar para cumplir con los requisitos del proyecto.

Las tareas a realizar se van a definir mediante historias de usuario, que tendrán asociado un valor numérico que medirá la complejidad que supone realizar dicha tarea y el valor de negocio asociado.

Las historias de usuario [31] son una descripción breve e informal de las tareas que hay que llevar a cabo para alcanzar dicha historia de usuario.

Las historias de usuario suelen seguir un patrón:

- Un identificador para distinguir las diferentes historias.
- Una descripción breve y concisa de lo que se tiene que hacer, la cual sigue la siguiente estructura:
    - **Como** (usuario que la va a realizar)
    - **Quiero** (objetivo a conseguir)
    - **Para** (el motivo para conseguir el objetivo)
- Una estimación del valor de negocio y del esfuerzo que supone la realización de esta.
- Criterios de aceptación: son criterios que validan la historia.

Entrando un poco más en el apartado de estimación, podemos definir que la estimación de valor indica cómo de valiosa es esa historia para el usuario, y la estimación de esfuerzo indica cuánto le va a costar al desarrollador implementarla. Teniendo en cuenta esta explicación, hay que conseguir el mayor valor, pero el mínimo esfuerzo para el desarrollador.

Para asignar un valor a las historias de usuario, tenemos que:

- Ver que riesgos puede llevar a cabo implementar la historia de usuario.
- La pérdida que puede ocasionarnos su implementación.
- Qué beneficios podemos tener al implementarla.

Por lo tanto, para analizar esos puntos, nos vamos a ayudar de la técnica de MoSCoW. Esta técnica, va a ayudar al equipo a dar prioridad a las historias que más aporten. MoSCoW se puede descomponer como:

- M (MUST) → Tiene que estar obligatoriamente esta funcionalidad.
- S (SHOULD) → Debería estar esta funcionalidad.
- C (COULD)→ Podría estar esta funcionalidad.
- W (WON'T)→ No está en los planes tener esta funcionalidad.

Por otro lado, para estudiar cuánto tiempo o esfuerzo va a costar realizar cada historia de usuario, nos vamos a ayudar con el sistema de las tallas de camisetas. Este sistema va desde la talla XS hasta la XXL, y cada talla tendrá asignada unos puntos de historia, que se van a detallar a continuación:

Talla de camisetas	Puntos de historia
XS	1
S	2
M	3
L	5
XL	8
XXL	13

Tabla 1: Tabla talla de camisetas

Como vemos en la tabla 1, cuanto más pequeña sea la camiseta, menos esfuerzo va a costar al desarrollador implementarla y cuanto mayor sea la talla, más esfuerzo va a costar realizarla.

Por otro lado, los criterios de aceptación, como antes hemos mencionado, son como una serie de pautas que se tienen que seguir para que la historia de usuario sea realizada correctamente.

5.1 Historias de usuario

En este sub-apartado, vamos a definir cuáles son las historias de usuario a realizar para conseguir los objetivos finales del proyecto. Como podemos ver en la siguiente tabla, aparece el identificador de la tarea junto con el título descriptivo y la estimación del esfuerzo y el valor de dicha tarea.

La tabla que vamos a ver a continuación es simplemente descriptiva. En el capítulo 7 se entrará más en detalle con cada historia de usuario.

Identificador	Título	Esfuerzo	Valor
01	Buscar tablero mediante identificador	L	M
02	Mostrar los miembros que componen el tablero	L	S
03	Buscar acciones	XXL	M
04	Mostrar el porcentaje de actividad miembro	M	S
05	Mostrar las acciones agrupadas por tipo	L	M
06	Mostrar las acciones y los cambios por los que ha pasado una tarjeta	L	S
07	Mostrar las acciones y los cambios por los que ha pasado una lista	L	S
08	Mostrar la fecha de la actividad realizada	M	C
09	Mostrar el número de acciones realizadas por tipo	S	M
10	Mostrar el número de acciones realizadas por miembro	S	S
11	Iniciar sesión mediante la cuenta de Trello	XXL	M
12	Mostrar las acciones filtradas por tarjetas	L	S

13	Mostrar las acciones filtradas por listas	L	S
14	Mostrar numéricamente los cambios realizados en cada tarjeta	S	M
15	Mostrar numéricamente los cambios realizados en cada lista	S	M
16	Mostrar el porcentaje de actividad realizado en cada tarjeta	M	S
17	Mostrar las tarjetas de un tablero	M	S
18	Mostrar el porcentaje de actividad realizado en cada lista	M	S
19	Mostrar las listas de un tablero	L	S
20	Mostrar las acciones de un tablero	L	S
21	Visualizar en un gráfico las personas que forman el tablero	XL	C
22	Visualizar en un gráfico el tipo de acciones	XL	C
23	Visualizar en un gráfico los cambios de una tarjeta	XL	C
24	Mostrar flujo de movimiento de las	XL	C

	tarjetas entre las listas		
25	Iniciar sesión en la aplicación	XXL	C
26	Registrar usuario en la aplicación	XXL	C
27	Gestionar urls de los tableros	XXL	C
28	Agrupar urls de los tableros	XXL	C

Tabla 2: Historias de usuario

Una vez conocidas las historias de usuario, vamos a calcular cual es el esfuerzo total que nos va a llevar realizar todas las historias de usuario.

Para calcular el esfuerzo es necesario mirar la columna de esfuerzo y sumar los valores correspondientes a cada talla. Haciendo la sumatoria en total se realizan unos 180 puntos historia.

Para hacer una estimación, vamos a suponer que queremos realizar 45 puntos de historia, esto lo suponemos para tener una idea de las iteraciones en las que se va a desarrollar el proyecto. Para ello tendremos que realizar el siguiente cálculo:

Número de iteraciones = PH totales ÷ PH hacer →

4 = 180 ÷ 45

Por lo tanto, vamos a realizar un total de cuatro iteraciones, y en cada una vamos a tratar de hacer unos 45 puntos de historia.

El siguiente paso, será calcular cuántos días de trabajo va a suponer realizar dichas historias de usuario. Suponemos que para realizar los 45 PH vamos a tardar aproximadamente 10 días.

Como podemos ver en la tabla 3, vamos a calcular los días que tardará en ejecutarse cada talla, y los días totales.

Fórmula 1: (talla × días totales) ÷ PH totales

Fórmula 2: Resultado F1 × nº cada tipo

Talla	Fórmula 1	Fórmula 2
XS	(1×10)÷45 = 0,22	0,22 × 0 = 0
S	(2×10)÷45 = 0,44	0,44 × 4 = 1,76
M	(3×10)÷45 = 0,66	0,66 × 4 = 2,64
L	(5×10)÷45 = 1,11	1,11 × 10 = 11,1
XL	(8×10)÷45 = 1,77	1,77 × 4 = 7,08
XXL	(13×10)÷45= 2,88	2,88 × 6 = 17,28
Total días		40

Tabla 3: Resultados fórmulas estimación

5.2 Backlog

Siguiendo en el ámbito del diseño, todas las historias mencionadas anteriormente, irán incluidas en la lista del Backlog de nuestro tablero Kanban. Para seguir con la planificación establecida, se ha definido que dichas tareas sean realizadas según las siguientes iteraciones.

Iteración	Historias de usuario
1	HU(11), HU(1), HU(20), HU(17), HU(19), HU(5), HU(9), HU(2)
2	HU(10), HU(4), HU(8), HU(12), HU(14), HU(16), HU(6), HU(13), HU(15), HU(18), HU(7), HU(3)
3	HU(3),HU(21), HU(22), HU(23), HU(25), HU(24)
4	HU(24),HU(26), HU(27), HU(28)

Tabla 4: Reparto historias de usuario en cada iteración

5.3 Planificación

En este apartado se va a estimar cuánto tiempo va a llevar a cabo la realización del proyecto y además cuál va a ser el coste que nos va implicar el desarrollo.

5.3.1 Estimación de tiempo

Vamos a realizar una estimación del tiempo que nos va a llevar el desarrollo del proyecto, en principio va a ser una estimación a priori ya que no será la definitiva. Para ello vamos a suponer que solo contamos los días lectivos, sin incluir fines de semana y se va a trabajar 6 horas diarias.

Tareas Previas	Duración		Fecha	
	Días	**Horas**	**Inicio**	**Fin**
Objetivos y metodología	2	12	10/01/2022	11/01/2022
Coste del proyecto	3	18	12/01/2022	14/01/2022
Herramienta Trello	10	60	17/01/2022	28/01/2022
Especificación de la API	9	54	31/01/2022	10/02/2022
Tecnologías utilizadas	3	18	11/02/2022	15/02/2022
Historias de usuario	2	12	16/02/2022	17/02/2022
Diseño de la interfaz	3	18	18/02/2022	22/02/2022
Diseño de los diagramas de secuencia	5	30	23/02/2022	01/03/2022
Total	37	222		

Tabla 5: Tabla duración tareas previas

Tarea	Duración		Fecha	
	Días	**Horas**	**Inicio**	**Fin**
Implementación	40	240	02/03/2022	26/04/2022

Tabla 6: Tabla duración de la implementación

Tareas Posteriores	Duración		Fecha	
	Días	**Horas**	**Inicio**	**Fin**
Pruebas de verificación	6	36	27/04/2022	04/05/2022
Anexos	5	30	05/05/2022	11/05/2022
Total	11	66		

Tabla 7: Tabla duración de las tareas posteriores

Tarea	Duración		Fecha	
	Días	**Horas**	**Inicio**	**Fin**
Documentación	116	696	10/01/2022	20/06/2022
Total	116	696		

Tabla 8: Tabla duración de la documentación

Estimación final de tiempo			
Días	Horas	Semanas	Meses
204	1.224	28	7

Tabla 9: Tabla final de la estimación de tiempo

En total, la realización del Trabajo Fin de Grado si nos fijamos en la tabla 9 se ha llevado a cabo en 204 días, empleando 1.224 horas de trabajo, lo que equivale a 7 meses de trabajo (28 semanas).

5.3.2 Diagrama de Gantt

Una vez realizada la estimación del tiempo que nos va a llevar el desarrollo del proyecto, vamos a utilizar un diagrama de Gantt para ver visualmente como sería la planificación. Como vemos en la ilustración 36, en la parte izquierda encontramos las tareas a realizar junto con su fecha de inicio y de finalización. En la parte derecha podemos encontrar un diagrama de barras que refleja el trabajo realizado.

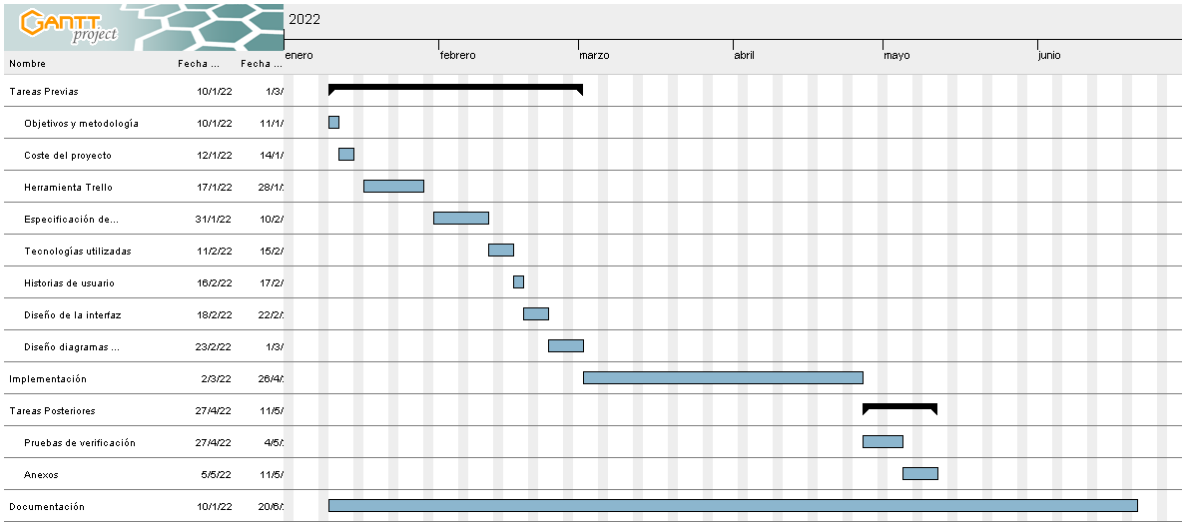

Ilustración 36: Diagrama de Gantt

5.3.3 Estimación de costes

Una vez calculado el tiempo que nos va a llevar el desarrollo del Trabajo de Fin de Grado, vamos a proceder a calcular el coste suponiendo que tardaremos 28 semanas, y teniendo en cuenta que el equipo para el desarrollo está formado por un solo miembro.

Según hemos consultado [32], el salario de un analista programador ronda los 29.600€ brutos/año, lo que significa 1.600€ netos/mes, hacemos una aproximación de 21 días que tiene el mes, y nos sale que cobra 71€/día, por lo que la hora cobra 12,69€/hora.

Por otro lado, para calcular el coste que nos va a suponer el hardware suponemos que tiene de vida útil 4 años.

En la tabla 10 podemos ver una clasificación de los gastos que nos va a suponer el desarrollo del proyecto.

Recurso		Cálculo	Coste
Gasto personal	Analista programador	12€/ hora x 1.224 horas	14.688€
Gasto hardware	Asus VivoBook S14 S435EA-KC035T-Portátil 14” (Intel Core i7-1165G7, 16GB RAM, 1TB SSD)	898€	130,95€
	TECKNET Ratón Inalámbrico Silencioso, Ratón Inalámbrico 2.4G	17,99€	2,62€
Gasto software	Spring Tools 4 for Eclipse	0,00€ x 7	0,00€
	Visual Studio Code 1.63	0,00€ x 7	0,00€
	Postman 8.12.4	0,00€ x 7	0,00€
	Giltab	0,00€ x 7	0,00€
	Visual Paradigm 16.3	Licencia estudiante	0,00€

	Microsoft Office 2016 Professional Plus	16,99€	16,99€
Gasto Inmaterial	Internet Fibra 600 Mb Jazztel	29,95€/mes	209,65€
Total			**15.048,21€**

Tabla 10: Tabla de costes del proyecto

6. DISEÑO INICIAL

En esta fase, vamos a definir cuál va a ser el diagrama de clases y cuál sería la idea inicial de la aplicación representada mediante wireframes.

6.1 Diagrama de clases

Como podemos ver en la ilustración 37, vamos a ver cuáles son las clases que componen el proyecto junto con sus atributos y métodos correspondientes.

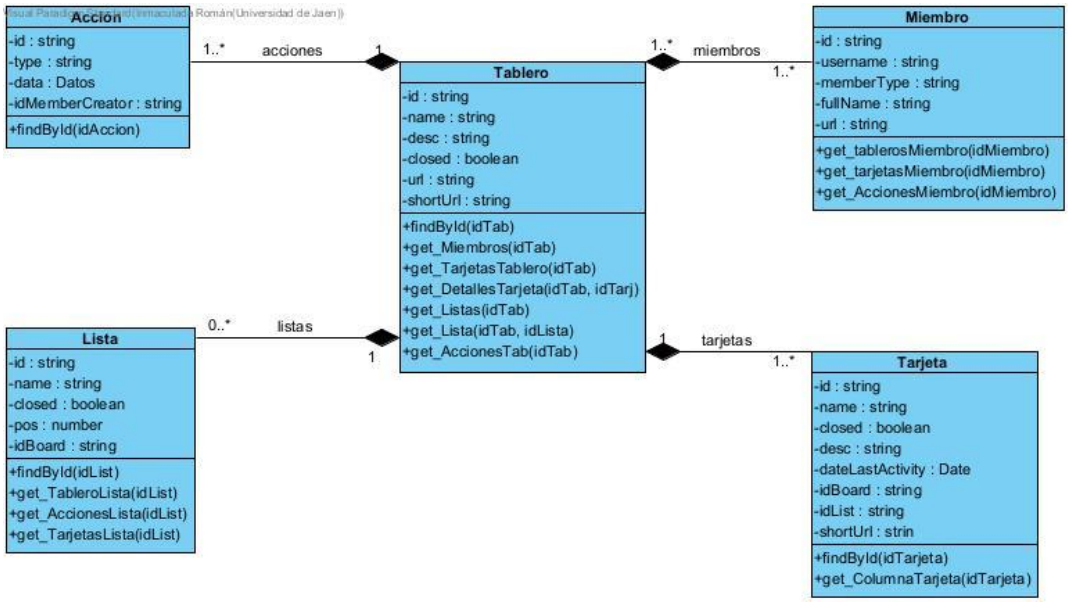

Ilustración 37: Diagrama de clases

6.2 Diseño de la interfaz

Antes de comenzar con el desarrollo del proyecto, se pensó en cómo podría ser la interfaz. Para ello se han realizado una serie de prototipos, los cuales serán con los que el usuario interaccione y, por lo tanto, deben ser lo más intuitivos posible. Comenzamos a mostrar los diseños de la interfaz para la aplicación web.

Página principal

La página principal, constará de un menú horizontal en la parte superior junto con un logo, el cual se encontrará en todas las páginas. En la ilustración 38 podemos ver que en la parte central de la página se encontrará una breve descripción de la aplicación.

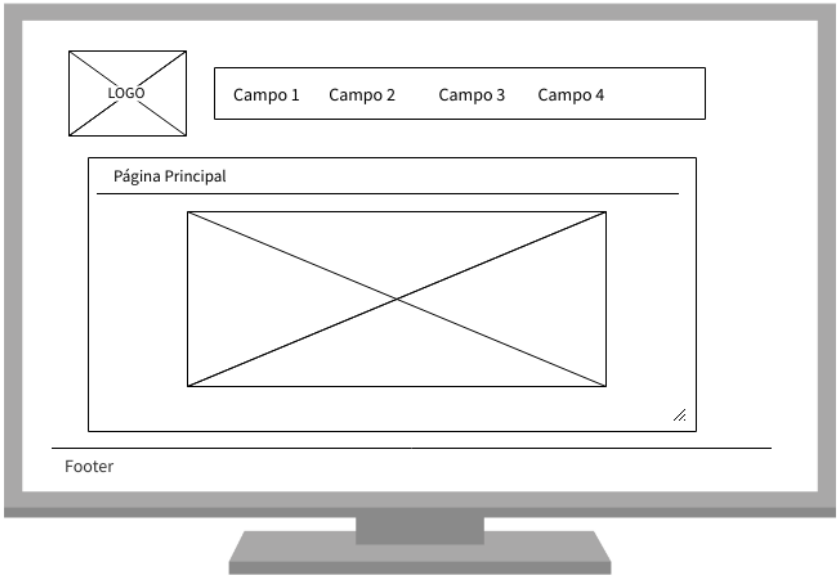

Ilustración 38: Prototipo de la página principal

Buscar un tablero mediante identificador

Para realizar la búsqueda un tablero, se necesitará un campo de búsqueda donde se pueda introducir el identificador, como podemos observar en la ilustración 39.

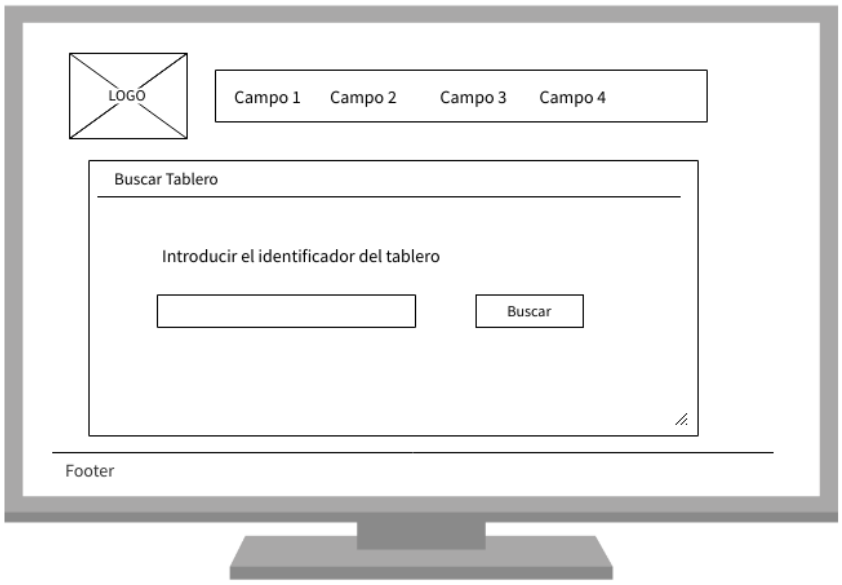

Ilustración 39: Prototipo de la página para buscar un tablero

Mostrar la información de un tablero

Una vez identificado el tablero, la aplicación redirigirá a otra página (ilustración 40) donde se mostrará la información correspondiente a ese tablero. También aparecerán unos botones para ver la información de otros aspectos del tablero.

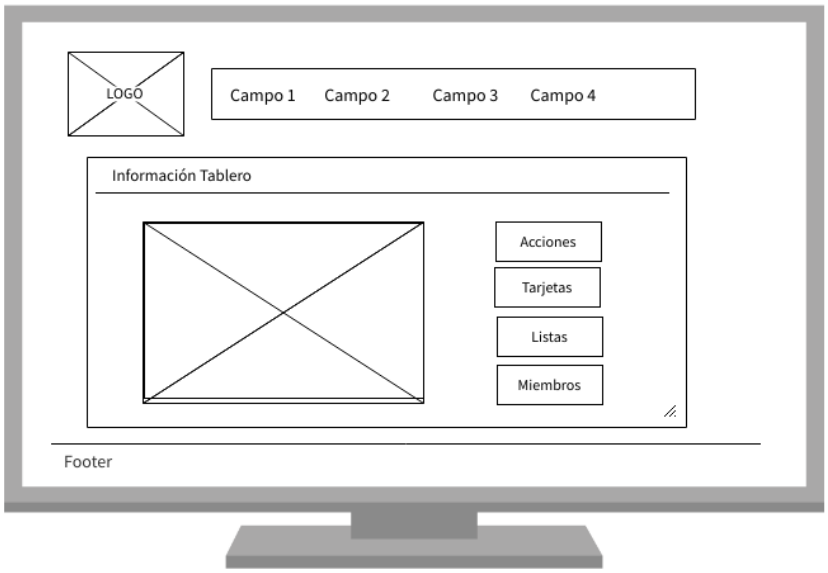

Ilustración 40: Prototipo de la página que muestra la información del tablero

Mostrar las tarjetas de un tablero

El usuario puede ver un listado de todas las tarjetas que se han creado en el tablero. Para ello ver ilustración 41.

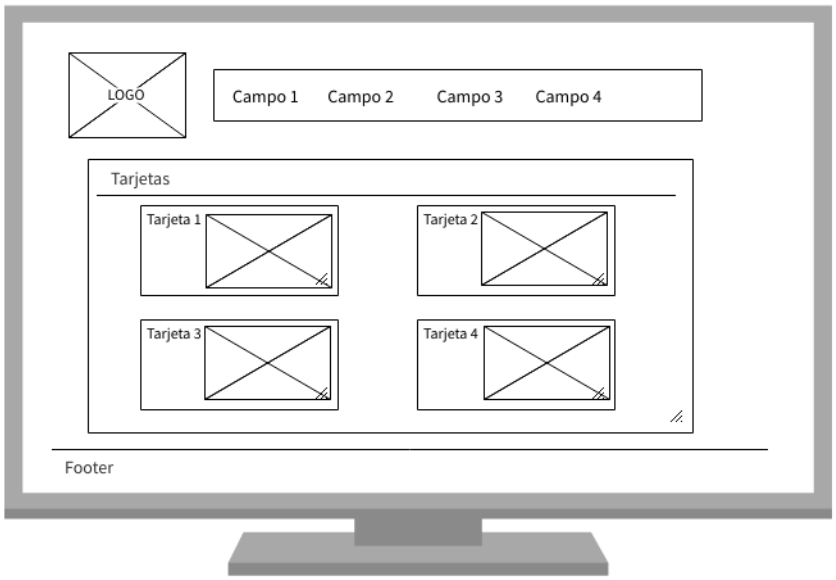

Ilustración 41: Prototipo para ver un listado de tarjetas

Mostrar información detallada de una tarjeta

Cuando el usuario esté visualizando el listado de las tarjetas, podrá pinchar sobre una tarjeta en concreto y ver toda su información (ilustración 42).

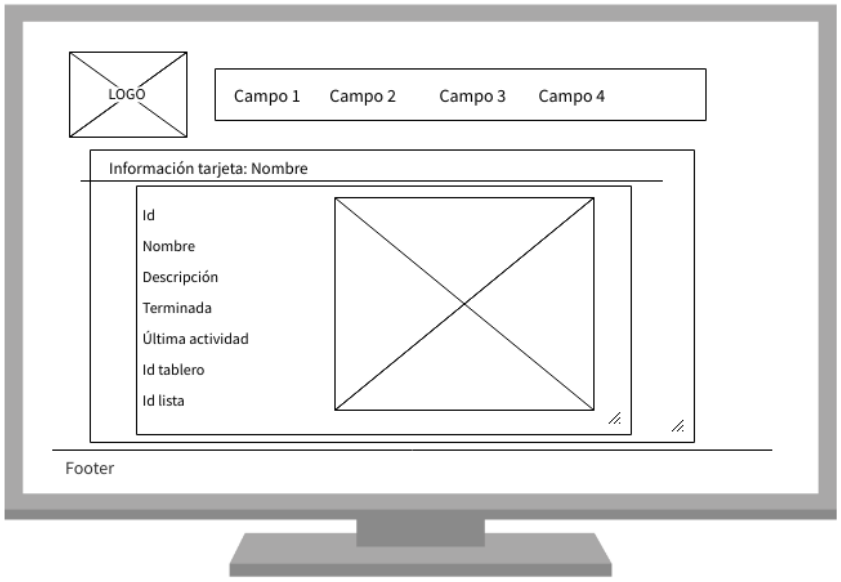

Ilustración 42: Prototipo para ver la información detallada de una tarjeta

Mostrar las listas de un tablero

Al igual que las tarjetas, el usuario puede ver un listado de todas las listas que se han creado en el tablero. Ver la ilustración 43.

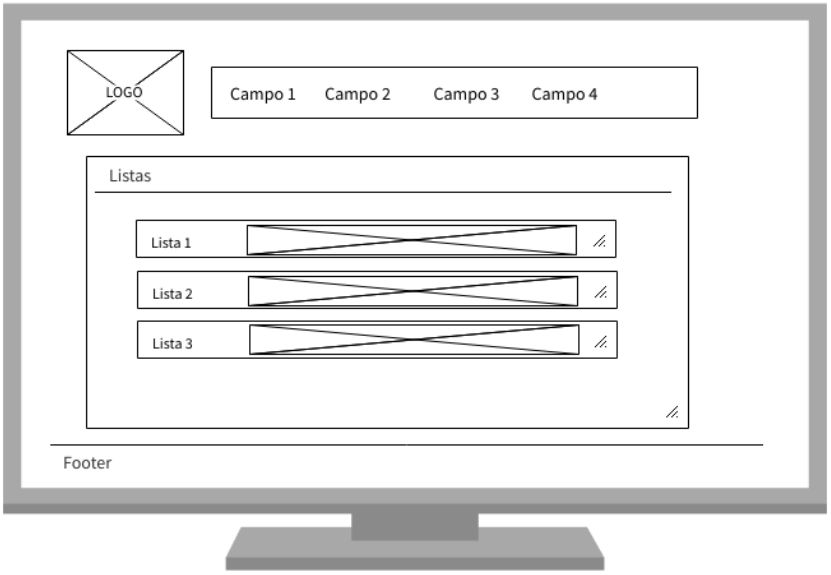

Ilustración 43: Prototipo para ver un listado de las listas

Mostrar información detallada de una lista

Si el usuario lo desea puede ver la información más detallada sobre una lista, como podemos ver en la ilustración 44.

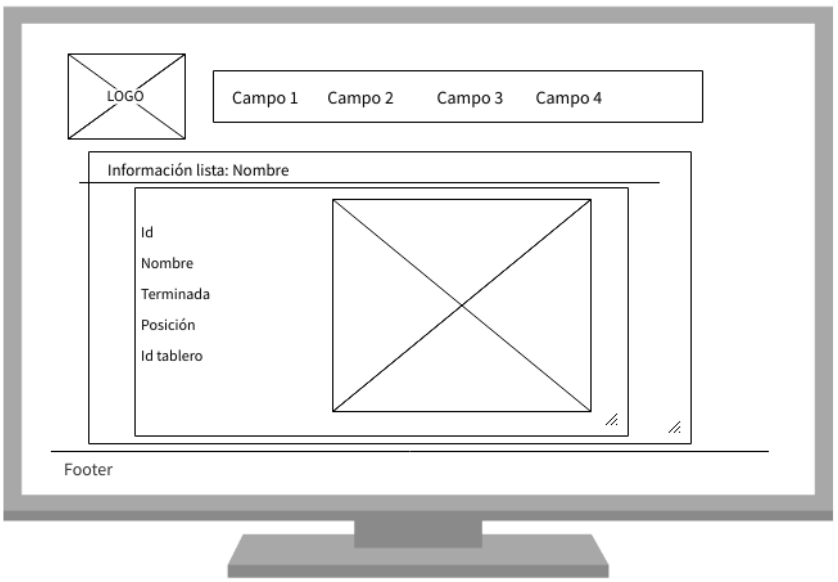

Ilustración 44: Prototipo para ver la información de una lista

Mostrar los miembros de un tablero

Si el usuario lo desea, podrá ver en una tabla cuáles son los miembros que forman parte del tablero, para ello ver ilustración 45.

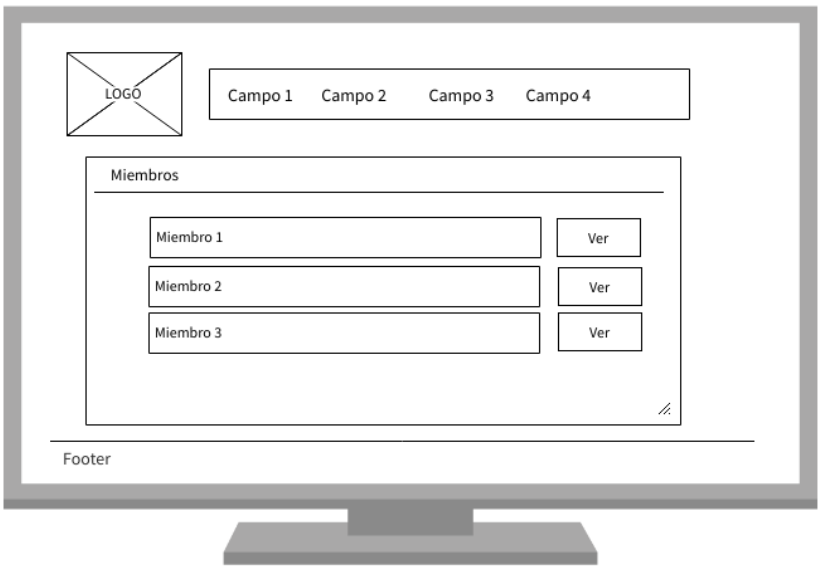

Ilustración 45: Prototipo para ver los miembros de un tablero

Mostrar las acciones de un tablero

Al igual que los miembros, si el usuario lo desea puede obtener información acerca de cuáles son todas las acciones que se han realizado sobre el tablero. Esta información se puede ver en la ilustración 46.

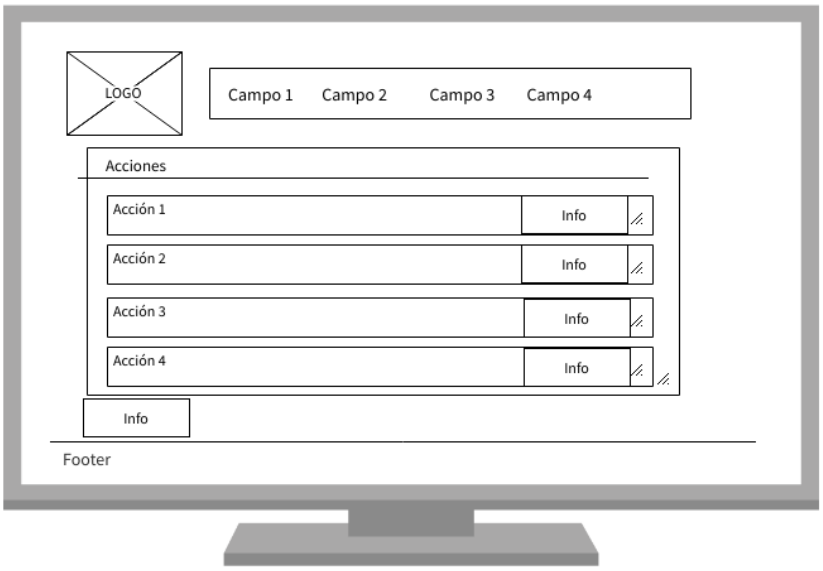

Ilustración 46: Prototipo para ver las acciones realizadas

Mostrar información detallada de una acción

Si el usuario lo desea podrá ver en detalle una acción, con sus correspondientes campos, como podemos ver en la ilustración 47.

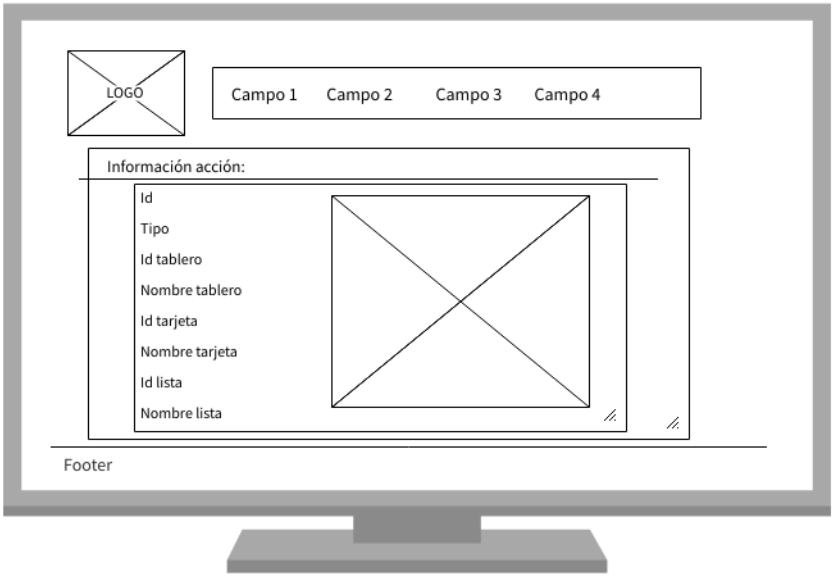

Ilustración 47: Prototipo para ver los detalles de una acción

Mostrar las acciones agrupadas por tipo y la actividad de los miembros

En esta página (ilustración 48) podemos encontrar un listado de las acciones agrupadas por tipo, junto con el número de veces que se han realizado cada una en el tablero. También podemos encontrar los miembros que componen el tablero, junto con el número y porcentaje de acciones que ha realizado cada uno.

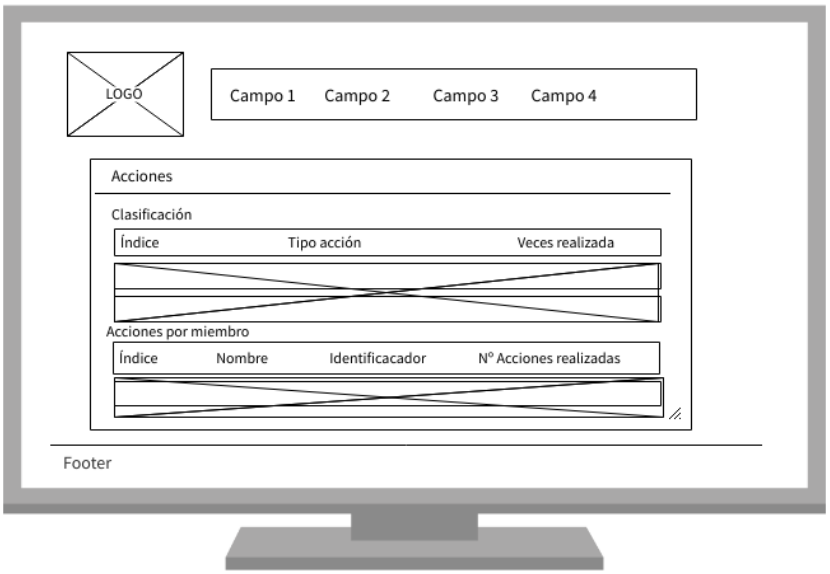

Ilustración 48: Prototipo de la página para ver las acciones del tablero

Desglosar la actividad de un miembro

Existirá una página donde se pueda ver de forma desglosada la actividad que ha realizado cada miembro, de forma que encontremos las acciones que ha realizado cada uno, con el número y porcentaje de veces que ha realizado cada tipo de actividad. En la parte inferior, se podrá encontrar un listado de las acciones a las que se refiere, para ello ver ilustración 49.

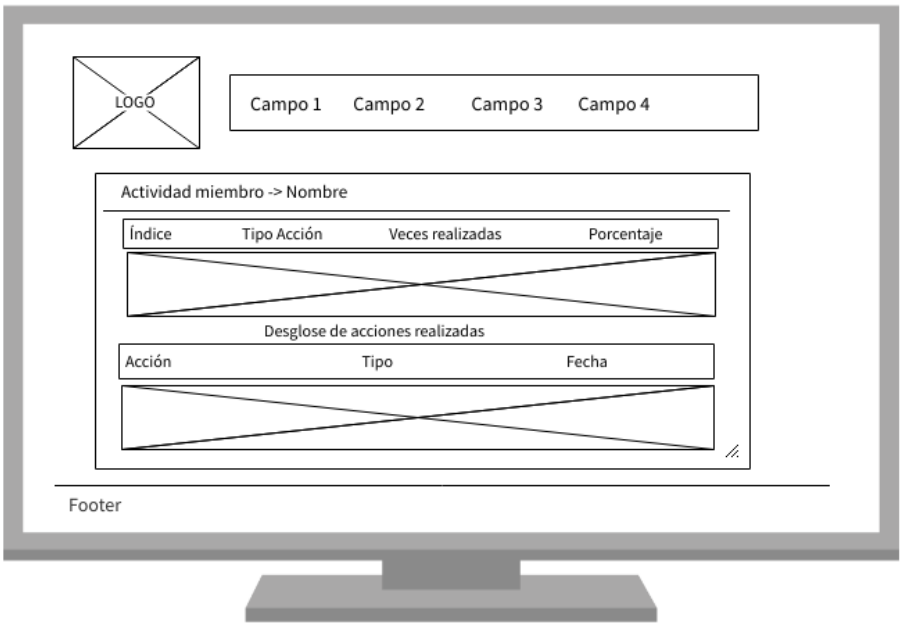

Ilustración 49: Prototipo para desglosar la actividad de cada miembro

Buscar las acciones de un tablero

Podemos encontrar una página para realizar la búsqueda de acciones, las cuales podemos filtrar por tipo o por miembro, y nos mostrará las acciones correspondientes. Esto lo podemos ver en la ilustración 50.

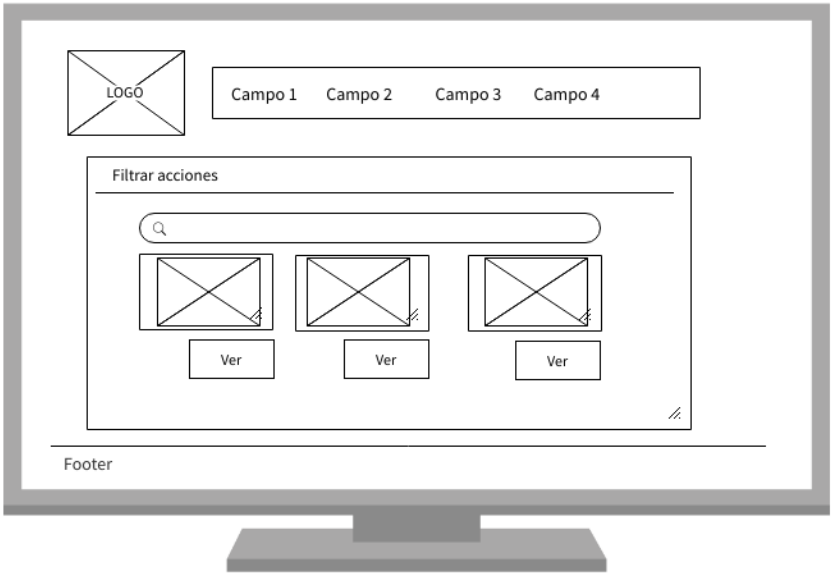

Ilustración 50: Prototipo para buscar acciones por tipo o miembro

Además de esta página, se ha pensado en el diseño de otra página referente a la búsqueda de acciones comprendidas entre dos fechas, para ello el diseño se puede apreciar en la ilustración 51.

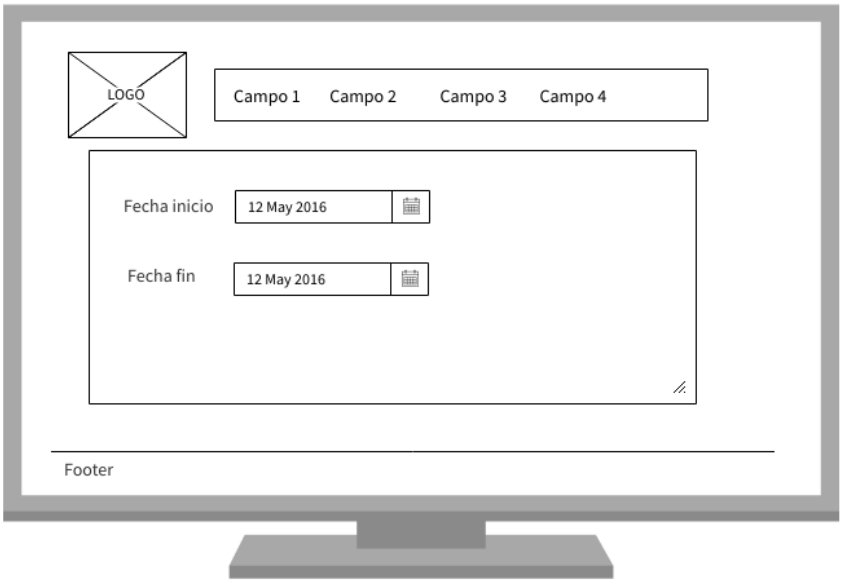

Ilustración 51: Prototipo para buscar acciones por fechas

Iniciar Sesión

Si el usuario que accede a la aplicación dispone de una cuenta de Trello, puede iniciar sesión en esta página, introduciendo su correo y su usuario (ilustración 52). Automáticamente la interfaz redirige al usuario a los tableros de los cuales forma parte.

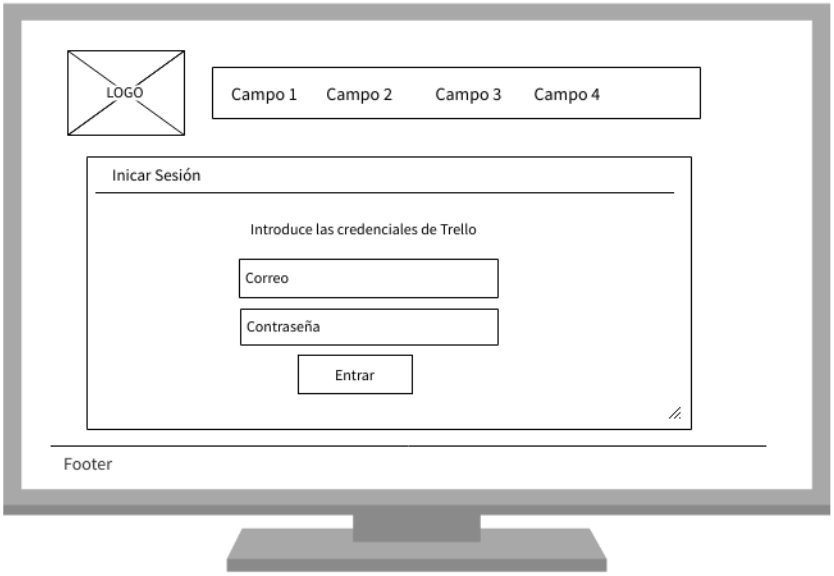

Ilustración 52: Prototipo de la página para iniciar sesión con Trello

Mostrar las acciones filtradas por tarjetas

Una vez obtenidas las acciones realizadas, se pueden obtener aquellas que son de tipo tarjeta. Para ello se mostrará un listado de las tarjetas junto con las acciones y los cambios que se han realizado en cada una. También aparecerá la actividad que ha realizado cada miembro, para ello observar ilustración 53.

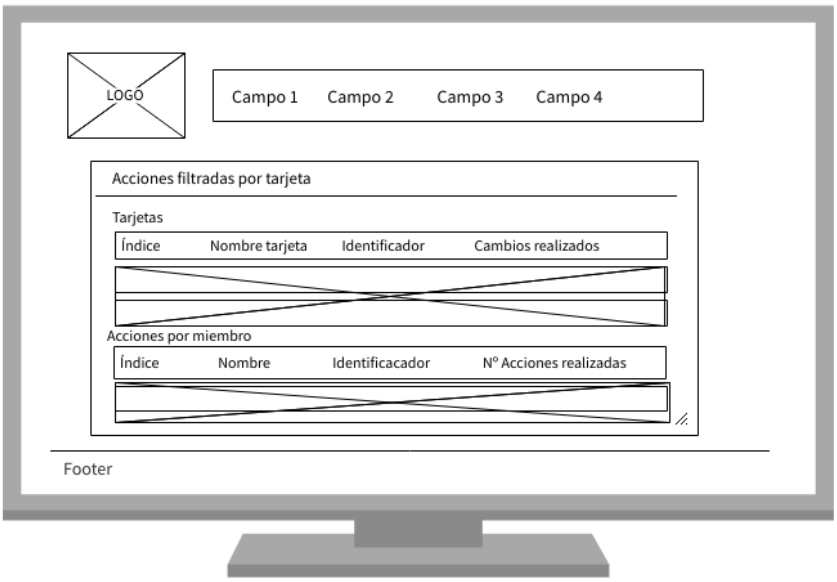

Ilustración 53: Prototipo para listar las acciones por tarjetas

Mostrar las acciones y los cambios realizados en una tarjeta

Como podemos observar en la ilustración 54, en esta página se puede ver cuáles son las acciones que se han realizado sobre una tarjeta y además podemos ver los cambios que se han ido realizando.

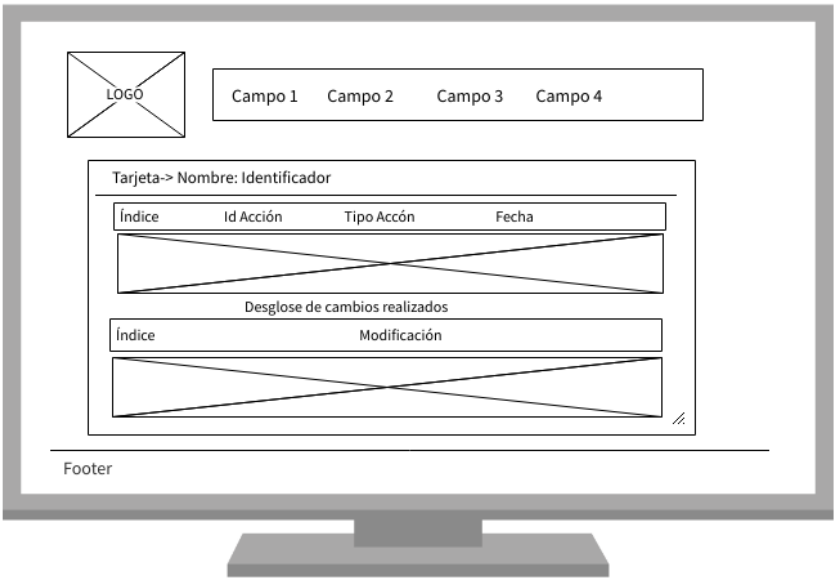

Ilustración 54: Prototipo para listar los cambios que ha sufrido una tarjeta

Mostrar las acciones filtradas por listas

Igual que las tarjetas, pero se mostrará un listado con las acciones que son de tipo listas junto con las acciones y los cambios que se han realizado en cada una. Y como antes, se muestra la actividad que ha realizado cada miembro, para ello ver la ilustración 55.

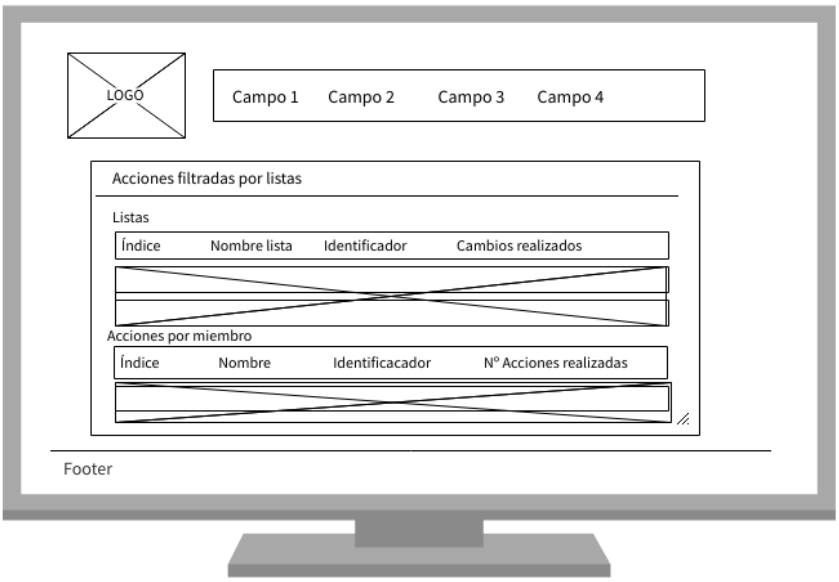

Ilustración 55: Prototipo para listar las acciones por listas

Mostrar las acciones y los cambios realizados en una lista

En esta página (ilustración 56) se puede ver en detalle las acciones que se han realizado en una lista y se puede desglosar los cambios por los que ha ido pasando la lista.

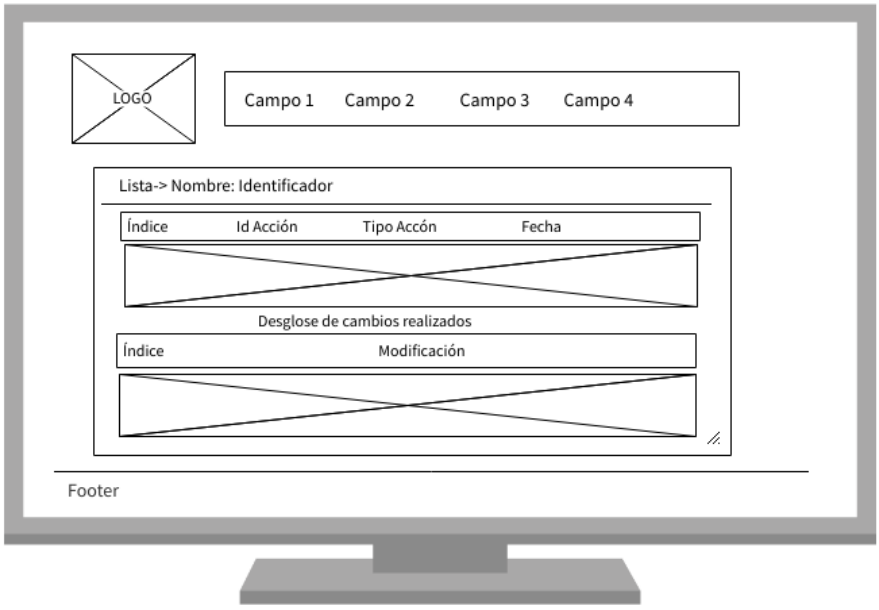

Ilustración 56: Prototipo para ver los cambios que ha sufrido una lista

7. DESARROLLO

Para comenzar, vamos a detallar más concretamente qué se ha ido realizando en cada iteración. Para llevar un mejor control se ha dividido en varias iteraciones, y en cada una se detallan cuáles han sido las historias de usuario llevadas a cabo, junto con los diagramas de comunicación correspondientes, que nos indican la comunicación que ha habido entre las distintas partes. También se ha detallado la implementación y por ultimo las pruebas realizadas sobre cada historia de usuario.

7.1 Primera iteración

7.1.1 Historias de usuario

El primer paso para llevar a cabo en la primera iteración consiste en detallar cuáles son las historias de usuario que se van a llevar a cabo. Para ello, cada historia de usuario va a ir representada en una tabla con los siguientes campos:

- Identificador.
- Título.
- Descripción.
- Estimación de valor y esfuerzo.
- Criterios de aceptación.

Identificador	11
Título	Iniciar Sesión mediante la cuenta de Trello
Descripción	Como auditor de un tablero Trello quiero poder acceder con mis credenciales de Trello para obtener toda la información almacenada en Trello.
Valor	M
Esfuerzo	XXL
Criterios de aceptación	-Disponer de una cuenta en la página de Trello. -Rellenarlos campos de la página de inicio de sesión.

Tabla 11: HU Inicio de sesión mediante la cuenta de Trello

Identificador	01
Título	Buscar tablero mediante identificador
Descripción	Como auditor de un tablero Trello quiero poder indicar el identificador de un tablero Trello para poder obtener información de las acciones que se llevan a cabo en él.
Valor	M
Esfuerzo	L
Criterios de aceptación	-Obtener un identificador válido de un tablero. -Introducir el id en un formulario de la aplicación.

Tabla 12: HU Buscar un tablero mediante identificador

Identificador	20
Título	Mostrar las acciones de un tablero
Descripción	Como auditor de un tablero Trello quiero ver cada acción para ver su información correspondiente.
Valor	S
Esfuerzo	L
Criterios de aceptación	-Introducido el id del tablero válido, pulsar el botón de acciones para ver las acciones realizadas.

Tabla 13: HU Mostrar las acciones de un tablero

Identificador	17
Título	Mostrar las tarjetas de un tablero
Descripción	Como auditor de un tablero Trello quiero ver todas las tarjetas que tiene un tablero para obtener información de ellas.
Valor	S
Esfuerzo	L
Criterios de aceptación	-Introducido el id del tablero, pulsar el botón tarjetas para ver todas las tarjetas creadas en él.

Tabla 14: HU Mostrar las tarjetas de un tablero

Identificador	19
Título	Mostrar las listas de un tablero
Descripción	Como auditor de un tablero Trello quiero poder obtener las listas de un tablero para ver su información y las tarjetas que tiene cada una.
Valor	S
Esfuerzo	L
Criterios de aceptación	-Introducido el id del tablero, pulsar el botón tarjetas para ver todas las listas creadas en él.

Tabla 15: HU Mostrar las listas de un tablero

Identificador	05
Título	Mostrar las acciones agrupadas por tipo
Descripción	Como auditor de un tablero Trello quiero saber qué tipos de acciones se han realizado en el tablero para agrupar las distintas actividades.
Valor	M
Esfuerzo	L
Criterios de aceptación	-Introducir id del tablero válido. -Obtener las acciones realizadas. -Agrupar las acciones por tipo.

Tabla 16: HU Mostrar las acciones agrupadas por tipo

Identificador	09
Título	Mostrar el número de acciones realizadas por tipo
Descripción	Como auditor de un tablero Trello quiero saber cuántas acciones de cada tipo se han realizado para cuantificar los distintos grupos.
Valor	M
Esfuerzo	S
Criterios de aceptación	-Introducir id del tablero válido. -Obtener las acciones realizadas. -Agrupar las acciones por tipo.

Tabla 17: HU Mostrar el número de acciones realizadas por tipo

Identificador	02
Título	Mostrar los miembros que componen el tablero
Descripción	Como auditor de un tablero Trello quiero visualizar numéricamente cuántas personas tienen acceso al tablero para saber el máximo número de personas que pueden editar el tablero.
Valor	S
Esfuerzo	L
Criterios de aceptación	-Introducir id del tablero válido. -Obtener miembros del tablero.

Tabla 18: HU Mostrar los miembros que componen el tablero

7.1.2 Diseño

Para el diseño de la aplicación se ha pensado usar los diagramas de secuencia, para explicar mejor la comunicación que existe entre las distintas partes que componen la aplicación.

Los diagramas de secuencia [33] son uno de los muchos diagramas existentes, los cuales modelan la comunicación entre el usuario y los distintos objetos de un sistema.

El uso de estos diagramas nos va a ser muy útil para entender la comunicación de las diferentes partes que componen el sistema.

Los diagramas correspondientes a las historias de usuario explicadas anteriormente son:

HU11- Iniciar sesión mediante la cuenta de Trello

Objetivo: Mostrar la comunicación que se produce para iniciar sesión en la aplicación mediante nuestra cuenta de Trello.

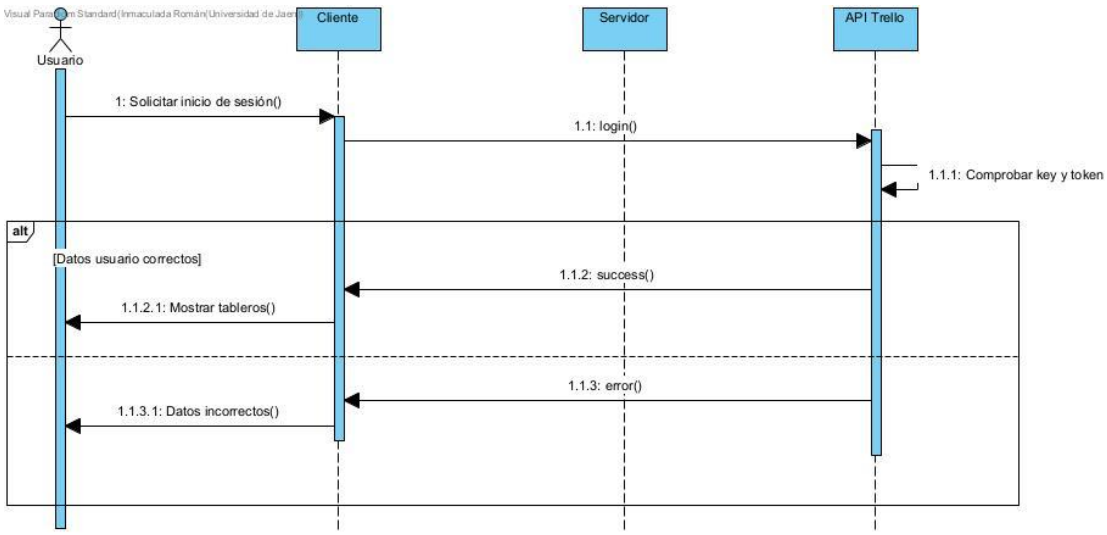

Ilustración 57: Diagrama de secuencia Inicio de Sesión

Para iniciar sesión es necesario:

I. El usuario solicita al cliente iniciar sesión con la cuenta de Trello.
II. El cliente realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/members/me/boards?key=${this.key}&token=${this.token}
III. La Api encapsula la información en formato JSON y se la devuelve al cliente.
IV. El cliente muestra al usuario los tableros a los que pertenece en formato HTML.

HU01- Buscar tablero mediante identificador

Objetivo: Al introducir el usuario un identificador de tablero, el sistema sea capaz de buscar y proporcionar la información correspondiente.

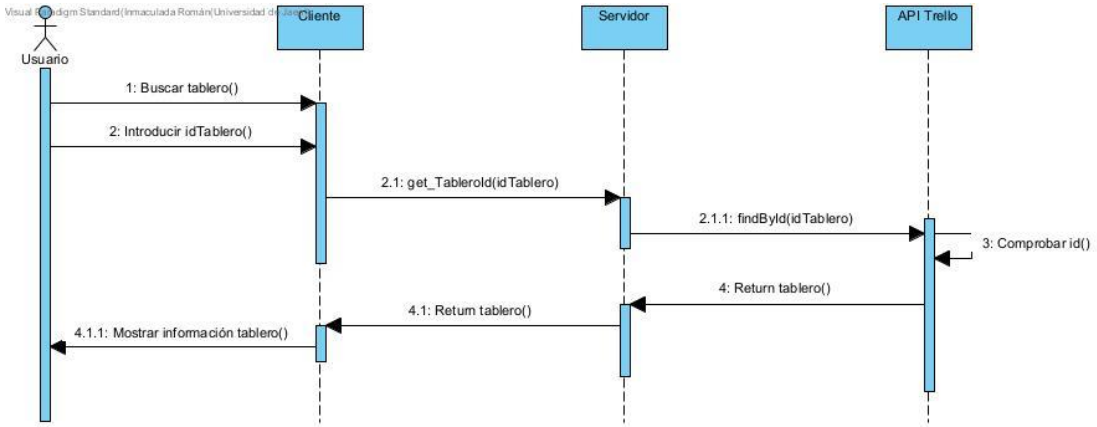

Ilustración 58: Diagrama de secuencia de la búsqueda de un tablero

Para buscar la información correspondiente a un tablero es necesario:

I. El usuario debe introducir el identificador del tablero a buscar.
II. El cliente solicita al servidor la información del tablero pasándole el identificador.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra lo datos al usuario en formato HTML.

HU20- Mostrar las acciones de un tablero

Objetivo: Obtenida la información del tablero, mostrar todas las acciones que se han realizado en él.

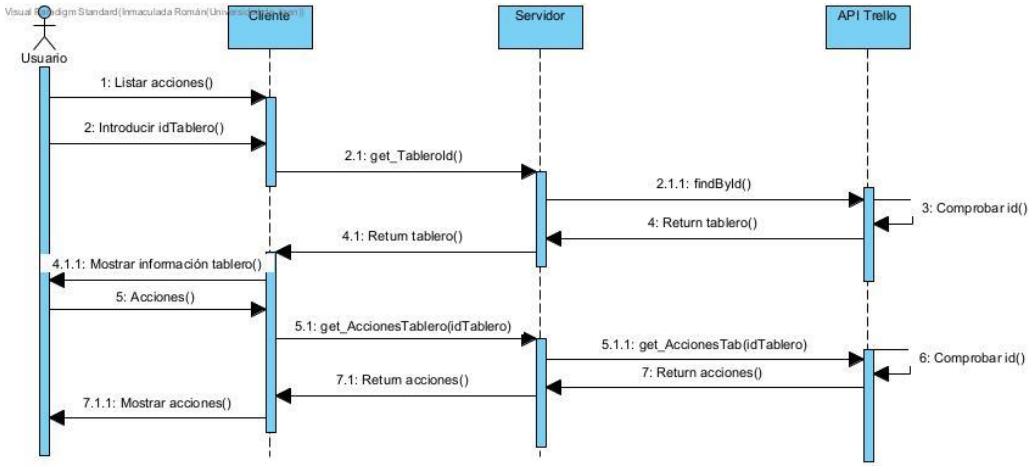

Ilustración 59: Diagrama de secuencia para mostrar las acciones

Para obtener las acciones realizadas en el tablero es necesario:

I. El usuario quiere visualizar todas las acciones que se han realizado en el tablero.
II. El cliente solicita al servidor información sobre las acciones realizadas indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra lo datos al usuario en formato HTML.

HU17- Mostrar las tarjetas de un tablero

Objetivo: Obtenida previamente la información del tablero, listar las tarjetas que se han creado en dicho tablero.

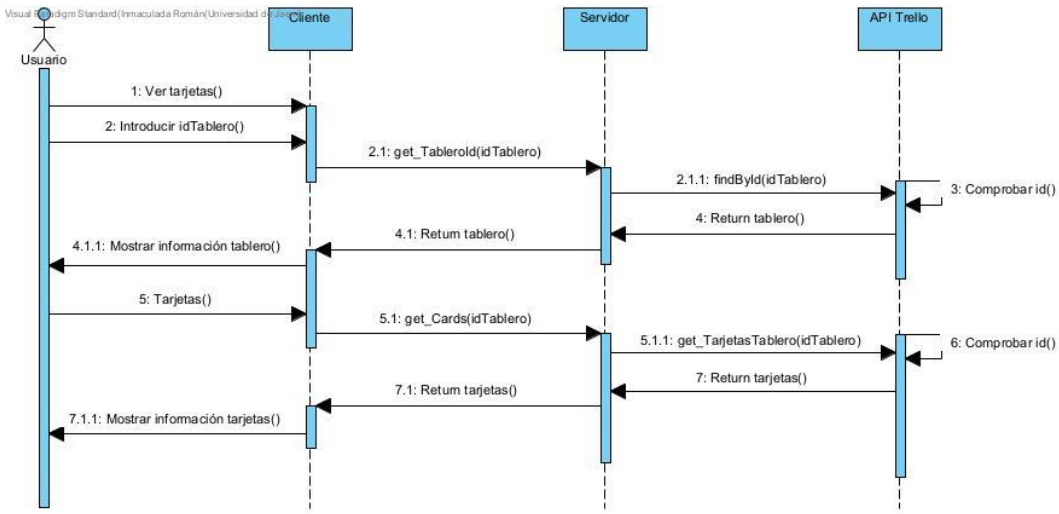

Ilustración 60: Diagrama de secuencia para mostrar las tarjetas

Para obtener las tarjetas del tablero es necesario:

I. El usuario quiere visualizar todas las tarjetas que se han creado en el tablero.
II. El cliente solicita al servidor información sobre las tarjetas realizadas indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/cards
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra lo datos al usuario en formato HTML.

HU19- Mostrar las listas de un tablero

Objetivo: Obtenida la información del tablero, listar las listas que se han creado en el tablero.

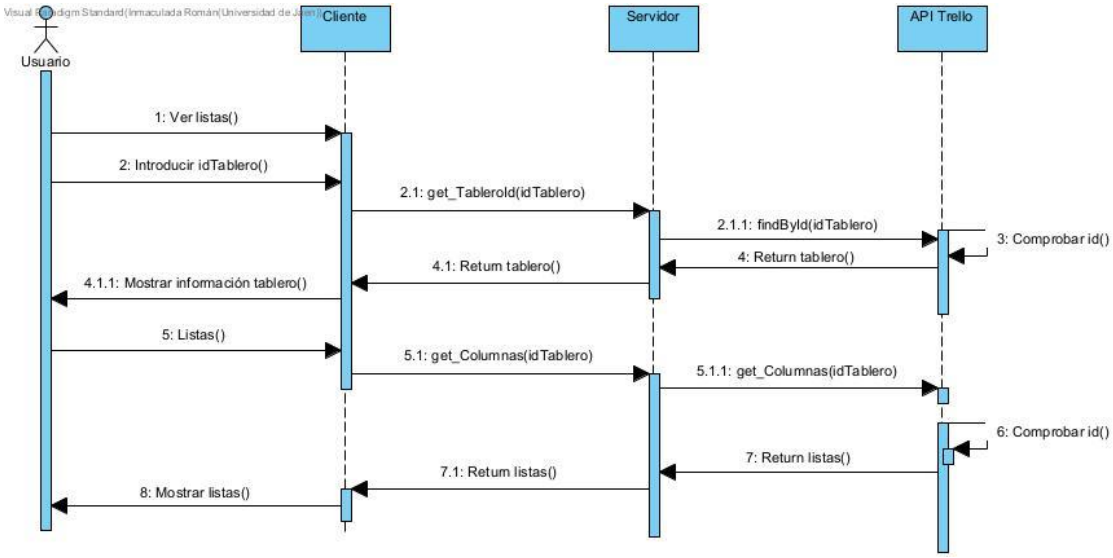

Ilustración 61: Diagrama de secuencia para mostrar las listas

Para obtener las listas del tablero es necesario:

I. El usuario quiere visualizar todas las listas creadas en el tablero.
II. El cliente solicita al servidor información sobre las listas creadas indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/lists
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra lo datos al usuario en formato HTML.

HU05- Mostrar las acciones agrupadas por tipo

Objetivo: De entre todas las acciones realizadas, realizar una agrupación por tipo y listarlas al usuario.

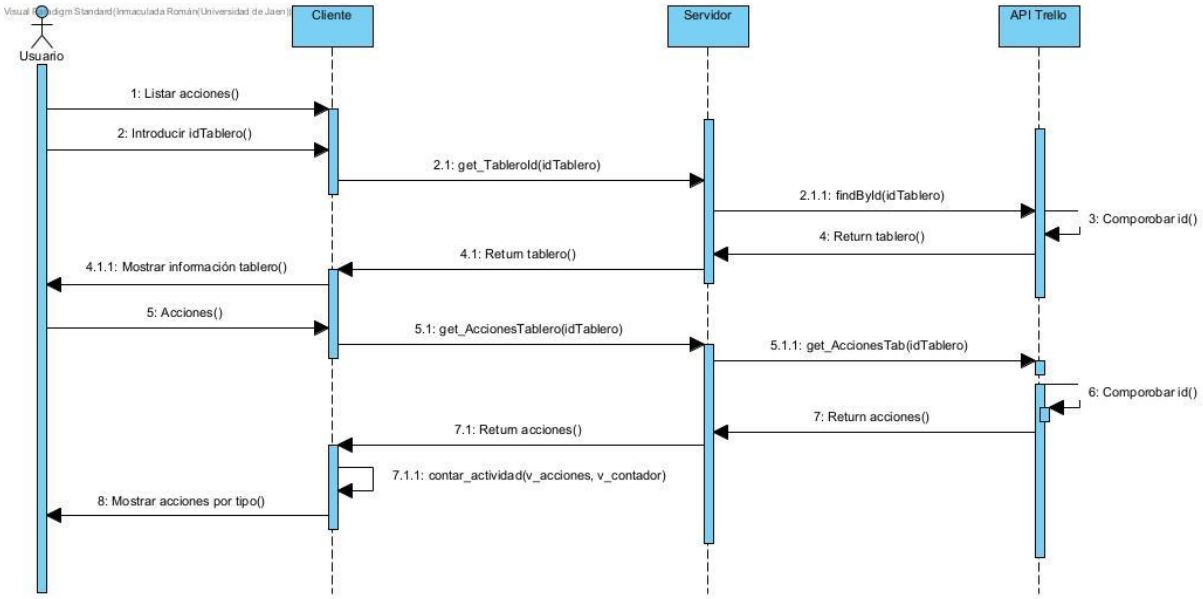

Ilustración 62: Diagrama de secuencia para agrupar las acciones por tipo

Para obtener las acciones agrupadas por tipo es necesario:

I. El usuario quiere obtener cuales son el tipo de acciones que se han realizado en el tablero.
II. El cliente solicita al servidor información sobre las acciones realizadas indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra lo datos al usuario en formato HTML, extrayendo las acciones clasificándolas por tipo.

HU09- Mostrar el número de acciones realizadas por tipo

Objetivo: Cuantificar las acciones obtenidas previamente por tipo, para saber el número que hay de cada tipo.

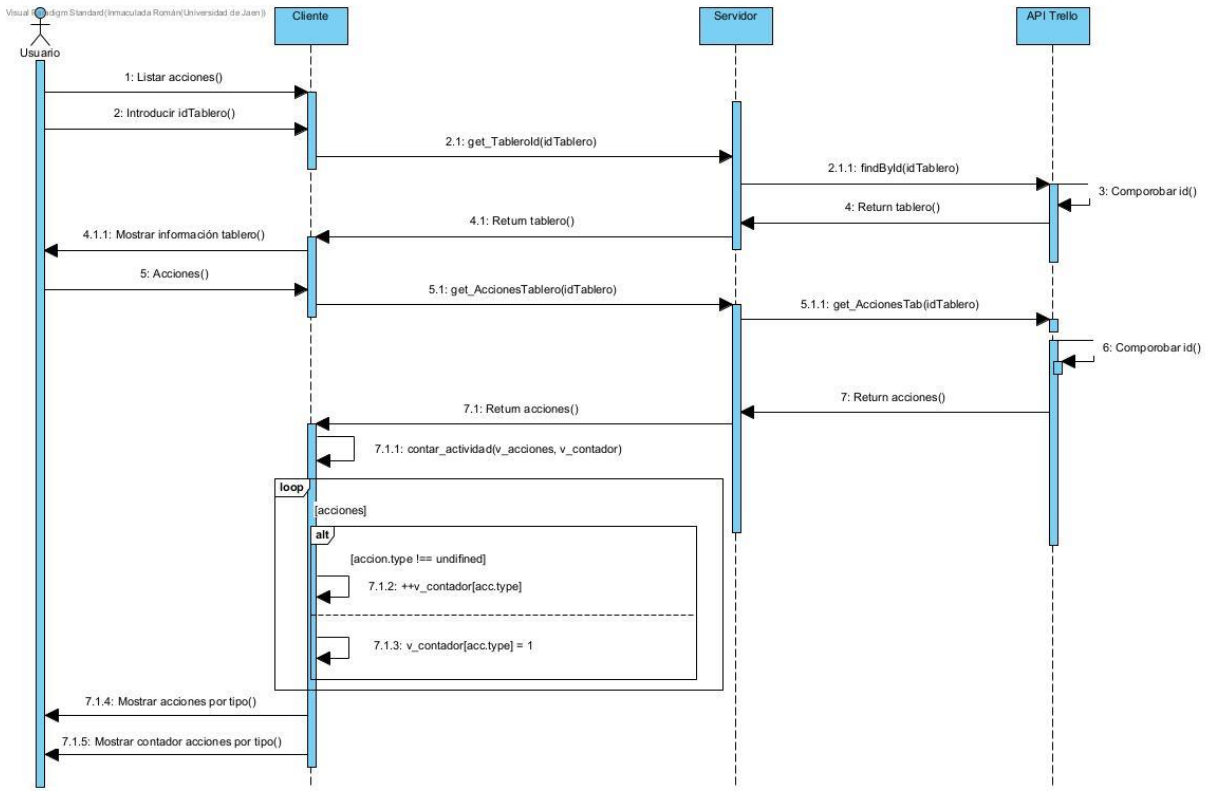

Ilustración 63: Diagrama de secuencia de las acciones realizadas por tipo

Para obtener el número de acciones agrupadas por tipo es necesario:

I. El usuario quiere obtener cuantificar cuantas acciones se han realizado según la clasificación por tipo.
II. El cliente solicita al servidor información sobre las acciones realizadas indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente realiza una clasificación de según el tipo de acciones realizadas y va sumando cuantas hay de cada tipo en formato HTML.

HU02- Mostrar los miembros que componen el tablero

Objetivo: Obtenida la información del tablero, obtener los miembros que componen el tablero.

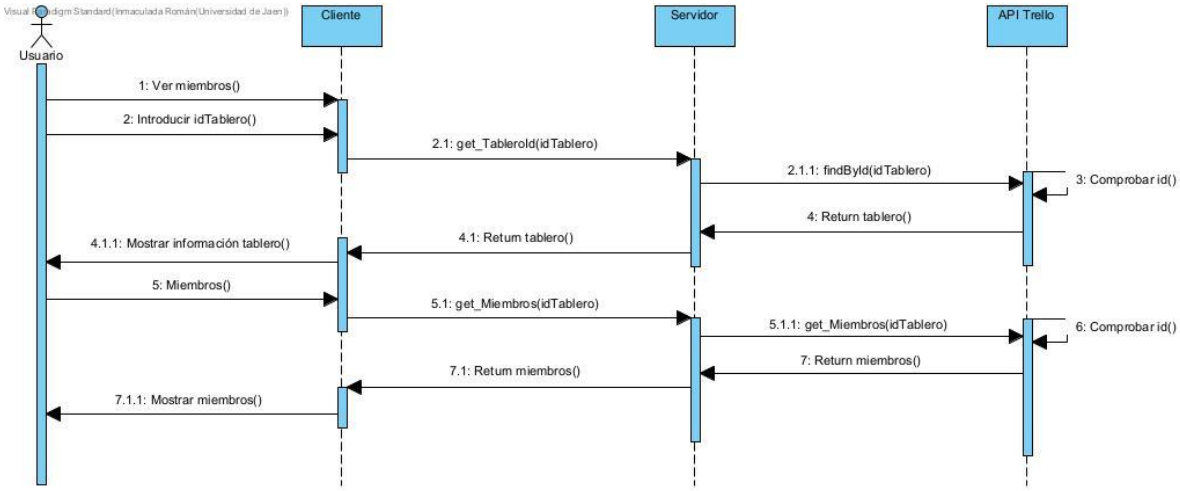

Ilustración 64: Diagrama de secuencia para listar los miembros del tablero

Para obtener los miembros que trabajan en el tablero es necesario:

I. El usuario quiere obtener un listado con los miembros que forman parte del tablero.
II. El cliente solicita al servidor información sobre los miembros indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/members
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra al usuario la información recibida en formato HTML.

7.1.3 Implementación

Una vez tenemos claro cuál es el diseño de la aplicación, vamos a comenzar con la implementación. Para ello, vamos a dividir la implantación en dos partes, empezando por el back-end y más tarde con el front-end.

Implementación del Servidor

Para comenzar con la implementación del servidor, es necesario la instalación Spring ToolSuite4 y la creación de un nuevo proyecto maven.

Cuando el proyecto haya sido creado, lo primero será modificar el archivo pom.xml [34]. Este archivo es un fichero de configuración para la creación del proyecto maven, al cual está formado por las librerías y dependencias necesarias para iniciar un proyecto Spring Boot.

En la ilustración 65, podemos ver el archivo pom.xml de nuestro proyecto. Aparte de contener las dependencias necesarias para la creación y configuración de un proyecto Spring Boot, contiene la dependencia WebFlux, la cual es necesaria para poder hacer uso de WebClient y así poder establecer la comunicación con la API de Trello.

spring-boot-WebClient/pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.2</version>
        <relativePath /> <!-- lookup parent from repository -->
    </parent>
    <groupId>es.ujaen.servidor.webClient</groupId>
    <artifactId>spring-boot-WebClient</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>spring-boot-WebClient</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-webflux</artifactId>
        </dependency>


        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.json</groupId>
            <artifactId>json</artifactId>
            <version>20211205</version>
        </dependency>
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.8.6</version>
            <type>jar</type>
        </dependency>

        <dependency>
            <groupId>io.projectreactor</groupId>
            <artifactId>reactor-test</artifactId>
            <scope>test</scope>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Ilustración 65: Fichero pom.xml

En la ilustración 66 podemos visualizar la estructura principal del proyecto, el cual se divide en 4 paquetes Java. A continuación, lo vamos a desglosar.

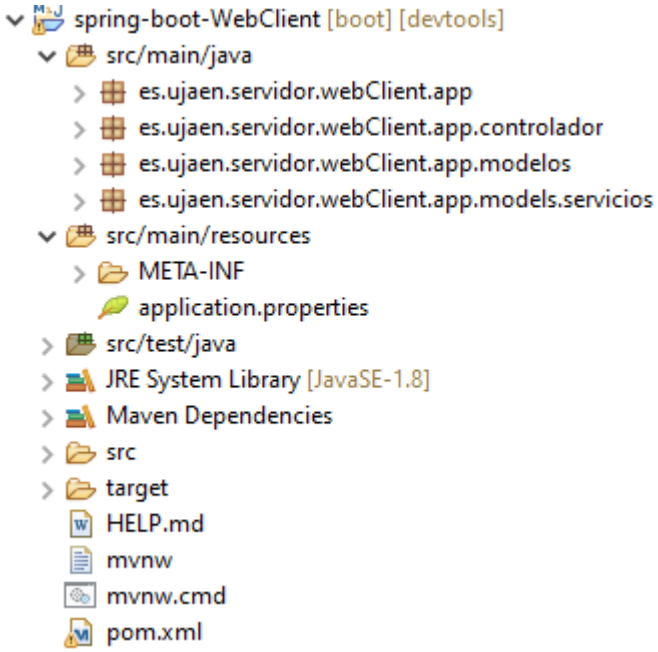

Ilustración 66: Estructura del servidor

Vamos a comenzar a detallar el paquete “app” (ilustración 67) el cual está formado por los siguientes ficheros java.

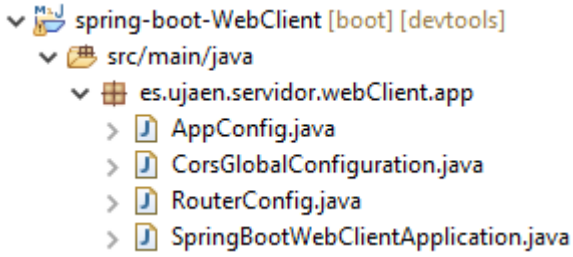

Ilustración 67: Ficheros paquete app del servidor

Dicho paquete contiene la clase SringBootWebClientApplication.java, la cual será la encargada de arrancar la aplicación.

También se encuentra la clase CorsGlobalConfiguration.java, que se encarga de la configuración de las CORS. Las CORS son un mecanismo de las solicitudes http, para permitir acceder a los recursos necesarios desde un punto origen diferente al servidor.

En la clase AppConfig.java, se define un WebClient, con la url a la que tiene que acceder para obtener la información correspondiente, se puede ver en la ilustración 68.

```java
@Configuration
public class AppConfig {

    @Bean
    public WebClient registrarWebClient() {
        return WebClient.create("https://api.trello.com/1");
    }

}
```

Ilustración 68: Definición del webClient

Y, por último, encontramos la clase RouterConfig.java. Esta es una clase de configuración en la cual se define un método que será el encargado de configurar las rutas para mapear.

En el paquete “app.controlador” (ver ilustración 69), encontramos el fichero controlador_Api.java. En esta clase se definen los controladores necesarios del API Rest y es la encargada de procesar las solicitudes http para mostrar la respuesta.

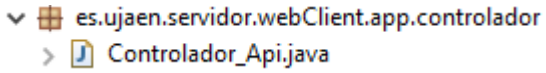

Ilustración 69: Ficheros paquete controlador del servidor

Por otro lado encontramos el paquete “app.modelos” (ilustración 70), en el cual se definen las clases que se utilizarán en la aplicación.

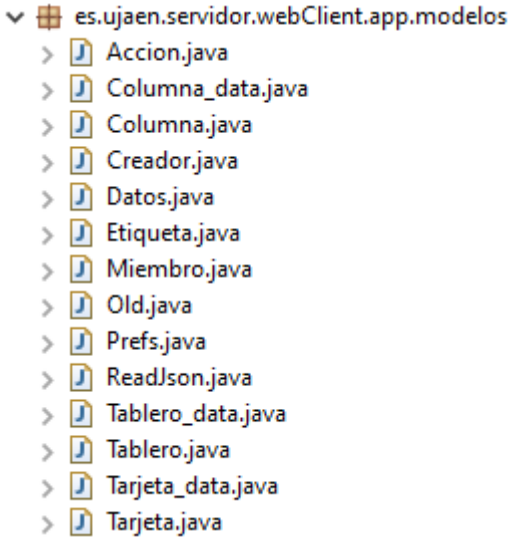

Ilustración 70: Ficheros del paquete modelos del servidor

También encontramos el paquete “app.modelos.servicios” (ver ilustración 71), en el cual vamos a definir las interfaces correspondientes a cada modelo con sus métodos necesarios. Además, se van a definir los servicios que componen cada interfaz.

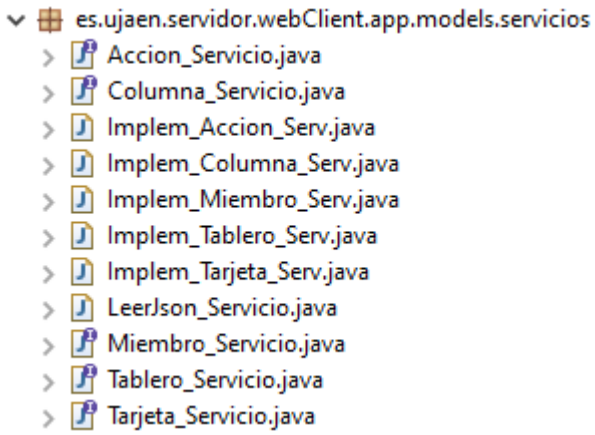

Ilustración 71: Ficheros del paquete servicios del servidor

Los servicios irán anotados con @Service, y en ellos se va a definir un WebClient. En estas clases se van a definir los métodos http necesarios para realizar las llamadas a la API de Trello.

Para realizar una solicitud a la API, cada método debe tener el siguiente formato:

```
public Mono<Tarjeta> get_DetallesTarjeta(String idTab, String idTarj) {
            Mono<Tarjeta> listaDetalle = client.get()
                        .uri("/boards/{idTab}/cards/{idTarj}", idTab,idTarj)
                        .accept(MediaType.APPLICATION_JSON)
                        .retrieve()
                        .bodyToMono(Tarjeta.class);
            return listaDetalle;

      }
```

Para entender el funcionamiento de la llamada anterior, simplemente queremos obtener información de una tarjeta en concreto, como el resultado a devolver es un único, el método será de tipo Mono. Con retrieve() indicamos a la solicitud, que el resultado devuelto tiene que seguir según lo definido.

Y por último, en la carpeta “resources”, más concretamente en el fichero “application.properties”, encontramos cual será el puerto del servidor, que en este caso es el puerto 8080.

Una vez explicada la estructura del servidor, vamos a proceder a explicar que consultas se han realizado, para obtener la información correspondiente de la API.

Título	Url	Método	Descripción
Obtener un tablero a partir de su identificador	/servidor/tableros/tablero/{idTab}	GET	Obtener información concreta de un tablero Trello
Obtener las acciones de un tablero	/servidor/tableros/tablero/{idTab} /acciones	GET	Obtener las acciones que se han realizado en el tablero
Obtener las tarjetas de un tablero	/servidor/tableros/tablero/{idTab} /tarjetas	GET	Obtener informacion de las tarjetas creadas en el tablero
Obtener información de una tarjeta	/servidor/tableros/tablero/{idTab} /tarjetas/{idTarjeta}	GET	Mostrar la información correspondiente a una tarjeta
Obtener las listas de un tablero	/servidor/tableros/tablero/{idTab} /listas	GET	Obtener información de todas las listas que componen el tablero
Obtener infomación de una lista	/servidor/tableros/tablero/{idTab} /listas/{idLista}	GET	Mostrar la información correspondiente a una lista
Obtener los miembros de un tablero	/servidor/tableros/tablero/{idTab} /miembros	GET	Obtener información sobre los miembros que pertenecen al tablero

Tabla 19: Solicitudes http del servidor

Con esto, la implementación del servidor quedaría terminada.

Implementación del cliente

Para el desarrollo de la aplicación cliente es necesario la instalación de Visual Studio Code, e instalar Angular Cli mediante comandos, los cuales han sido previamente explicados en el apartado 4.3. Una vez instalado y creado el proyecto, procedemos a explicar su estructura, la cual podemos ver en la ilustración 72.

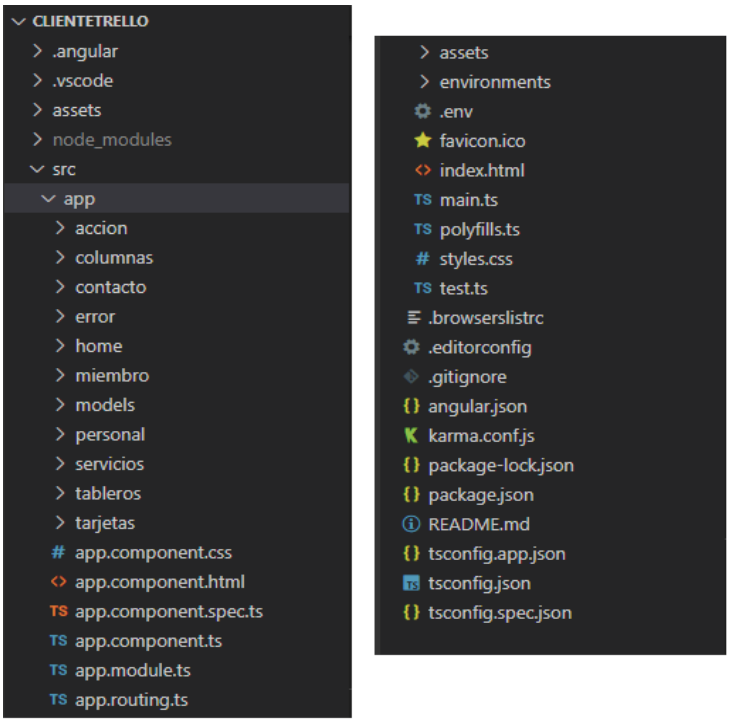

Ilustración 72: Estructura de ficheros y directorios del Front-end

En la carpeta “tableros” encontramos las clases necesarias que nos van a permitir realizar la búsqueda de un tablero y ver en detalle la información que tiene dicho tablero. Dichas clases las podemos ver en la ilustración 73.

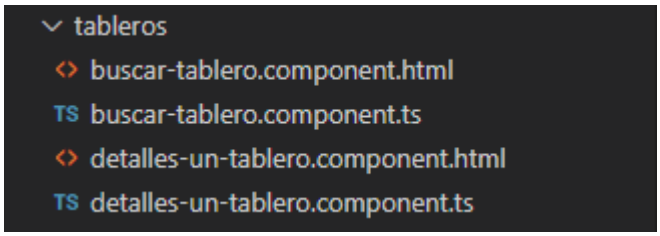

Ilustración 73: Ficheros del paquete tableros del cliente

La carpeta “tarjetas” tiene los archivos necesarios (ilustración 74), para listar todas las tarjetas que tiene un tablero y además poder ver la información de una tarjeta en concreto.

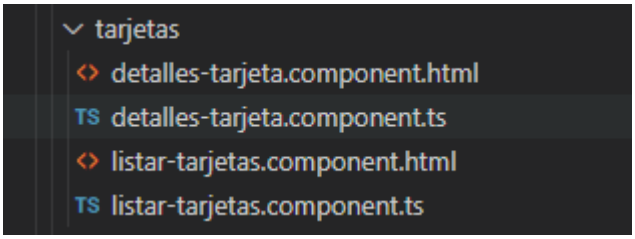

Ilustración 74: Ficheros del paquete tarjetas del cliente

La carpeta “miembro” está formada por las clases necesarias (ilustración 75) para ver en un listado los miembros que forman parte del tablero.

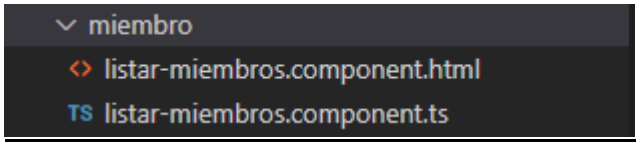

Ilustración 75: Ficheros del paquete miembros del cliente

La carpeta “listas”, al igual que las carpetas anteriores, tendrá las clases correspondientes para mostrar las listas que tiene un tablero y ver información más detallada de una lista. Su estructura la podemos ver en la ilustración 76.

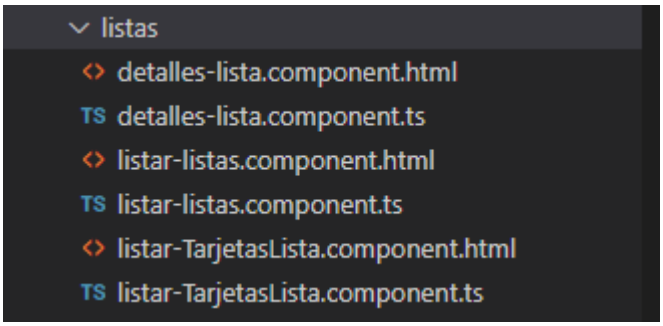

Ilustración 76: Ficheros del paquete listas del cliente

La carpeta “modelos” contiene los modelos de las clases que se van a utilizar, ver ilustración 77.

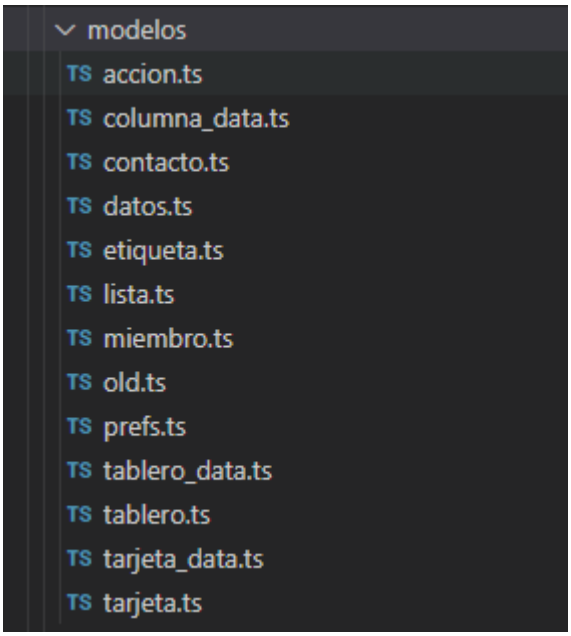

Ilustración 77: Ficheros del paquete modelos del cliente

La carpeta “servicios” (ilustración 78) contiene aquellos controladores de los métodos HTTP que vamos a utilizar. Cada archivo tendrá en el constructor la inicialización correspondiente a la ruta con la que tiene que obtener la información, en este caso, con el servidor.

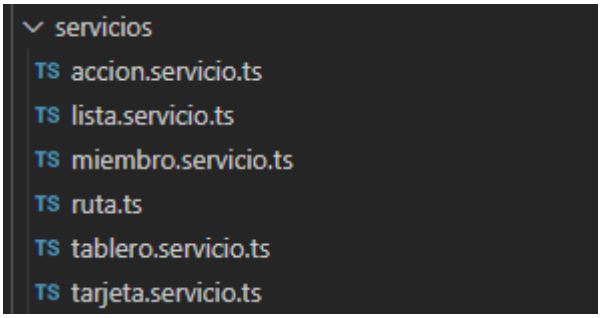

Ilustración 78: Ficheros del paquete servicios del cliente

La carpeta “personal” tiene la funcionalidad correspondiente para que un usuario pueda iniciar sesión con su cuenta de Trello (ver ilustración 79).

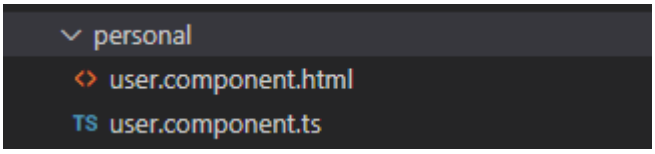

Ilustración 79: Ficheros del paquete personal del cliente

Si el inicio de sesión es correcto, automáticamente conducirá al usuario a los tableros a los que pertenece, en caso contrario mostrará un error. Para poder iniciar

sesión, la API de Trello nos ofrece el código correspondiente para permitir esta llamada. Para ello es necesario cargar los scripts proporcionados por la documentación de la API.

Como vemos en la ilustración 80, es necesario cargar la biblioteca JQuery y la url con la clave correspondiente del usuario que desea iniciar sesión.

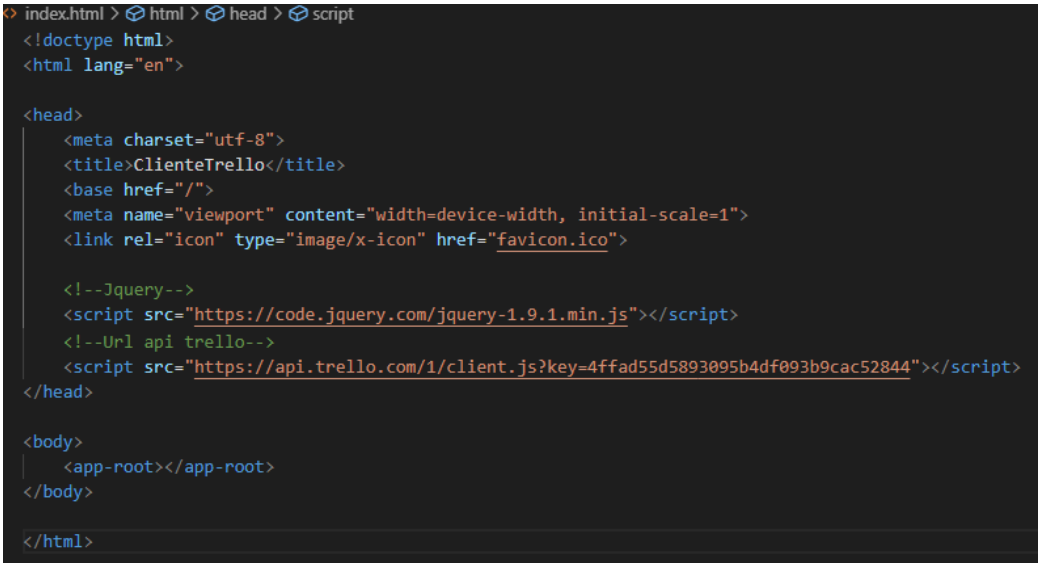

Ilustración 80: Scripts necesarios para iniciar sesión con Trello

La carpeta “home” (ilustración 81) hace referencia a la página inicial de la aplicación, justo la página que aparece cuando arrancamos la aplicación, su estructura aparece en la siguiente ilustración.

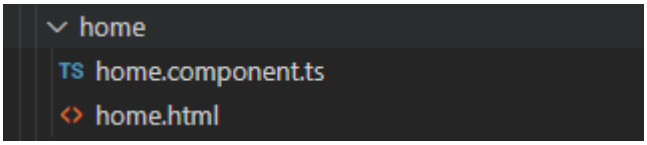

Ilustración 81: Ficheros del paquete home del cliente

7.1.4 Pruebas de verificación y validaciones

Una vez implementada la aplicación según el diseño inicial, vamos a proceder a realizar unas pruebas de verificación para ver que las historias de usuario detalladas en la primera iteración tienen la funcionalidad descrita y cumplen con los criterios establecidos.

Para comprobar lo descrito anteriormente, se van a realizar una serie de pruebas denominadas pruebas de caja negra [35]. Estas son un método de software para verificar que se cumplen las funcionalidades de la aplicación sin saber el código interno.

Prueba	Resultado	Valor
Iniciar Sesión		
El usuario accede a la aplicación	La aplicación muestra los tableros a los que pertenece el usuario	Correcto
El usuario no accede a la aplicación	La aplicación muestra un mensaje de error	Correcto

Tabla 20: Pruebas de validación para el inicio de sesión

Prueba	Resultado	Valor
Buscar un tablero mediante identificador		
Identificador válido	La aplicación redirige a mostrar la información correspondiente	Correcto
Identificador no válido	Se muestra un mensaje de error	Correcto

Tabla 21: Pruebas de validación para la búsqueda de un tablero

Prueba	Resultado	Valor
Mostrar las acciones		
Identificador de tablero válido	La aplicación muestra la información del tablero y las opciones a visualizar	Correcto
El usuario selecciona las acciones	Se muestra en una tabla las acciones realizadas, con su identificador, el tipo de acción y la fecha en la que se realizó	Correcto

Tabla 22: Pruebas de validación para mostrar las acciones

Prueba	Resultado	Valor
Mostrar información de una acción		
Identificador de tablero válido	La aplicación muestra la información del tablero y las opciones a visualizar	Correcto
El usuario selecciona las acciones	Se muestra en tablas, las acciones realizadas y su información	Correcto
El usuario selecciona una acción	Se muestra en una tabla toda la información que tiene una acción	Correcto

Tabla 23: Pruebas de validación para ver los detalles de una acción

Prueba	Resultado	Valor
Mostrar las tarjetas		
Identificador de tablero válido	La aplicación muestra la información del tablero y las opciones a visualizar	Correcto
El usuario selecciona las tarjetas	Se muestra en tablas, las tarjetas creadas y su información	Correcto

Tabla 24: Pruebas de validación para ver las tarjetas

Prueba	Resultado	Valor
Mostrar información de una tarjeta		
Identificador de tablero válido	La aplicación muestra la información del tablero y las opciones a visualizar	Correcto
El usuario selecciona las tarjetas	Se muestra en tablas, las tarjetas creadas y su información	Correcto
El usuario selecciona una tarjeta	Se muestra una tabla con la información concreta de una tarjeta	Correcto

Tabla 25: Pruebas de validación para ver los detalles de una tarjeta

Prueba	Resultado	Valor
Mostrar las listas		
Identificador de tablero válido	La aplicación muestra la información del tablero y las opciones a visualizar	Correcto
El usuario selecciona las listas	Se muestra en una tabla las listas creadas con su id y nombre	Correcto

Tabla 26: Pruebas de validación para ver las listas

Prueba	Resultado	Valor
Mostrar información de una lista		
Identificador de tablero válido	La aplicación muestra la información del tablero y las opciones a visualizar	Correcto
El usuario selecciona las listas	Se muestra en una tabla las listas creadas con su id y nombre	Correcto
El usuario selecciona ver la información	Se muestra en una tabla la información de la lista	Correcto
El usuario selecciona ver las tarjetas que tiene	Se muestra un listado de tablas con las tarjetas que componen esa lista	Correcto

Tabla 27: Pruebas de validación para ver los detalles de una lista

Prueba	Resultado	Valor
Mostrar los miembros		
Identificador de tablero válido	La aplicación muestra la información del tablero y las opciones a visualizar	Correcto
El usuario selecciona los miembros	Se muestra en tablas a cada miembro del tablero con su información	Correcto

Tabla 28: Pruebas de validación para ver los miembros

Prueba	Resultado	Valor
Mostrar el número de acciones realizadas por tipo		
El usuario selecciona ver acciones	Se muestra una tabla con las acciones agrupadas por tipo y el número de veces que se ha realizado cada una	Correcto

Tabla 29: Pruebas de validación para clasificar y contar las acciones realizadas por tipo

7.1.5 Diseño final de la interfaz

En este apartado se van a mostrar las imágenes que van a mostrar el diseño final de la aplicación web.

Página Principal

La página principal como podemos ver en la ilustración 82, se ha diseñado con un menú horizontal con los distintos apartados a los que se puede acceder y el logo. En el centro de la página se encuentra una breve descripción del Trabajo Fin de Grado.

Ilustración 82: Diseño final de la página principal

Inicio de Sesión

En la ilustración 83, el usuario puede iniciar sesión y seguidamente se muestra otra pantalla con la información de los tableros a los que el usuario pertenece (ilustración 84).

Ilustración 83: Diseño final para iniciar sesión

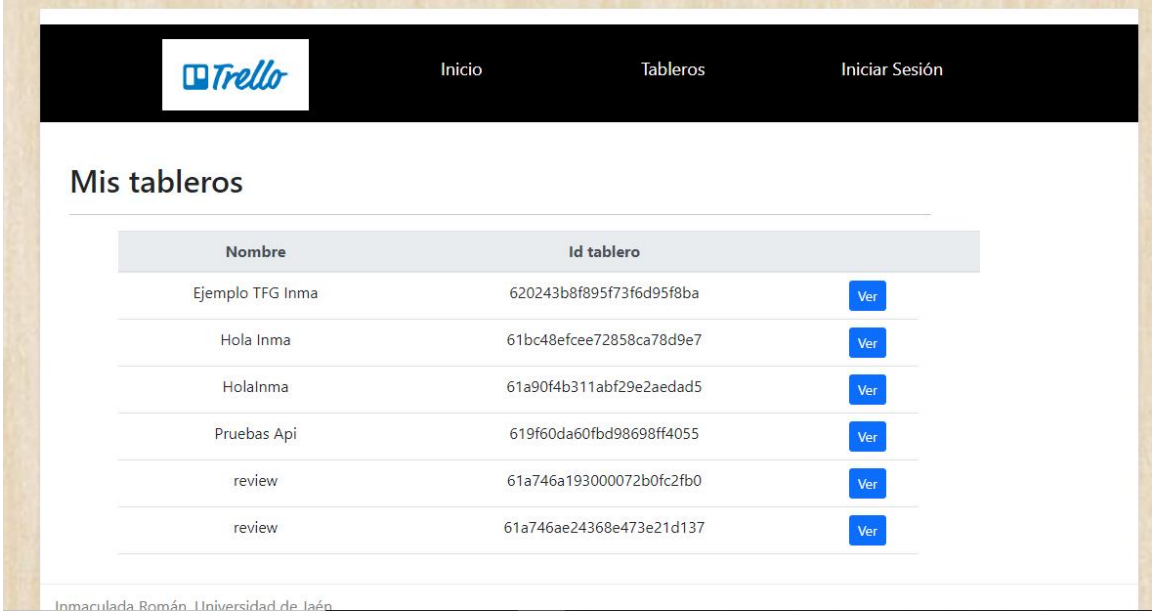

Ilustración 84: Diseño final para ver los tableros de un miembro

Buscar un tablero mediante un identificador

El diseño final para realizar la búsqueda de un tablero consta de un campo de búsqueda como podemos ver en la ilustración 85, en el cual el usuario tiene que introducir el identificador.

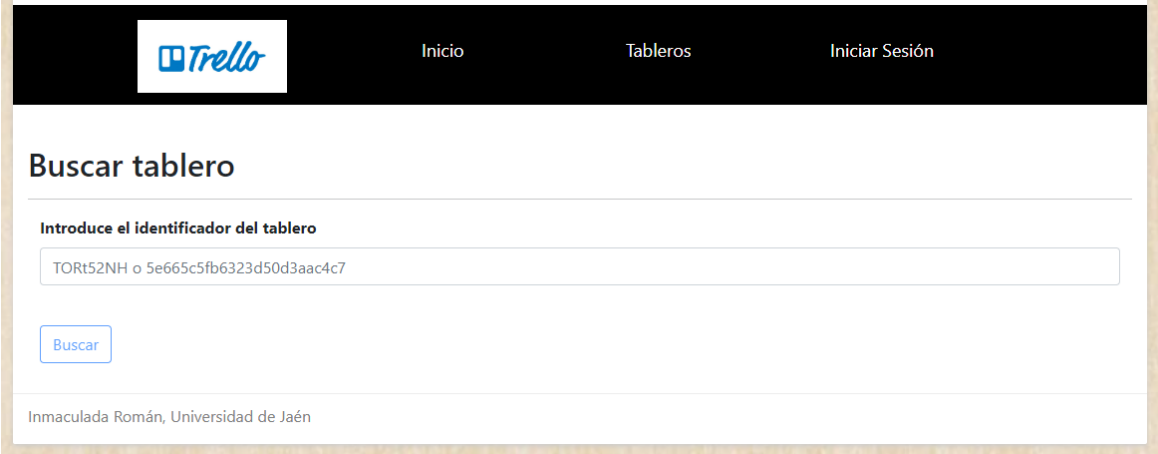

Ilustración 85: Diseño final para buscar un tablero

Mostrar las acciones de un tablero

Para ver las acciones que se han realizado en el tablero, la página consta de una tabla (ilustración 86) en la que se muestra un índice, el identificador y el tipo de acción realizada junto con la fecha en la que se realizó.

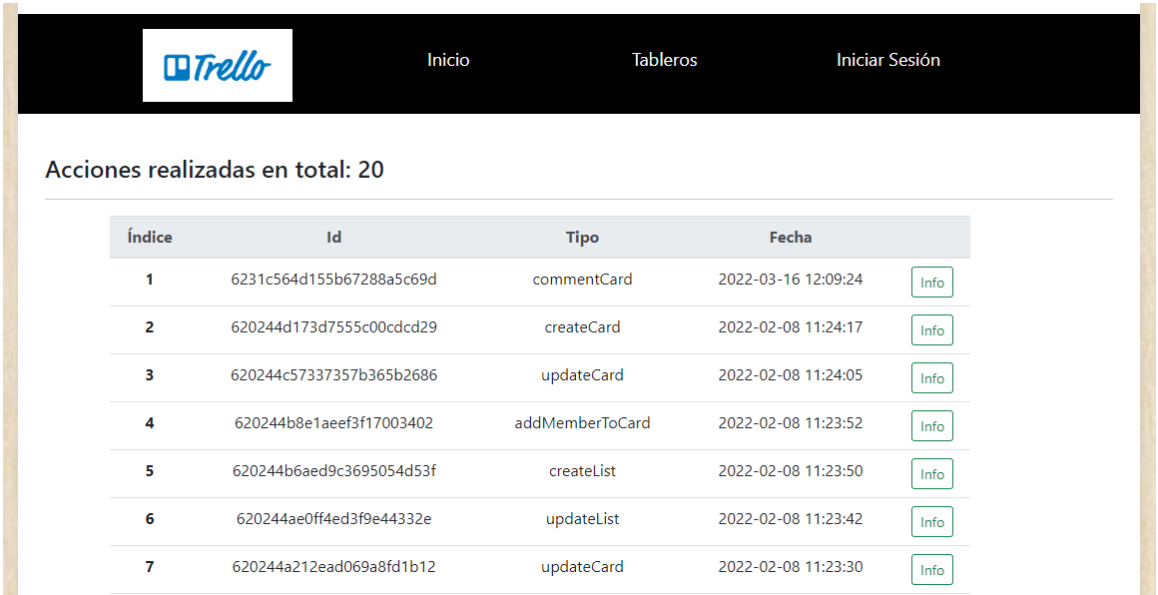

Ilustración 86: Diseño final para mostrar las acciones del tablero

Mostrar las tarjetas de un tablero

Para que el usuario vea las tarjetas que se han creado en el tablero, se ha elegido que cada tarjeta se represente mediante una tabla (ilustración 87), indicando los campos más importantes de una tarjeta, como lo son el nombre, el identificador del tablero al que pertenece, el identificador de la lista, etc.

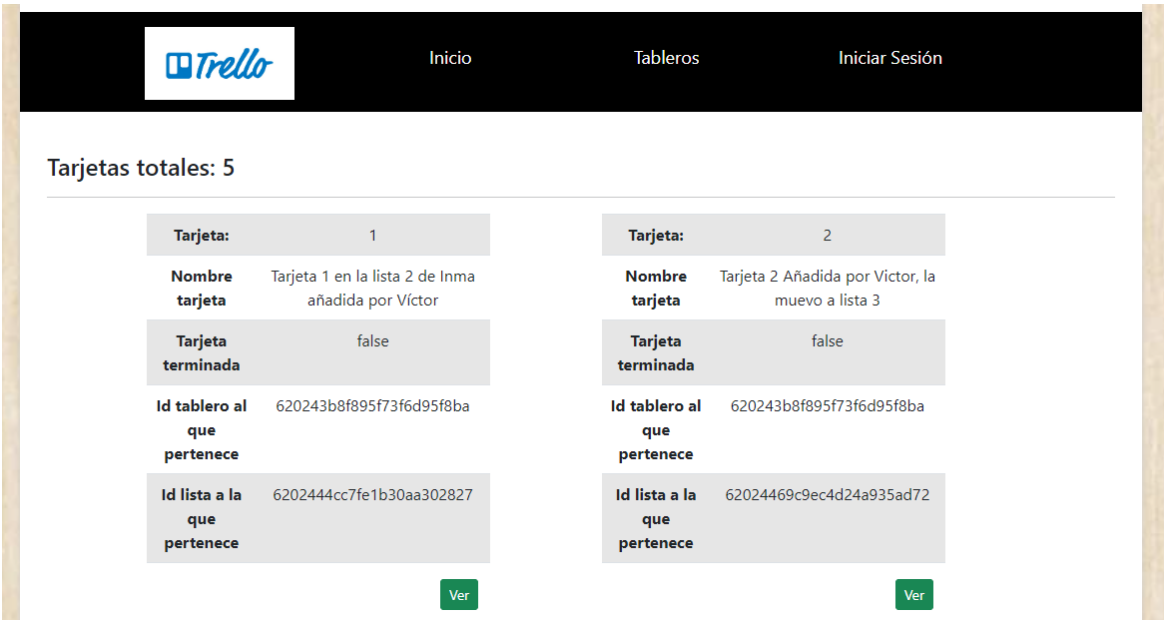

Ilustración 87: Diseño final para mostrar las tarjetas del tablero

Mostrar las listas de un tablero

Para mostrar las listas del tablero, se ha seguido utilizando una tabla (ilustración 88) en la que se muestra el nombre de la lista junto con su identificador y el identificador del tablero al que pertenece.

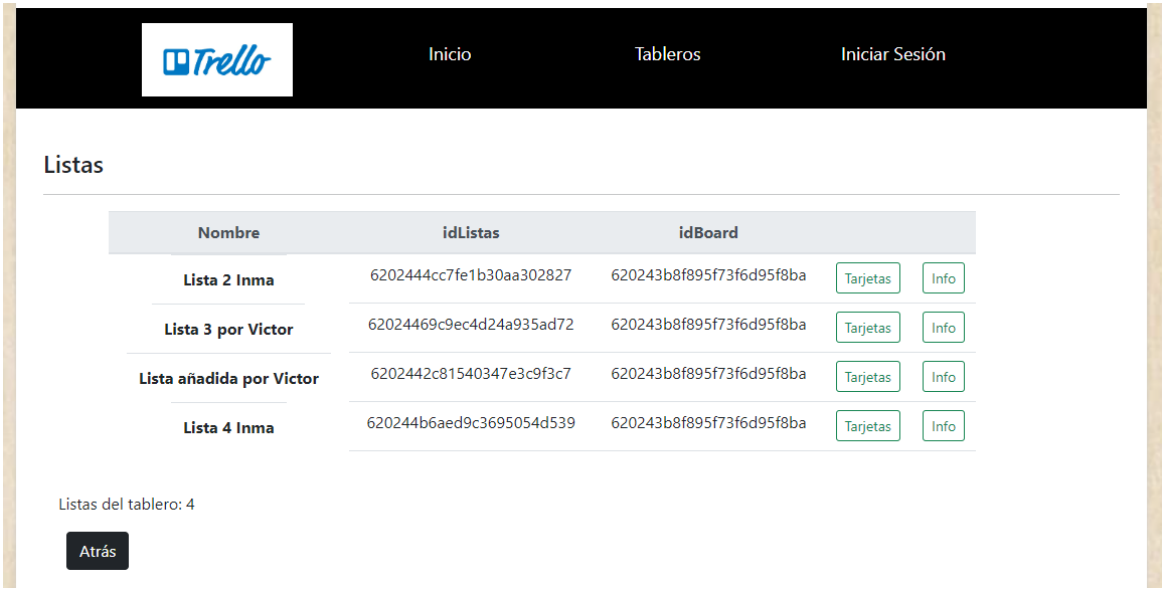

Ilustración 88: Diseño final para mostrar las listas del tablero

Mostrar la actividad de un tablero

Para ver toda la actividad realizada en el tablero, la página se ha dividido en dos secciones. La primera sección muestra un listado con las acciones agrupadas por tipo. En la ilustración 89 aparece una tabla en la que se indica cual es el tipo de acción que se ha llevado a cabo y el número que se ha realizado de cada una.

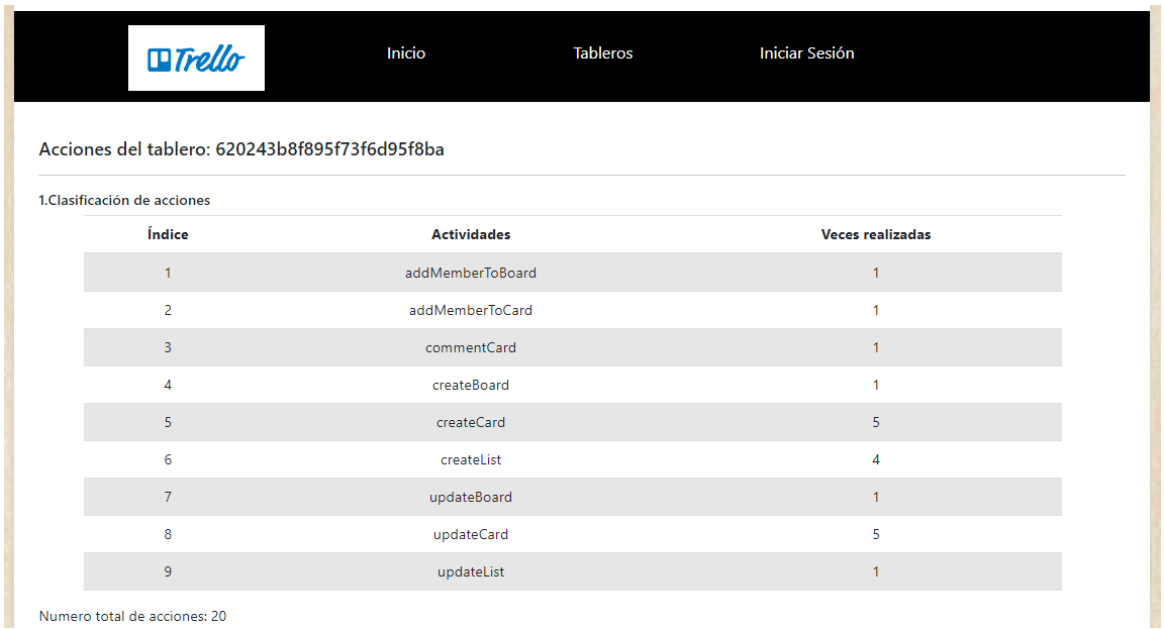

Ilustración 89: Diseño final para mostrar las acciones agrupadas por tipo

En la segunda sección encontramos la actividad que ha realizado cada miembro. Haciendo uso de una tabla (ilustración 90) en la que se indica el nombre del miembro

junto con su identificador, el número de acciones que ha realizado, indicado también en porcentaje.

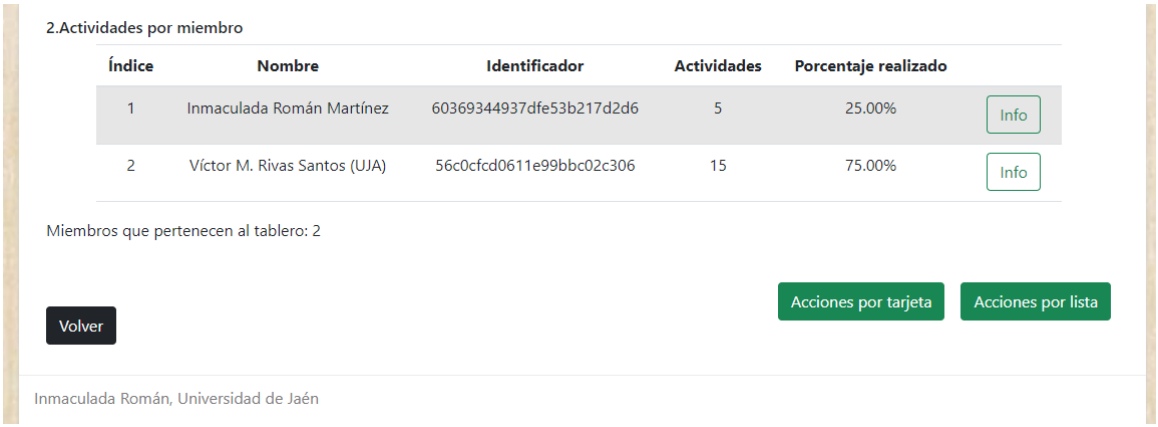

Ilustración 90: Diseño final para mostrar la actividad realizada por miembro

7.2 Segunda iteración

Una vez finalizada la primera iteración, vamos a continuar con la definición de las historias de usuario que vamos a llevar a cabo en esta iteración. Una vez descritas las historias de usuario pasaremos al apartado de diseño, donde crearemos los diagramas de secuencia para explicar la comunicación existente entre las diferentes partes.

Lo siguiente será llevar a cabo la implementación y las pruebas de verificación para comprobar que la aplicación funciona correctamente.

Por último, habrá un apartado en el que mostraremos el diseño final de la aplicación.

7.2.1 Historias de usuario

Identificador	10
Título	Mostrar el número de acciones realizadas por miembro
Descripción	Como auditor de un tablero Trello quiero saber de cada persona que pertenece al tablero cuántas acciones ha realizado de cada tipo para poder cuantificar el trabajo que están haciendo.
Valor	S
Esfuerzo	S
Criterios de aceptación	-Obtener un listado de las acciones totales. -Agrupar por id de miembro.

Tabla 30: HU Mostrar el número de acciones realizadas por miembro

Identificador	04
Título	Mostrar el porcentaje de actividad de cada miembro
Descripción	Como auditor de un tablero Trello quiero saber cuántas personas han realizado alguna acción en el tablero para conocer el porcentaje de personas que realmente trabajan en é
Valor	S
Esfuerzo	M
Criterios de aceptación	-Obtener un listado de las acciones totales. -Agrupar por id de miembro. -Sacar porcentaje.

Tabla 31: HU Mostrar el porcentaje de actividad de cada miembro

Identificador	08
Título	Mostrar la fecha de la actividad realizada
Descripción	Como auditor de un tablero Trello quiero saber en qué fecha se ha realizado cada actividad para poder observar el flujo de trabajo temporal.
Valor	C
Esfuerzo	M
Criterios de aceptación	-Obtener un listado de las acciones y su fecha.

Tabla 32: HU Mostrar la fecha de la actividad realizada

Identificador	12
Título	Mostrar las acciones filtradas por tarjetas
Descripción	Como auditor de un tablero Trello quiero poder filtrar de todas las acciones cuáles son las realizadas en tarjetas, para ver los cambios que se han realizado.
Valor	S
Esfuerzo	L
Criterios de aceptación	-Obtener las acciones totales. -Restringir la búsqueda para agrupar por tarjeta.

Tabla 33: HU Mostrar las acciones filtradas por tarjetas

Identificador	14
Título	Mostrar numéricamente los cambios realizados en una tarjeta
Descripción	Como auditor de un tablero Trello quiero contar los cambios realizados en una tarjeta para ver el flujo de trabajo seguido.
Valor	M
Esfuerzo	S
Criterios de aceptación	-Obtener las acciones filtradas por tarjeta. -Contar y mostrar numéricamente los cambios realizados en cada tarjeta.

Tabla 34: HU Mostrar numéricamente los cambios realizados en una tarjeta

Identificador	16
Título	Mostrar el porcentaje de actividad realizado en cada tarjeta
Descripción	Como auditor de un tablero Trello quiero ver el porcentaje de actividad realizado en cada tarjeta para saber la actividad realizada.
Valor	M
Esfuerzo	M
Criterios de aceptación	-Obtener las acciones filtradas por tarjeta. -Mostrar numéricamente los cambios realizados en cada tarjeta. -Mostrar un porcentaje de los cambios realizados.

Tabla 35: HU Mostrar el porcentaje de actividad realizado en cada tarjeta

Identificador	06
Título	Mostrar las acciones y los cambios por los que ha pasado una tarjeta
Descripción	Como auditor de un tablero Trello quiero poder visualizar los cambios de una tarjeta para identificar el flujo de trabajo de esta.
Valor	S
Esfuerzo	L
Criterios de aceptación	-Obtener las acciones filtradas por tarjeta. -Seleccionar una tarjeta y mostrar los cambios realizados.

Tabla 36: HU Mostrar las acciones y los cambios por los que ha pasado una tarjeta

Identificador	13
Título	Mostrar las acciones filtradas por listas
Descripción	Como auditor de un tablero Trello quiero poder filtrar de todas las acciones cuáles son las realizadas en listas, para ver los cambios que se han realizado.
Valor	S
Esfuerzo	L
Criterios de aceptación	-Obtener las acciones totales. -Restringir la búsqueda para agrupar por listas.

Tabla 37: HU Mostrar las acciones filtradas por listas

Identificador	15
Título	Mostrar numéricamente los cambios realizados en cada lista
Descripción	Como auditor de un tablero Trello quiero los cambios realizados en una lista para ver que ha pasado por ella.
Valor	M
Esfuerzo	S
Criterios de aceptación	-Obtener las acciones agrupadas por listas. -Contar y mostrar numéricamente los cambios que se han realizado sobre cada lista.

Tabla 38: HU Mostrar numéricamente los cambios realizados en cada lista

Identificador	18
Título	Mostrar el porcentaje de actividad realizado en una lista
Descripción	Como auditor de un tablero Trello quiero ver el porcentaje de cambios que se ha realizado sobre una lista para saber que parte pertenece del total.
Valor	S
Esfuerzo	M
Criterios de aceptación	-Obtener las acciones agrupadas por listas. -Contar y mostrar numéricamente los cambios que se han realizado sobre cada lista. -Sacar el porcentaje realizado en cada lista.

Tabla 39: HU Mostrar el porcentaje de actividad realizado en una lista

Identificador	07
Título	Mostrar las acciones y los cambios por los que ha pasado una lista
Descripción	Como auditor de un tablero Trello quiero poder visualizar los cambios de una lista para identificar el flujo de trabajo de esta
Valor	S
Esfuerzo	L
Criterios de aceptación	-Obtener las acciones agrupadas por listas. -Al seleccionar una lista, que aparezca un listado mostrando por los cambios que ha pasado

Tabla 40: HU Mostrar las acciones y los cambios por los que ha pasado una lista

Identificador	03
Título	Buscar acciones
Descripción	Como auditor de un tablero Trello quiero filtrar las acciones por distintos campos, como persona, tipo de actividad, fecha, etc. para conseguir más información sobre el flujo de trabajo.
Valor	M
Esfuerzo	XXL
Criterios de aceptación	-Realizar la búsqueda en un formulario mediante tipo de acción o miembro.

Tabla 41: HU Buscar acciones mediante tipo de acción o miembro

7.2.2 Diseño

En el diseño, vamos a seguir con lo explicado en la iteración anterior. Vamos a utilizar los diagramas de secuencia para explicar la comunicación entre las distintas partes que componen la aplicación.

HU10- Mostrar el número de acciones realizadas por miembro

Objetivo: Obtenidos los miembros que componen el tablero, contar el número de acciones que ha realizado cada uno.

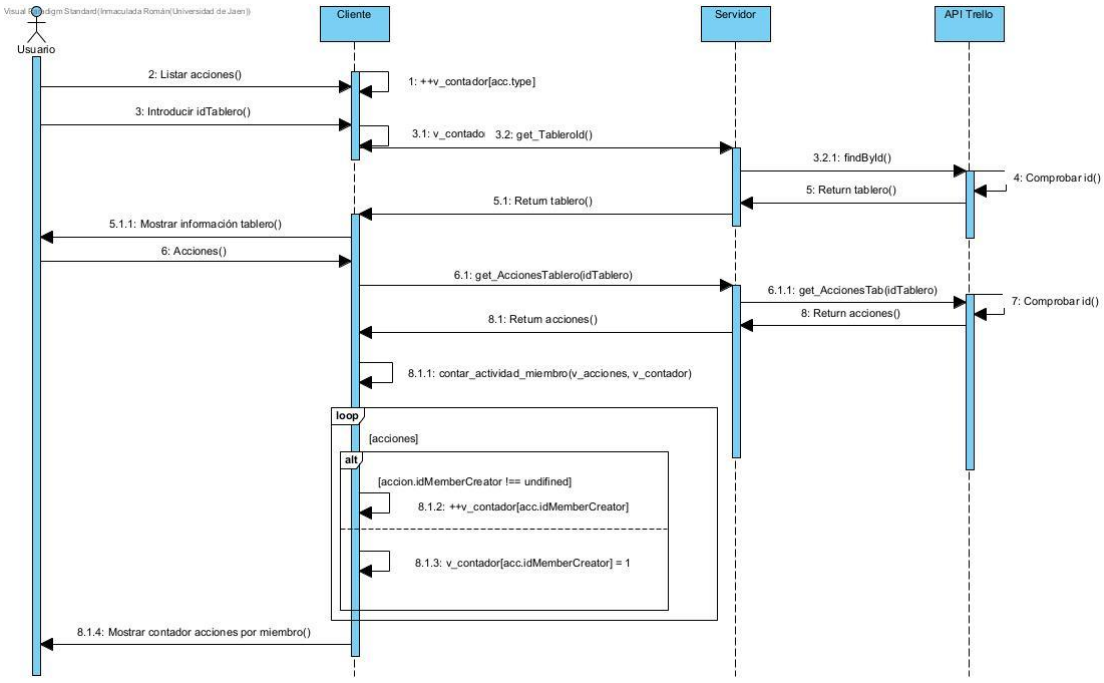

Ilustración 91: Diagrama de secuencia para mostrar el número de acciones realizadas por miembro

Para obtener el número de acciones realizadas por miembro necesitamos:

I. El usuario quiere visualizar numéricamente la actividad que ha realizado cada miembro.
II. El cliente solicita al servidor información sobre los miembros indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/members
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra al usuario un listado con los miembros que pertenecen al tablero y el número de acciones que ha realizado cada uno en formato HTML.

HU04- Mostrar el porcentaje de actividad de cada miembro

Objetivo: Calculado el número de acciones que ha realizado cada miembro, mostrarlo en porcentaje

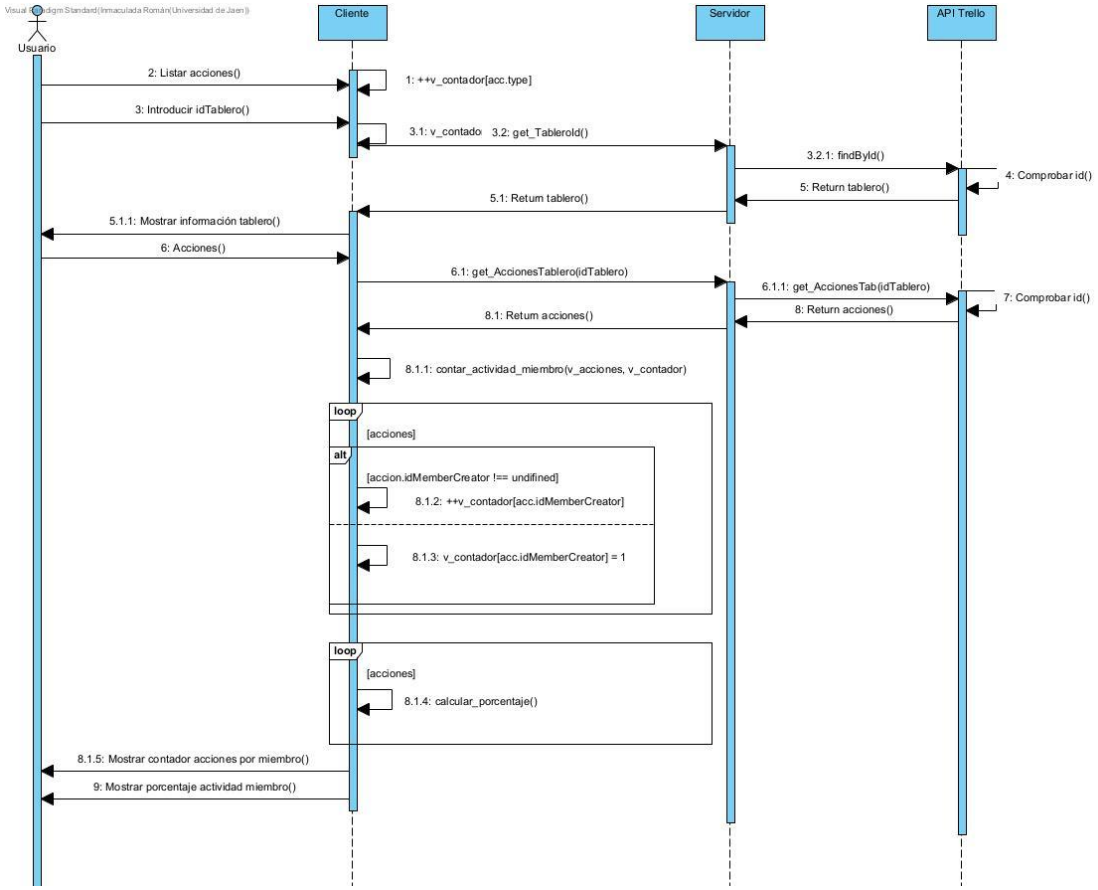

Ilustración 92: Diagrama de secuencia para mostrar el porcentaje de actividad de cada miembro

Para obtener el porcentaje de actividad de cada miembro necesitamos:

I. El usuario quiere visualizar el porcentaje de la actividad que ha realizado cada miembro.
II. El cliente solicita al servidor información sobre los miembros indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/members
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.

VI. El cliente muestra al usuario junto a cada miembro, el porcentaje de la actividad realizado por cada miembro.

HU-08- Mostrar la fecha de la actividad realizada

Objetivo: Mostrar la fecha en la que se realizó cada acción

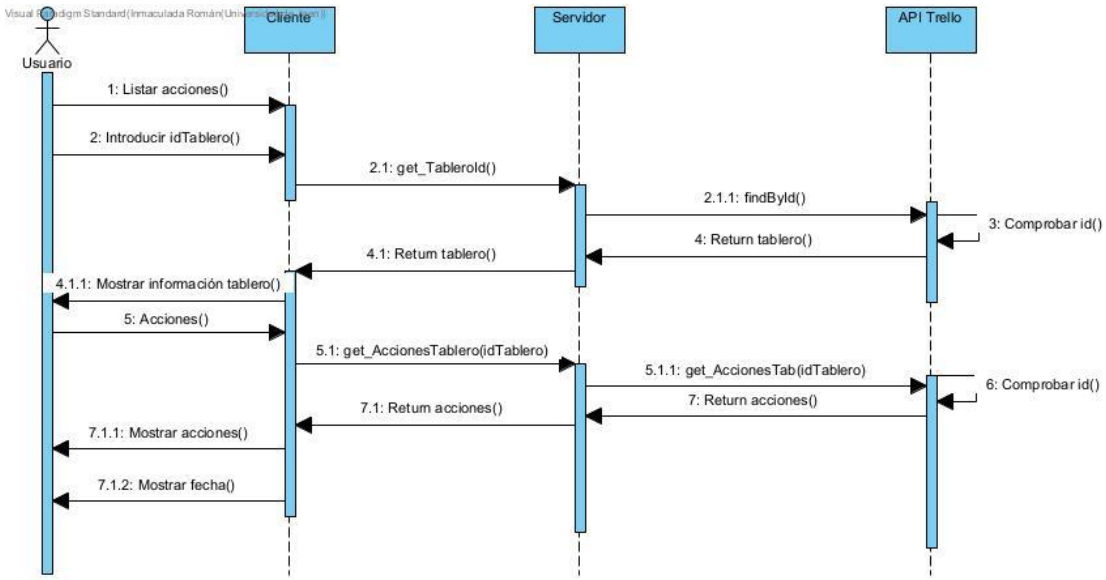

Ilustración 93: Diagrama de secuencia para mostrar la fecha en la que se realizó cada acción

Para obtener la fecha en la que se realizó cada acción necesitamos:

I. El usuario quiere visualizar junto a cada acción, la fecha cuando se realizó cada una.
II. El cliente solicita al servidor información sobre las acciones indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra al usuario cada acción realizada junto con su fecha en formato HTML.

HU12- Mostrar las acciones filtradas por tarjetas

Objetivo: Obtenidas todas las acciones, centrar la búsqueda para sacar las relacionadas con las tarjetas

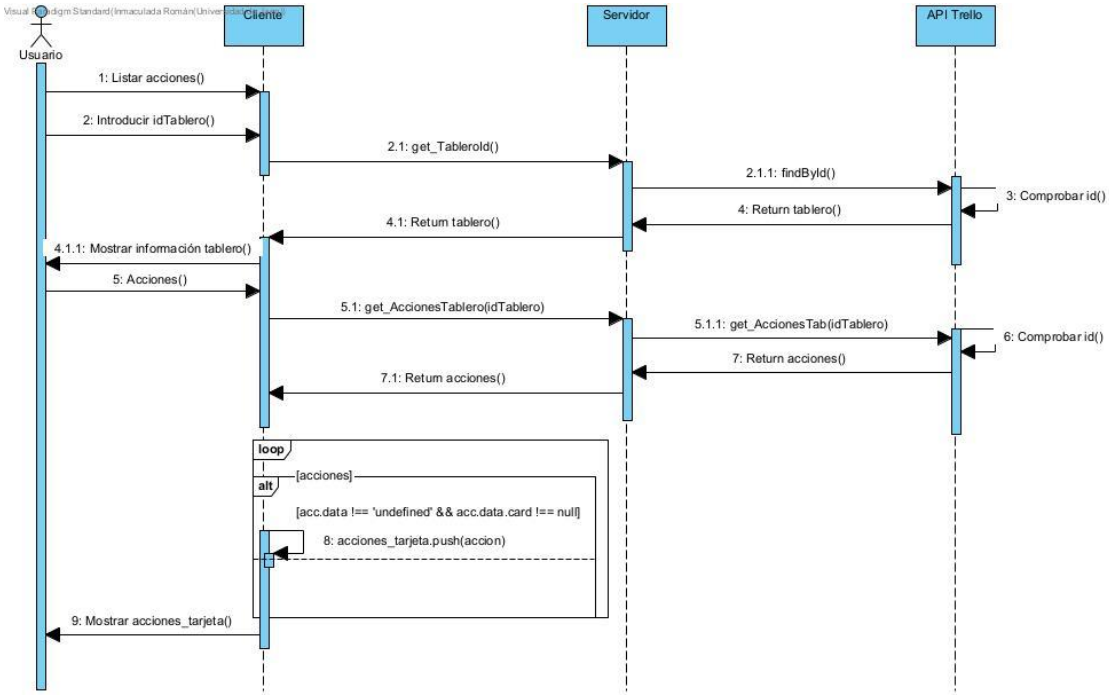

Ilustración 94: Diagrama de secuencia para mostrar las acciones filtradas por tarjetas

Para obtener las acciones realizadas sobre las tarjetas necesitamos:

I. El usuario quiere obtener cuales son las acciones que han sido realizadas sobre las tarjetas.
II. El cliente solicita al servidor información sobre las acciones indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra al usuario un listado de las acciones que tienen que ver con las tarjetas.

HU14- Mostrar numéricamente los cambios realizados en cada tarjeta

Objetivo: Obtenidas las acciones por tarjeta, contabilizar los cambios que se han realizado en cada una.

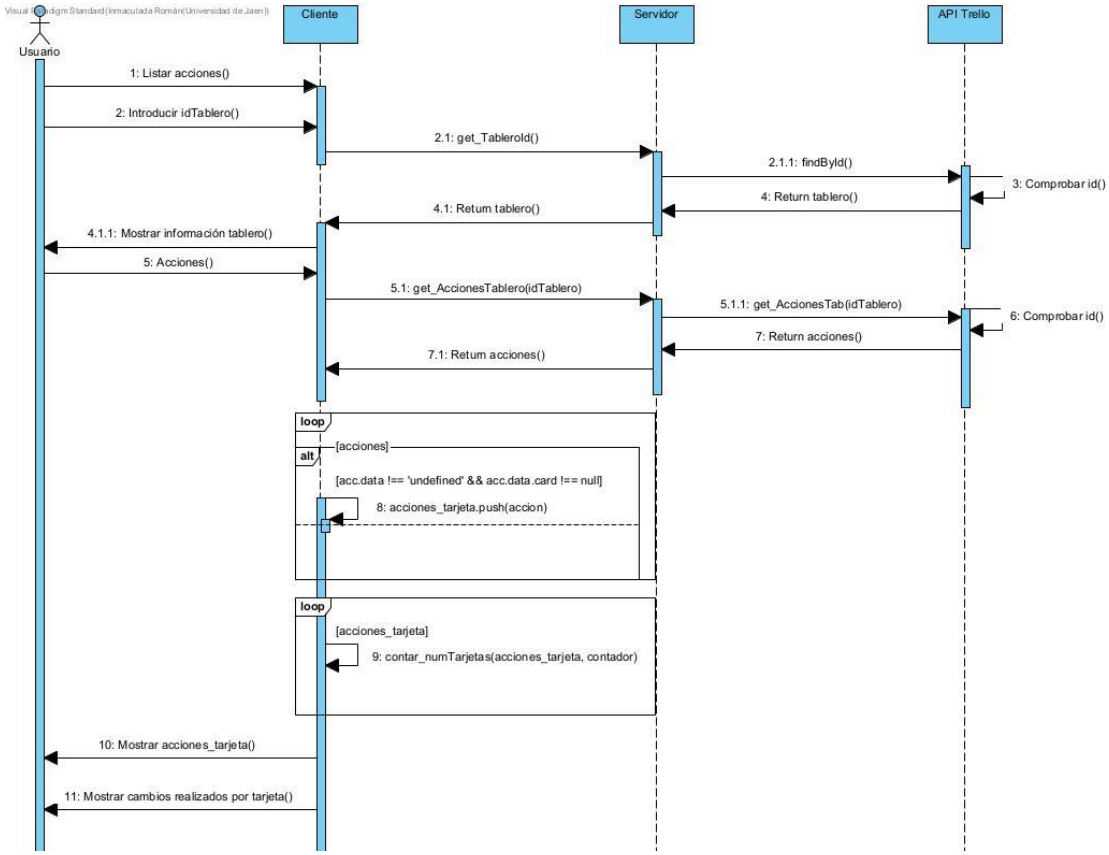

Ilustración 95: Diagrama de secuencia para mostrar numéricamente los cambios realizados en cada tarjeta

Para mostrar numéricamente los cambios realizados sobre cada tarjeta necesitamos:

I. El usuario quiere obtener numéricamente los cambios realizados sobre las tarjetas.
II. El cliente solicita al servidor información sobre las acciones indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.

VI. El cliente muestra al usuario un listado de las tarjetas y el número de cambios que se han realizado en cada una en formato HTML.

HU16: Mostrar el porcentaje de actividad realizado en cada tarjeta

Objetivo: Una vez contabilizados los cambios en cada tarjeta, mostrar el porcentaje de actividad

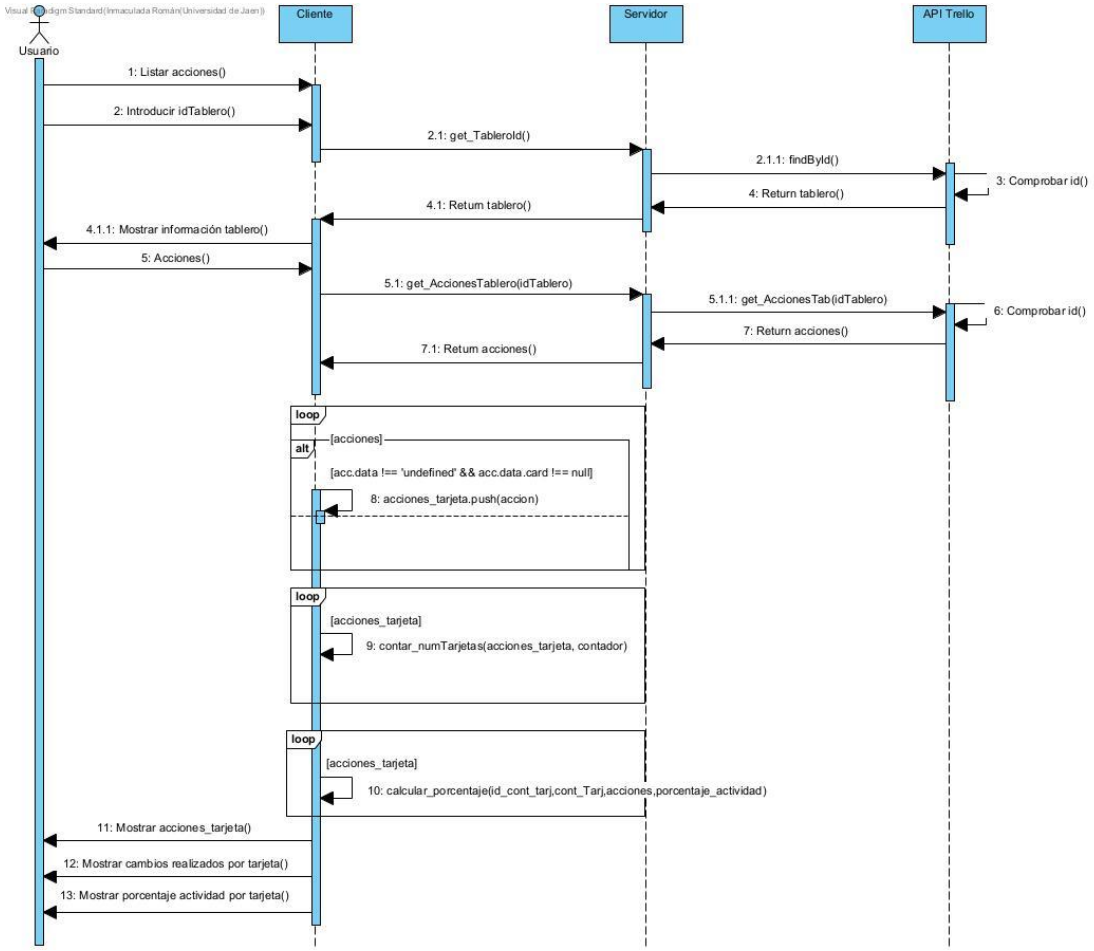

Ilustración 96: Diagrama de secuencia para mostrar el porcentaje de actividad realizado en cada tarjeta

Para mostrar el porcentaje de actividad realizado sobre cada tarjeta necesitamos:

I. El usuario quiere obtener cual es el porcentaje de los cambios realizados sobre las tarjetas.
II. El cliente solicita al servidor información sobre las acciones indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions

IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra al usuario un listado con las tarjetas y el porcentaje de actividad que se ha realizado en cada una en formato HTML.

HU06- Mostrar las acciones y los cambios realizados en una tarjeta

Objetivo: Al entrar en una tarjeta para ver el número de cambios realizados, mostrar un listado con los cambios realizados y la fecha en la que se realizó cada uno.

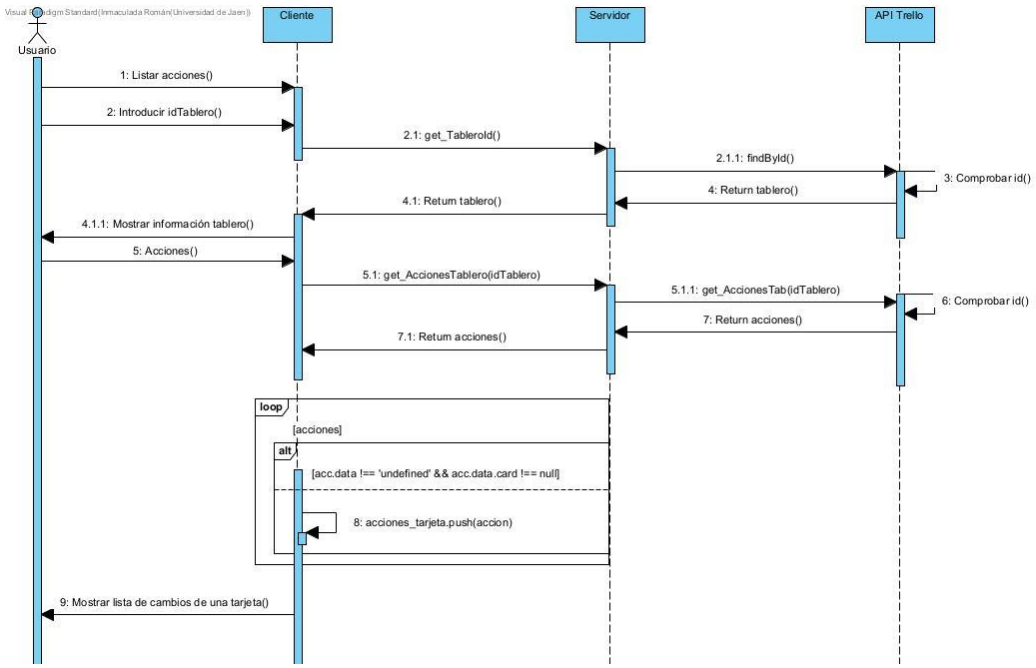

Ilustración 97: Diagrama de secuencia para mostrar las acciones y los cambios por los que ha pasado una tarjeta

Para mostrar las acciones y los cambios realizados sobre cada tarjeta necesitamos:

I. El usuario quiere obtener cuales son las acciones y los cambios realizados sobre las tarjetas.
II. El cliente solicita al servidor información sobre las acciones indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.

V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra al usuario un listado con todas las acciones y los cambios que se han realizado en una tarjeta en formato HTML.

HU13- Mostrar las acciones filtradas por listas

Objetivo: Obtenidas todas las acciones, centrar la búsqueda para sacar las relacionadas con las listas

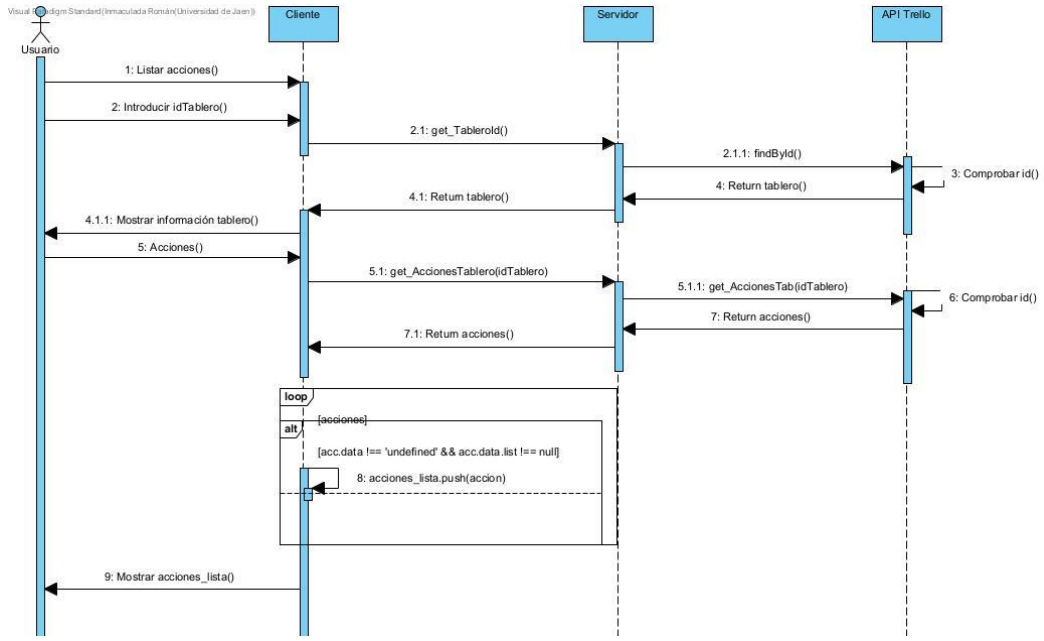

Ilustración 98: Diagrama de secuencia para mostrar las acciones filtradas por listas

Para obtener las acciones realizadas sobre las listas necesitamos:

I. El usuario quiere obtener cuales las acciones que se han realizado sobre las listas.
II. El cliente solicita al servidor información sobre las acciones indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra al usuario un listado con las acciones que están relacionadas con las listas en formato HTML.

HU15- Mostrar numéricamente los cambios realizados en cada lista

Objetivo: Filtrada la búsqueda de las acciones por listas, contabilizar los cambios que se han realizado en cada lista

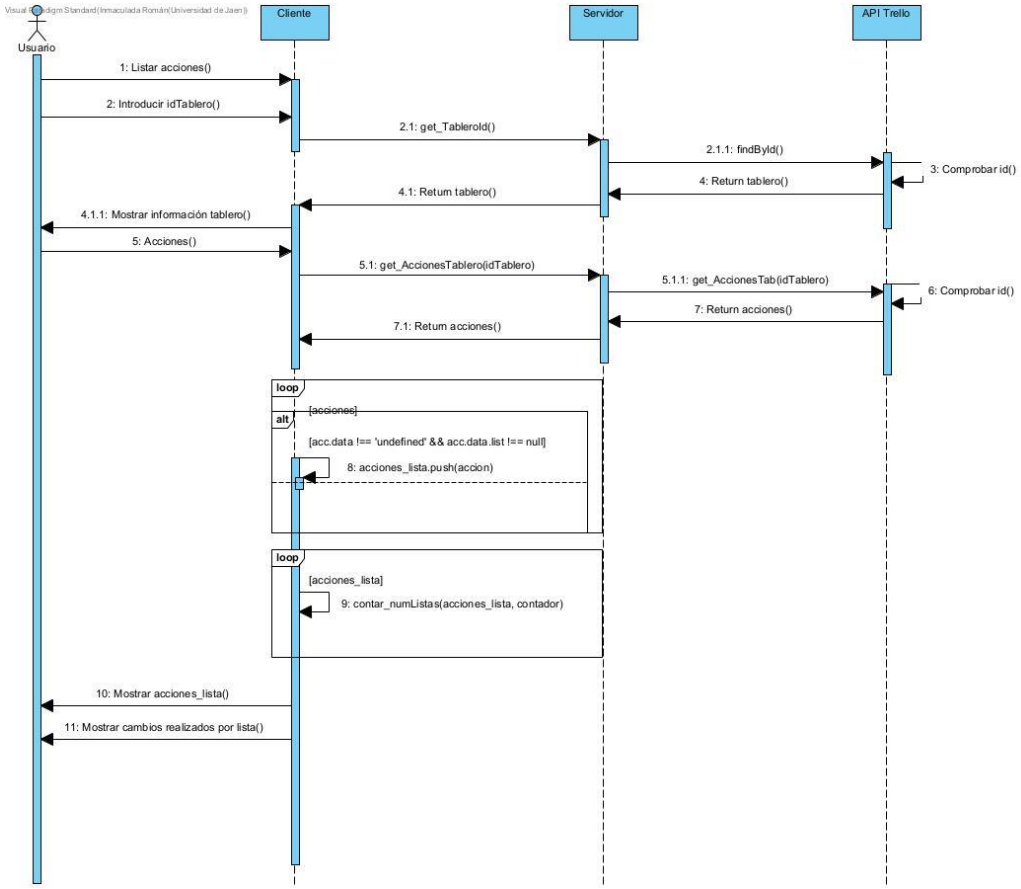

Ilustración 99: Diagrama de secuencia para mostrar numéricamente los cambios realizados en cada lista

Para obtener el número de cambios realizados en cada lista necesitamos:

I. El usuario quiere obtener numéricamente los cambios realizados sobre las listas.
II. El cliente solicita al servidor información sobre las acciones indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.

VI. El cliente muestra al usuario un listado con las acciones que están relacionadas con las listas y el número de cambios realizados en formato HTML.

HU18- Mostrar el porcentaje de actividad realizado en cada lista

Objetivo: Contabilizados los cambios en cada lista, calcular y mostrar el porcentaje.

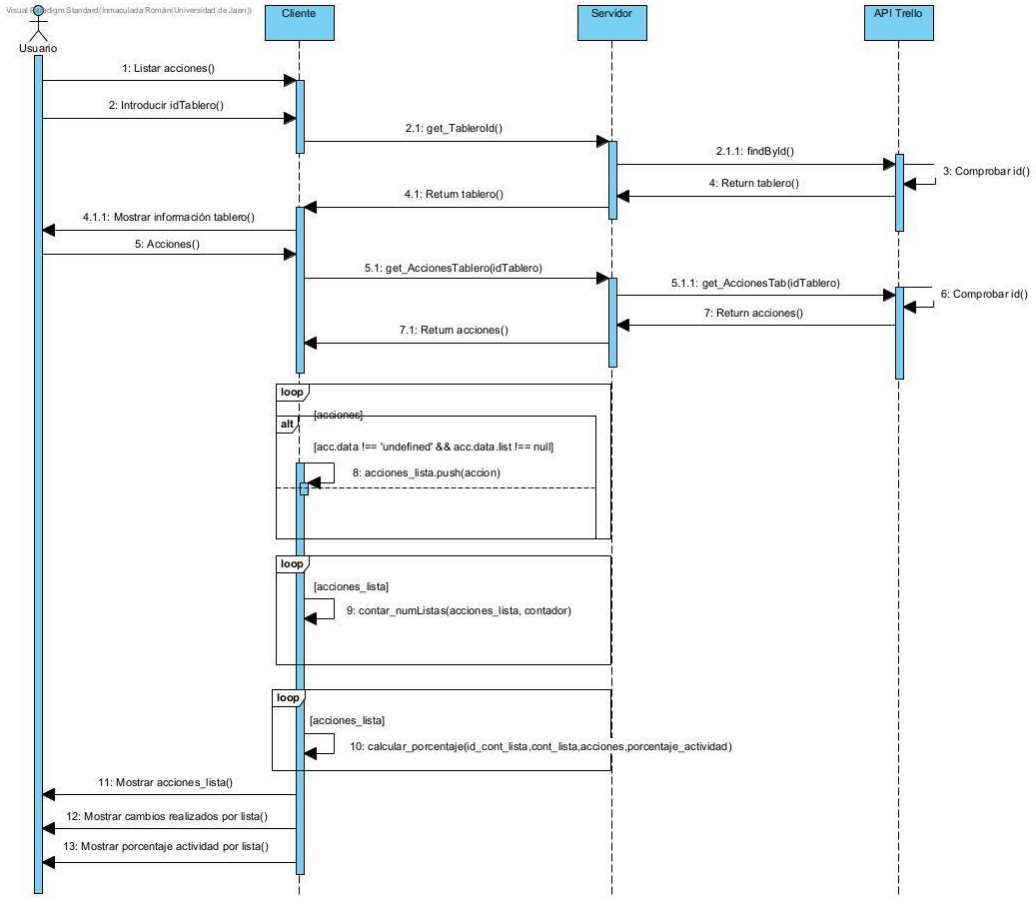

Ilustración 100: Diagrama de secuencia para mostrar el porcentaje de actividad realizado en cada lista

Para obtener el porcentaje de actividad realizado en cada lista necesitamos:

I. El usuario quiere obtener el porcentaje de los cambios realizados sobre las listas.
II. El cliente solicita al servidor información sobre las acciones indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions

IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra al usuario un listado con las acciones que están relacionadas con las listas y el porcentaje de cambios realizados en formato HTML.

HU07- Mostrar las acciones y los cambios realizados en una lista

Objetivo: Contabilizados los cambios de una lista, mostrar un listado con todos los cambios realizados

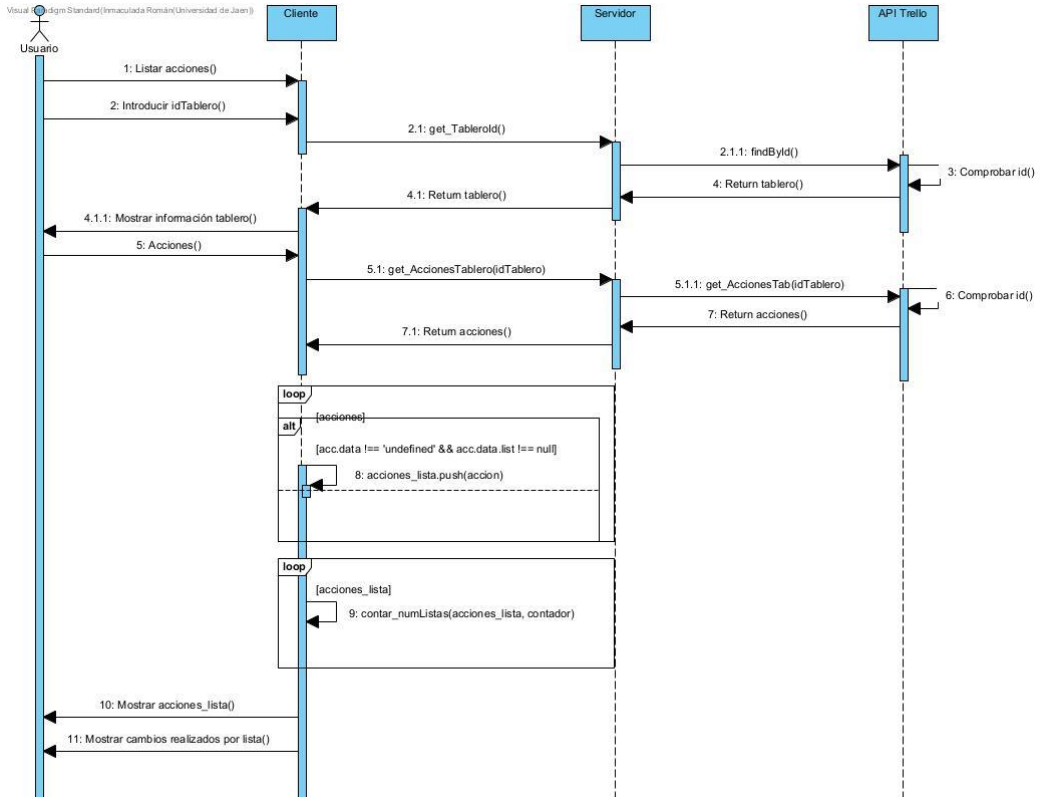

Ilustración 101: Diagrama de secuencia para mostrar las acciones y los cambios por los que ha pasado una lista

Para obtener las acciones y los cambios realizados en una lista necesitamos:

I. El usuario quiere obtener un listado con las acciones y los cambios realizados en cada lista.
II. El cliente solicita al servidor información sobre las acciones indicándole el identificador del tablero.

III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra al usuario un listado con todas las acciones y los cambios que se han realizado en una lista en formato HTML.

HU03- Buscar acciones

Objetivo: Obtener todas las acciones, y realizar una búsqueda para obtener por tipo y miembro.

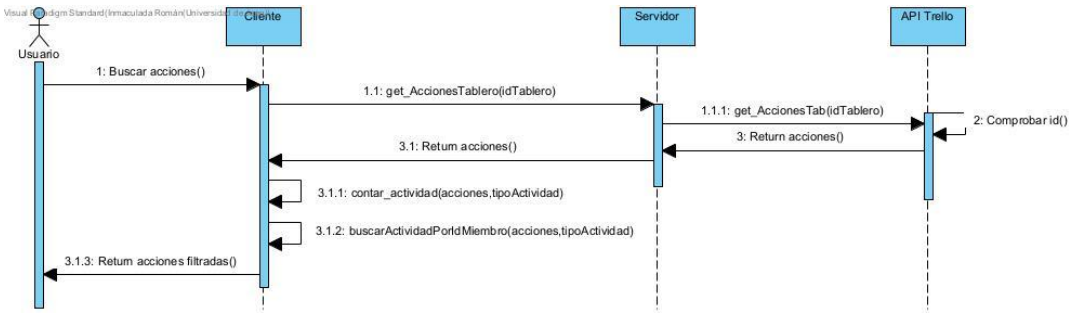

Ilustración 102: Diagrama de secuencia para buscar acciones por tipo de acción y miembro

Para buscar acciones necesitamos:

I. El usuario quiere todas las acciones poder realizar una búsqueda rápida sobre ellas.
II. El cliente solicita al servidor información sobre las acciones indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra al usuario un listado según la búsqueda realizada en formato HTML.

7.2.3 Implementación

Siguiendo con la implementación, una vez obtenida la funcionalidad básica de la aplicación, vamos a continuar para poder obtener más información. Como anotación he de decir, que en la primera iteración se consiguió la funcionalidad completa del servidor.

Implementación del cliente

Siguiendo con la carpeta “acción” vamos a crear los archivos necesarios para contabilizar tanto las acciones realizadas, como la actividad que ha realizado cada miembro. Partiendo de las historias de usuario comentadas anteriormente, vamos a desglosar las partes más importantes de cada una. Con los últimos cambios realizados, la estructura de la carpeta sería la siguiente:

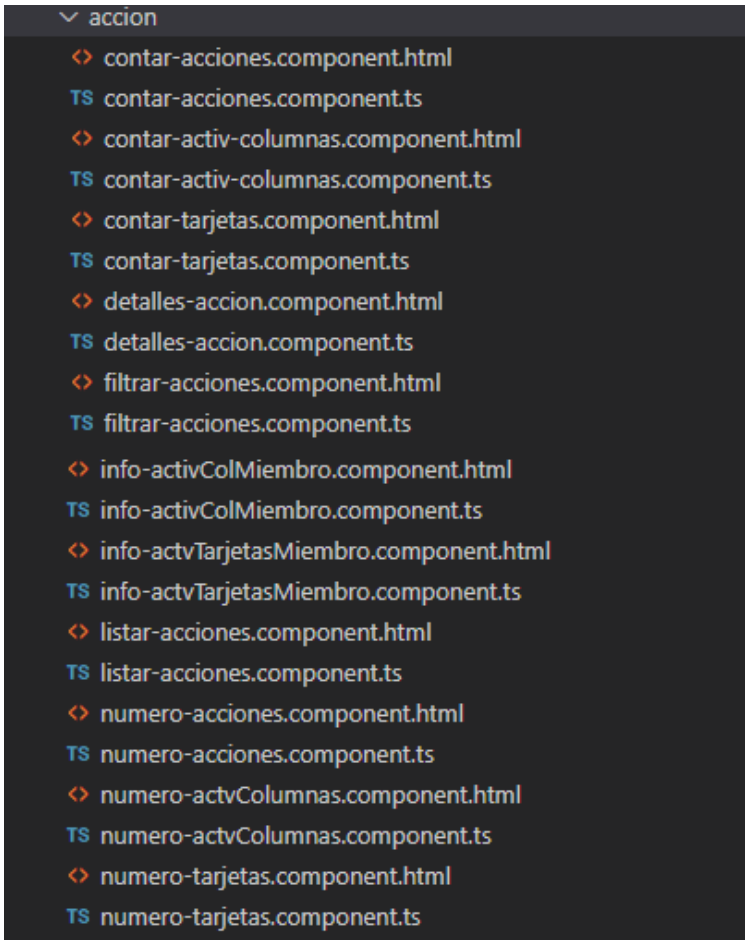

Ilustración 103: Ficheros del paquete acción del cliente

Lo primero de todo es de todas las acciones que se han realizado, agruparlas según el tipo de acción. Para ello, las acciones cuentan con un campo denominado “type”. La siguiente funcion cuenta las acciones por tipo y va introduciendo en un nuevo array el identificador de la accion y el numero de veces que se ha realizado cada una.

```
contar_actividad(v_acciones: any, v_contador: any) {
        for (let acc of v_acciones) {
            if (typeof v_contador[acc.type] !== 'undefined') {
                ++v_contador[acc.type];
            } else {
                v_contador[acc.type] = 1;
            }
        }
    }
```

Para obtener la actividad que ha realizado cada miembro, las acciones cuentan con un campo denominado “idMemberCreator”, por ello con el código que vemos a continuación podemos ir almacenando en un nuevo array el identificador del miembro si no está definido, y el número de acciones que ha realizado.

```
contar_actividad_miembro(v_acciones: any, v_contador: any) {
        for (let acc of v_acciones) {
            if (typeof v_contador[acc.idMemberCreator] !== 'undefined') {
                ++v_contador[acc.idMemberCreator];
            } else {
                v_contador[acc.idMemberCreator] = 1;
            }
        }
    }
```

Si queremos desglosar y entrar más profundamente en la actividad que ha realizado cada miembro, tenemos que agrupar en un nuevo array, aquellas acciones que coincidan con el identificador del usuario, como podemos ver en el siguiente código.

```
 this.acciones_miembro = this.acciones.filter((e:any) =>{
     return e.memberCreator.id === this.idUsuario;
})
```

Después solo nos quedaría contar de ese array de “acciones_miembro” cuantas acciones ha realizado ese miembro y mostrarlo tanto numéricamente como en porcentaje.

Para calcular el porcentaje de acciones que ha realizado cada miembro, hay que dividir el número de acciones realizadas entre las acciones totales, y todo eso multiplicarlo por 100.

Para obtener las acciones que se han realizado de tipo tarjeta, las acciones cuentan con un campo denominado data. Este objeto cuenta el tablero, la tarjeta y la lista, sobre la que se ha realizado.

```
this.acciones.forEach((acc) => {
   if (typeof acc.data !== 'undefined' && acc.data.card !== null) {
          this.acciones_tarjeta.push(acc);
   }
})
```

Como vemos en el código de arriba, lo primero será obtener todas las acciones, y a partir de ahí comprobar que los datos de la tarjeta tengan información y si es así, introducimos la acción en un nuevo array, en este caso “acciones_tarjeta”.

Como ya tenemos las acciones filtradas por tarjeta, vamos a contabilizar los cambios que se han realizado en cada tarjeta y mostrarlo tanto numéricamente como en porcentaje.

Para contar los cambios, la función difiere de la anterior como podemos ver a continuación, ya que hay acceder al campo “data” y comprobar que hay información. El porcentaje se realizaría como se ha explicado anteriormente.

```
contar_numTarjetas(v_acciones:any,v_contador:any){
        for (let a of v_acciones) {
           if(typeof a.data !== 'undefined' && a.data.card !== null) {
                if(typeof v_contador[a.data.card.id] !== 'undefined') {
                    ++v_contador[a.data.card.id] ;
                }else{
                    v_contador[a.data.card.id] =1;
                }
            }
        }
    }
```

Lo siguiente va a ser desglosar los cambios por los que ha pasado una tarjeta. Para esto, vamos a ir comparando los campos “old” y los correspondientes a “data”. Esto se hace más concretamente el fichero HTML.

Lo relacionado con mostrar las acciones filtradas por listas y contar los cambios que se han realizado es prácticamente igual que lo explicado anteriormente, pero cambiando en el código “card” por “list”.

Por último, nos quedaría como implementar la búsqueda de acciones. Para ello es necesario obtener las acciones totales que se han realizado en el tablero y a partir de ahí es necesario contar cuantas acciones se han hecho de cada tipo, para posteriormente que nos aparezcan según el tipo de acción seleccionado.

```
contar_actividad(v_acciones: any, v_contador: any) {
        for (let acc of v_acciones) {
            if (typeof v_contador[acc.type] !== 'undefined') {
                ++v_contador[acc.type];
            } else {
                v_contador[acc.type] = 1;
            }
        }
    }
```

Para buscar por miembro y tenerlo todo agrupado en el mismo vector, a partir de las acciones totales obtenidas ir contabilizando cuantas ha realizado cada miembro para así obtener la búsqueda deseada. El código lo podemos ver a continuación.

```
buscarAcccionesPorIdMiembro(v_acciones:any, v_tipo:any){
        for (let acc of v_acciones) {
            if(typeof v_tipo[acc.memberCreator.id] !== 'undefined'){
                ++v_tipo[acc.memberCreator.fullName];
             }else{
                 v_tipo[acc.memberCreator.fullName] = 1;
             }
        }
    }
```

7.2.4 Pruebas de verificación y validaciones

Una vez implementado todo lo anterior vamos a comprobar que su funcionamiento es el correcto, haciendo uso de las pruebas de caja negra, explicado en el apartado 7.1.4

Prueba	Resultado	Valor
Mostrar actividad realizada por los miembros		
Identificador válido	La aplicación redirige a mostrar la información correspondiente	Correcto
El usuario selecciona ver actividad	Mostrará una tabla con las acciones realizadas, clasificadas por tipo y cuantas se han realizado de cada una	Correcto
	La aplicación mostrará una tabla con los miembros que forman parte del tablero, indicando numéricamente el número de actividades realizadas y el porcentaje	Correcto

Tabla 42: Pruebas de validación para ver las acciones que ha realizado cada miembro

Prueba	Resultado	Valor
Mostrar las acciones filtradas por tarjetas		
Identificador válido	La aplicación redirige a mostrar la información correspondiente	Correcto
El usuario selecciona ver actividad	Mostrará una tabla con las acciones realizadas filtradas por tipo, cuantas se ha realizado de cada una, y una tabla con la actividad de cada miembro	Correcto
El usuario selecciona ver acciones por tarjeta	La aplicación mostrará una tabla con las tarjetas que se han realizado y mostrará tanto numéricamente como en porcentaje, los cambios realizados en cada una	Correcto

Tabla 43: Pruebas de validación para ver las acciones filtradas por tarjetas

Prueba	Resultado	Valor
Mostrar las acciones y los cambios realizados en cada tarjeta		
Identificador válido	La aplicación redirige a mostrar la información correspondiente	Correcto
El usuario selecciona ver actividad	Mostrará una tabla con las acciones realizadas filtradas por tipo, cuantas se ha realizado de cada una, y una tabla con la actividad de cada miembro	Correcto
El usuario selecciona ver acciones por tarjeta	La aplicación mostrará una tabla con las tarjetas que se han realizado y mostrará tanto numéricamente como en porcentaje, los cambios realizados en cada una	Correcto
El usuario selecciona información de una tarjeta	La aplicación muestra en una tabla las acciones ejecutadas en una tarjeta junto con la fecha, y desglosa en otra tabla, los cambios	Correcto

Tabla 44: Pruebas de validación para ver los cambios que ha sufrido una tarjeta

Prueba	Resultado	Valor
Mostrar las acciones filtradas por listas		
Identificador válido	La aplicación redirige a mostrar la información correspondiente	Correcto
El usuario selecciona ver actividad	Mostrará una tabla con las acciones realizadas filtradas por tipo, cuantas se ha realizado de cada una, y otra tabla con la actividad de cada miembro	Correcto
El usuario selecciona ver acciones por listas	La aplicación mostrará una tabla con las listas que se han creado y mostrará los cambios realizados en cada una, con número y porcentaje. En otra tabla se mostrará la actividad que ha realizado cada miembro	Correcto

Tabla 45: Pruebas de validación para ver las acciones filtradas por listas

Prueba	Resultado	Valor
Mostrar las acciones y los cambios realizados en una lista		
Identificador válido	La aplicación redirige a mostrar la información correspondiente	Correcto
El usuario selecciona ver actividad	Mostrará una tabla con las acciones realizadas filtradas por tipo, cuantas se ha realizado de cada una, y otra tabla con la actividad de cada miembro	Correcto
El usuario selecciona ver acciones por lista	La aplicación mostrará una tabla con las listas que se han creado y mostrará los cambios realizados en cada una, con número y porcentaje. En otra tabla se mostrará la actividad que ha realizado cada miembro	Correcto
El usuario selecciona información de una lista	La aplicación muestra en una tabla las acciones ejecutadas en una lista junto con la fecha, y desglosa en otra tabla, los cambios	Correcto

Tabla 46: Pruebas de validación para ver los cambios que ha sufrido cada lista

Prueba	Resultado	Valor
Buscar acciones		
Identificador válido	La aplicación redirige a mostrar la información correspondiente	Correcto
El usuario selecciona filtrar acciones	La aplicación muestra un desplegable para seleccionar la búsqueda ya sea por tipo de acciones o miembro del equipo	Correcto

Tabla 47: Pruebas de validación para buscar las acciones por tipo y miembro

7.2.5 Diseño final de la interfaz

Una vez terminadas las pruebas de verificación y validación se van a indicar cuál es el diseño final de aplicación en esta iteración.

Mostrar el número y el porcentaje de actividad de un miembro

Para ver más concretamente la actividad que ha realizado cada miembro, se ha diseñado una página en la que se detalla cuál ha sido el trabajo realizado por cada miembro. La actividad de un miembro se divide en dos secciones.

Como podemos observar en la ilustración 104, la primera sección contiene una tabla con todas las acciones que ha realizado un miembro. Las acciones vienen descritas por el tipo de acción junto con el número de veces que ha realizado ese tipo de acción y el porcentaje de actividad.

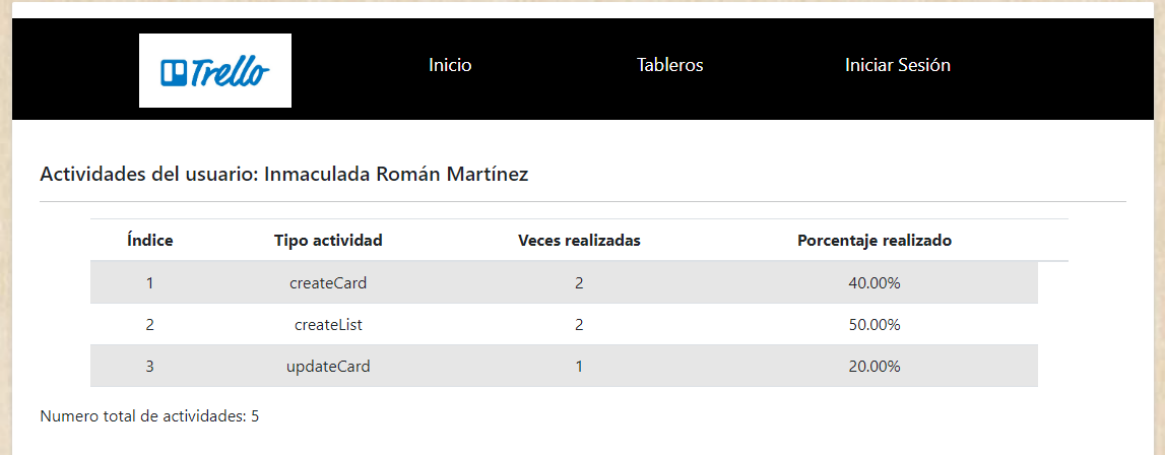

Ilustración 104: Diseño final para mostrar el número y el porcentaje de actividad de un miembro

En la segunda sección, encontramos una tabla (ilustración 105) en la que se desglosa cada acción que ha realizado, junto con la fecha en la que se realizó.

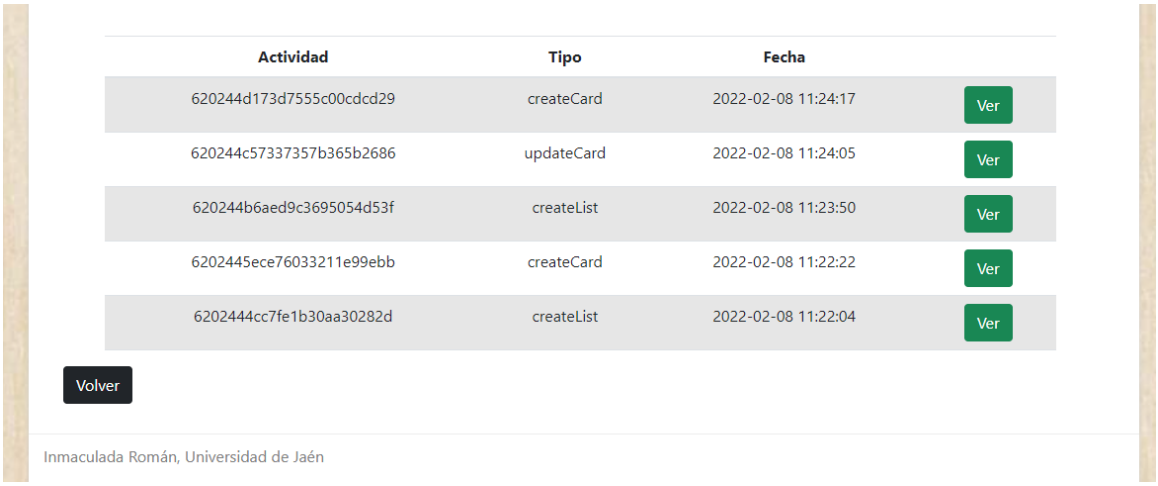

Ilustración 105: Diseño final para mostrar las acciones realizadas por un miembro

Mostrar las acciones filtradas por tarjetas

Para centrarnos solo en la actividad que se ha realizado sobre las tarjetas, se ha diseñado una página que muestra al usuario dicha información. La página se divide en dos secciones.

La primera sección (ilustración 106), muestra una tabla con todas las tarjetas que se han creado en el tablero. Estas van numeradas con un índice, el nombre e identificador de la tarjeta junto con los cambios que se han realizado en cada una y el porcentaje de actividad.

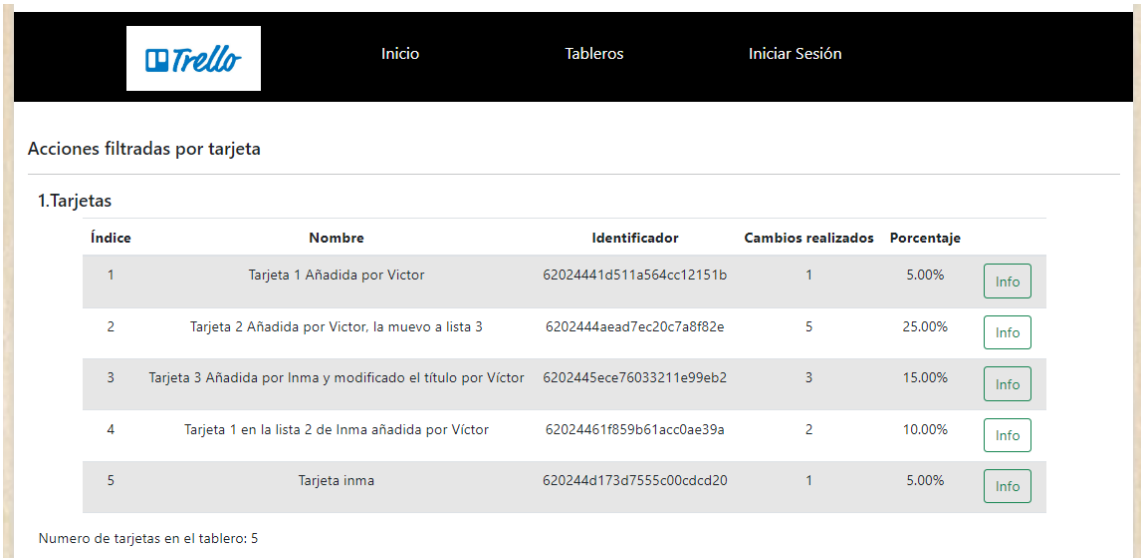

Ilustración 106: Diseño final para mostrar las acciones filtradas por tarjetas

En la segunda sección encontramos una tabla con la actividad que ha realizado cada miembro sobe cada tarjeta, como se ha explicado anteriormente. Ver ilustración 107.

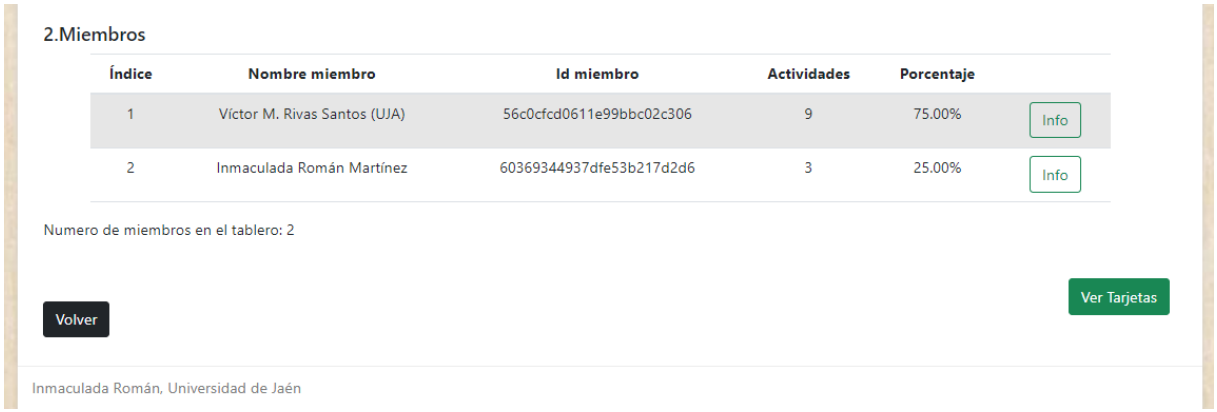

Ilustración 107: Diseño final para mostrar la actividad realizada por cada miembro

Mostrar las acciones y los cambios realizados en una tarjeta

Para mostrar más detalladamente los cambios por los que ha pasado una tarjeta, se ha creado la siguiente página. Referente a una tarjeta en concreto como podemos ver en la ilustración 108, la primera sección contiene una tabla indicando las distintas acciones que se han realizado sobre ella. Esta tabla contiene un índice para después buscar la acción en la sección 2, el identificador y tipo de acción y la fecha en la que se realizó.

Ilustración 108: Diseño final para mostrar las acciones realizadas en una tarjeta

En la segunda sección como podemos ver en la ilustración 109, se muestra una tabla en la que se desglosan los cambios. Esta contiene el índice de la acción y cuál ha sido la modificación realizada.

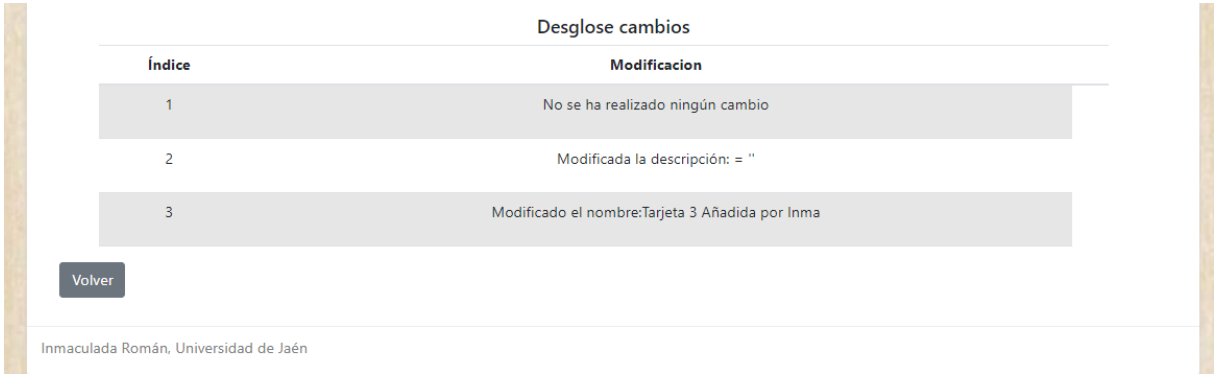

Ilustración 109: Diseño final para desglosar los cambios realizados en una tarjeta

Mostrar las acciones filtradas por listas

El diseño de esta página es significativamente igual que el empleado con las tarjetas, como podemos ver en las siguientes ilustraciones 110 y 111.

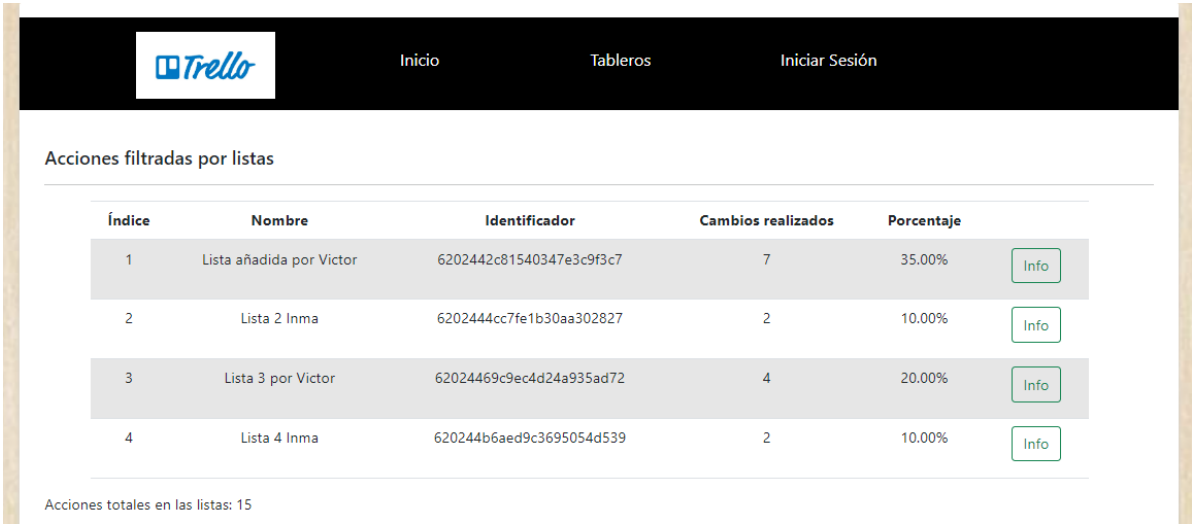

Ilustración 110: Diseño final para mostrar las acciones filtradas por listas

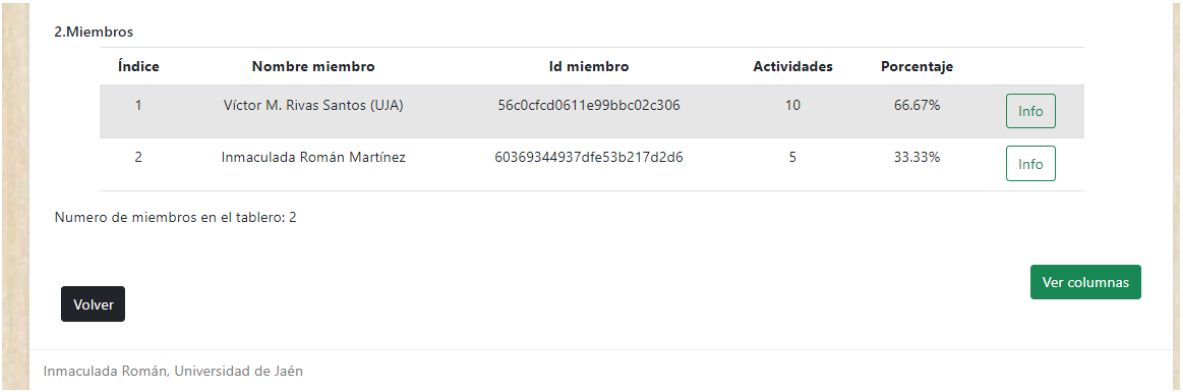

Ilustración 111: Diseño final para mostrar la actividad realizada por cada miembro

Mostrar las acciones y los cambios realizados en una lista

Si el usuario desea ver los cambios por los que ha pasado una lista, el diseño es igual que en las tarjetas. Para ver las dos secciones en las que se divide la página podemos ver las ilustraciones 112 y 113.

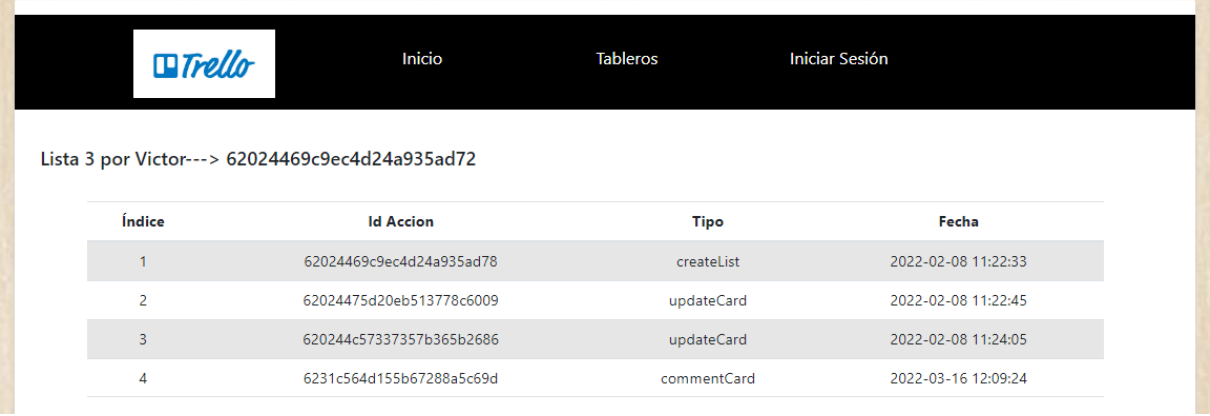

Ilustración 112: Diseño final para mostrar las acciones realizadas en una lista

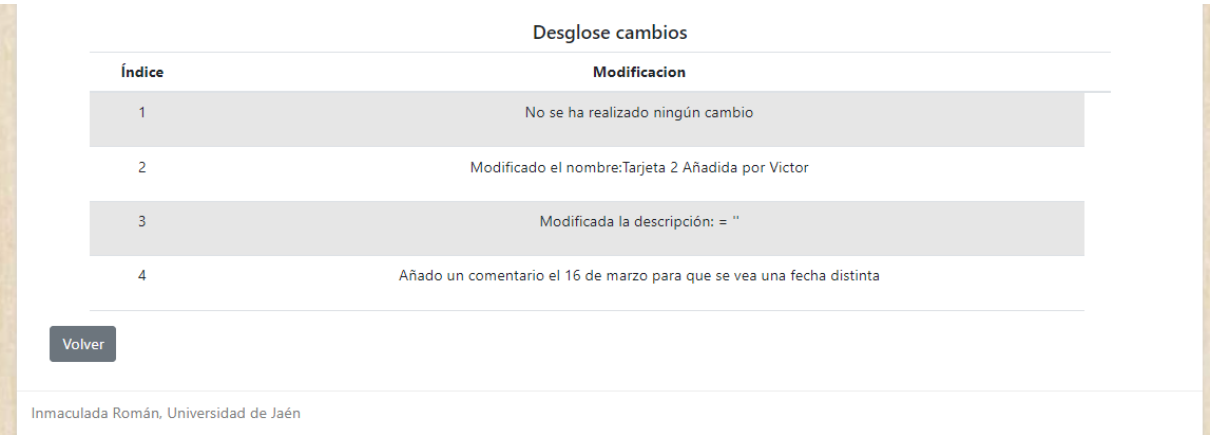

Ilustración 113: Diseño final para desglosar los cambios realizados en una lista

Búsqueda de acciones

Si el usuario desea realizar la búsqueda de alguna acción o acciones, sólo tiene que dirigirse a esta página en la cual podemos encontrar un desplegable, como podemos ver la ilustración 114 y el usuario automáticamente selecciona de que tipo o de que miembro quiere encontrar las acciones.

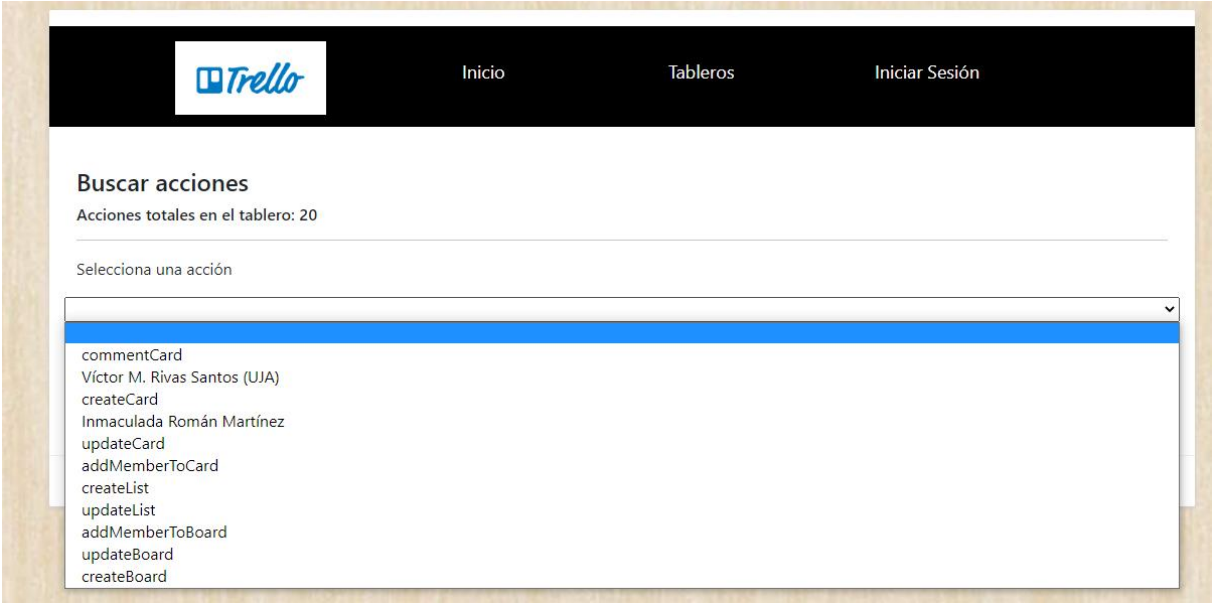

Ilustración 114: Diseño final para buscar acciones por tipo o miembro

7.3 Tercera iteración

En esta última iteración, sólo queda por terminar la historia de usuario para realizar la búsqueda de acciones. En la iteración anterior se desarrolló que el usuario pudiera realizar la búsqueda por tipo de acción o miembros que pertenecen al tablero. Ahora se va a realizar la búsqueda de acciones, pero sólo de aquellas que quedan comprendidas entre una fecha de inicio y una fecha de fin.

Para ello vamos a explicar la historia de usuario en sí, la comunicación existente entre las distintas partes mediante el diagrama de secuencia y por último vamos a detallar la implementación y las pruebas de validación realizadas.

7.3.1 Historias de usuario

Identificador	03
Título	Buscar acciones
Descripción	Como auditor de un tablero Trello quiero filtrar las acciones por distintos campos, como persona, tipo de actividad, fecha, etc. para conseguir más información sobre el flujo de trabajo.
Valor	M
Esfuerzo	XXL
Criterios de aceptación	-Realizar la búsqueda en un formulario mediante tipo de acción o miembro. -Seleccionar la fecha de inicio y de fin para realizar la búsqueda

Tabla 48: HU Buscar acciones por tipo, miembro y un rango de fechas

7.3.2 Diseño

Siguiendo los mismos pasos que en las iteraciones anteriores, se va a realizar el diseño correspondiente a la historia de usuario mediante un diagrama de secuencia para entender la comunicación existente.

HU03- Buscar acciones

Objetivo: Obtener las acciones que se han realizado entre un rango de fecha

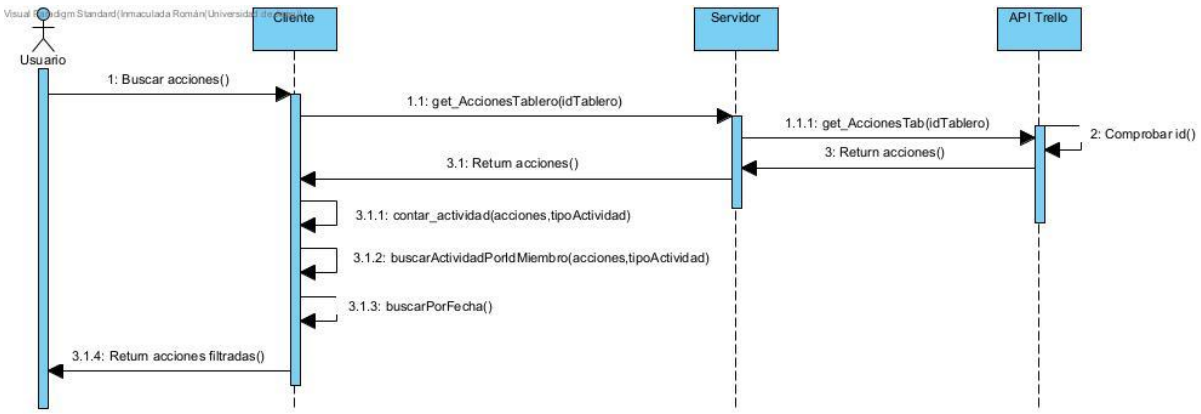

Ilustración 115: Buscar acciones por tipo, miembro y un rango de fechas

Para realizar la búsqueda de acciones necesitamos:

I. El usuario quiere obtener aquellas acciones que se encuentren entre la fecha de inicio y de fin indicadas.
II. El cliente solicita al servidor información sobre las acciones indicándole el identificador del tablero.
III. El servidor realiza la llamada a la Api, a través de la solicitud http **GET**: http://api.trello.com/1/boards/{idTablero}/actions
IV. La Api encapsula la información en formato JSON y se la devuelve al servidor.
V. El servidor manda los datos obtenidos en formato JSON al cliente.
VI. El cliente muestra al usuario un listado de las acciones que están en el rango, en formato HTML.

7.3.3 Implementación

A continuación, se va a detallar la implementación del cliente de la iteración final.

Implementación del cliente

Siguiendo con la carpeta “acción” es necesario añadir los ficheros necesarios para realizar la búsqueda de acciones comprendidas entre dos fechas. Por ello la estructura final de la carpeta quedaría como vemos en la ilustración 116.

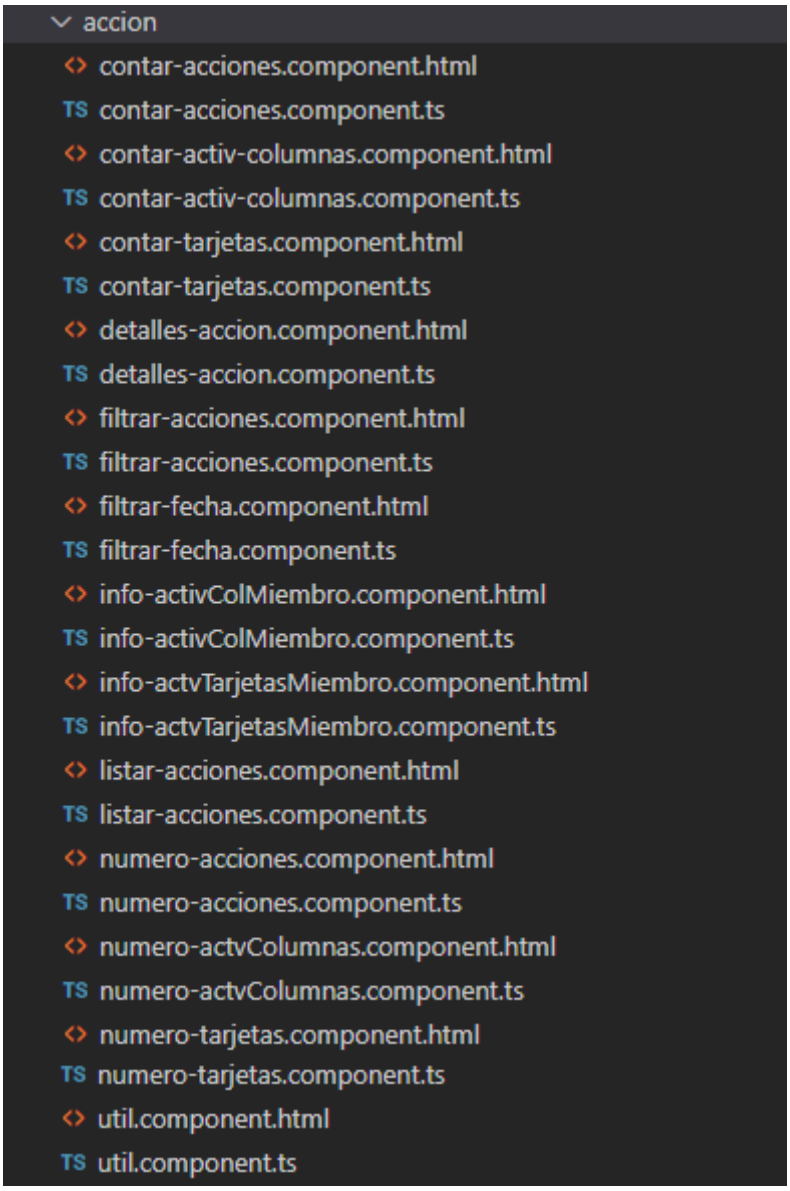

Ilustración 116: Ficheros del paquete acción del cliente

El fichero “útil.component.ts” contiene los métodos compartidos entre todos los archivos.

Para poder realizar la búsqueda de acciones comprendidas entre dos fechas, es necesario obtener todas las acciones y además necesitamos almacenar en dos variables distintas las fechas. Lo siguiente será buscar las acciones comprendidas entre esas dos fechas y mostrarlas al usuario.

En el siguiente código podemos ver como guardamos las fechas de inicio y fin, elegidas en los campos del formulario HTML. Es necesario ir guardando la fecha por día, año y mes para luego crear la fecha entera.

```
var anio = this.formFecha.controls['fecha_ini'].value.substr(0,4);
var mes = this.formFecha.controls['fecha_ini'].value.substr(5,2);
var dia = this.formFecha.controls['fecha_ini'].value.substr(8,2);
var dateIni = new  Date (anio,mes-1,dia);


anio = this.formFecha.controls['fecha_fin'].value.substr(0,4);
mes = this.formFecha.controls['fecha_fin'].value.substr(5,2);
dia = this.formFecha.controls['fecha_fin'].value.substr(8,2);
var dateFin = new  Date (anio,mes-1,dia);
```

Ahora solo queda buscar las acciones comprendidas entre ambas fechas y mostrarlas al usuario. Para realizar la búsqueda utilizamos el siguiente código.

```
this.fechas_filtradas = this.acciones.filter(e => {
  var es = new Date(e.date)
  return (dateIni <= es && dateFin >= es);
})
```

7.3.4 Pruebas de verificación y validaciones

Por último, vamos a comprobar que la búsqueda de acciones por fecha, funciona correctamente aplicando como anteriormente las pruebas de caja negra.

Prueba	Resultado	Valor
Buscar acciones		
Identificador válido	La aplicación redirige a mostrar la información correspondiente	Correcto
El usuario selecciona filtrar acciones por fecha	La aplicación muestra un formulario para rellenar la fecha de inicio y de fin para realizar la búsqueda	Correcto

Tabla 49: Pruebas de validación para buscar las acciones por fecha

7.3.5 Diseño final de la interfaz

Búsqueda de acciones

El diseño final para realizar la búsqueda de acciones comprendidas entre dos fechas, ha sido crear dos campos en los que el usuario pueda seleccionar ambas fechas como podemos observar en la ilustración 117.

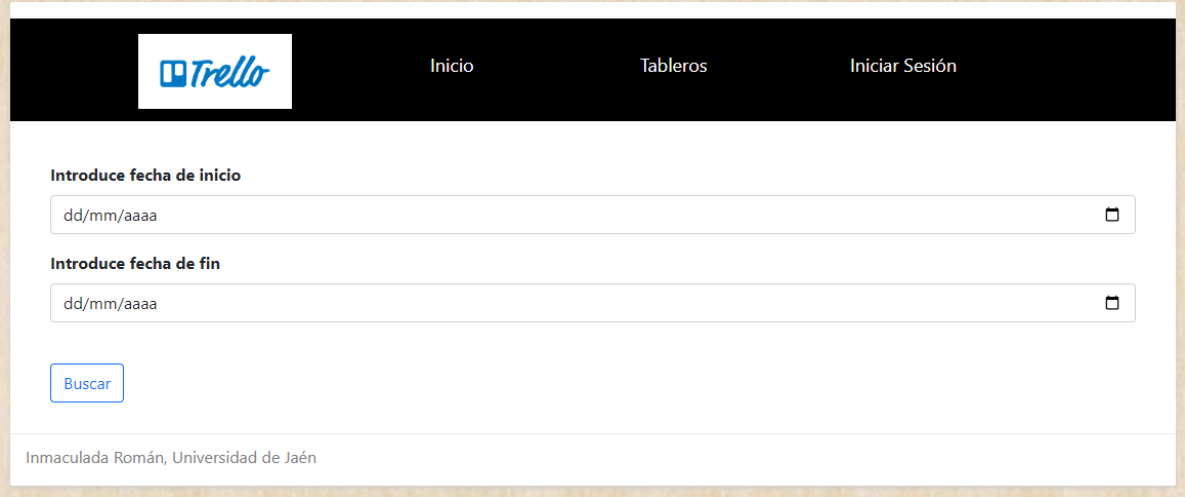

Ilustración 117: Diseño final para buscar acciones entre un rango de fechas

8. TRABAJO FUTURO

A continuación, voy a proponer una serie de ideas que se pueden tener en cuenta para añadir y mejorar este trabajo de fin de grado, entre ellas:

- Introducir gráficos para mejorar la visualización del trabajo de los estudiantes.
- Permitir que un usuario pueda registrarse con su nombre de usuario y contraseña que, junto con una base de datos en el servidor, permita ir almacenando las urls de los tableros que va visitando.

Bibliografía

[1] «Trello,» s.f. [En línea]. Available: https://trello.com/es/about. [Último acceso: Enero 2022].

[2] S. Ramos, «Socialgeek,» 2016. [En línea]. Available: https://socialgeek.co/startups/fog-creek-creadores-trello-otros-softwares/. [Último acceso: Enero 2022].

[3] «Glitch,» s.f. [En línea]. Available: https://glitch.com/. [Último acceso: Enero 2022].

[4] «Atlassian,» s.f. [En línea]. Available: https://www.atlassian.com/es/company. [Último acceso: Enero 2022].

[5] A. L. García Fernandez, «Apuntes de Desarrollo Ágil,» 2019. [En línea]. [Último acceso: Enero 2022].

[6] «Kanbanize,» s.f. [En línea]. Available: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban. [Último acceso: Enero 2022].

[7] «Trello,» s.f. [En línea]. Available: https://trello.com/guide/trello-101. [Último acceso: Enero 2022].

[8] Regina, «agilmania,» 10 Octubre 2018. [En línea]. Available: https://agilmania.com/que-es-jira/. [Último acceso: Enero 2022].

[9] Victor, «Walkiria Apps,» s.f. [En línea]. Available: https://walkiriaapps.com/tecnologia/jira-vs-trello-vs-asana/#:~:text=Jira%20se%20encarga%20de%20la,informes%20y%20gesti%C3%B3n%20de%20problemas.. [Último acceso: Enero 2022].

[10] «atlassian,» s.f. [En línea]. Available: https://www.atlassian.com/es/software/jira/pricing?gclsrc=aw.ds&&aceid=&adposition=&adgroup=109687534024&campaign=10332064740&creative=566167737815&device=c&keyword=jira%20software%20precio&matchtype=e&network=g&placement=&ds_kids=p55122862406&ds_e=GOOGLE. [Último acceso: Enero 2022].

[11] M. Duò, «Kinsta,» 31 Diciembre 2020. [En línea]. Available: https://kinsta.com/es/blog/trello-vs-asana/. [Último acceso: Enero 2022].

[12] «Asana,» s.f. [En línea]. Available: https://asana.com/es/pricing. [Último acceso: Enero 2022].

[13] «Kanbanflow,» s.f. [En línea]. Available: https://kanbanflow.com/. [Último acceso: Enero 2022].

[14] «Basecamp,» s.f. [En línea]. Available: https://basecamp.com/how-it-works. [Último acceso: Enero 2022].

[15] «Developer.Atlassian,» s.f. [En línea]. Available: https://developer.atlassian.com/cloud/trello/rest/. [Último acceso: Febrero 2022].

[16] E. Rodríguez, «seoestudios,» 27 Enero 2020. [En línea]. Available: https://www.seoestudios.es/blog/que-es-backend-web/. [Último acceso: Febrero 2022].

[17] R. Gerszenswit, «somospnt,» 03 enero 2020. [En línea]. Available: https://somospnt.com/blog/117-spring-webclient. [Último acceso: Febrero 2022].

[18] Edwise, «anotherdayanotherbug,» SOFTWARE-DEVELOPMENT, 05 05 2015. [En línea]. Available: https://anotherdayanotherbug.wordpress.com/2015/05/05/implementar-un-cliente-rest-con-spring-resttemplate/. [Último acceso: Febrero 2022].

[19] «ProjectReactor,» s.f. [En línea]. Available: https://projectreactor.io/. [Último acceso: Febrero 2022].

[20] «blog.auriboxtraining,» 16 Enero 2018. [En línea]. Available: https://blog.auriboxtraining.com/java/entorno-trabajo-sts/. [Último acceso: Febrero 2022].

[21] F. Redondo, «paradigmadigital,» 2017. [En línea]. Available: https://www.paradigmadigital.com/dev/postman-gestiona-construye-tus-apis-rapidamente/. [Último acceso: Febrero 2022].

[22] N. Maldeadora, «Platzi,» 2018. [En línea]. Available: https://platzi.com/blog/que-es-frontend-y-backend/. [Último acceso: Febrero 2022].

[23] «Angular,» s.f. [En línea]. Available: https://angular.io/guide/what-is-angular. [Último acceso: Febrero 2022].

[24] «Angular,» s.f. [En línea]. Available: https://angular.io/cli. [Último acceso: Febrero 2022].

[25] J. Lucas, «OpenWebinars,» 4 Septiembre 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-nodejs/. [Último acceso: Febrero 2022].

[26] R. Velasco, «softzone,» 26 Mayo 2021. [En línea]. Available: https://www.softzone.es/programas/utilidades/visual-studio-code/. [Último acceso: Febrero 2022].

[27] A. Pathak, «geekflare,» 25 Febrero 2021. [En línea]. Available: https://geekflare.com/es/gitlab-hosting/. [Último acceso: Febrero 2022].

[28] «rockcontent,» 12 Abril 2020. [En línea]. Available: https://rockcontent.com/es/blog/bootstrap/. [Último acceso: Marzo 2022].

[29] R. Content, «rockcontent,» 5 Junio 2019. [En línea]. Available: https://rockcontent.com/es/blog/que-es-java/. [Último acceso: Marzo 2022].

[30] C. Simões, «itdo,» 6 Julio 2021. [En línea]. Available: https://www.itdo.com/blog/que-es-typescript-y-por-que-utilizarlo/. [Último acceso: Marzo 2022].

[31] Digité, «Digité,» s.f. [En línea]. Available: https://www.digite.com/es/agile/historias-de-usuarios/. [Último acceso: Marzo 2022].

[32] «Jobted,» s.f. [En línea]. Available: https://www.jobted.es/salario/analista-programador. [Último acceso: Marzo 2022].

[33] M. Cillero, «manuel.cillero,» s.f. [En línea]. Available: https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-interaccion/diagrama-de-secuencia/. [Último acceso: Marzo 2022].

[34] J. A. Ramos, «Adictosaltrabajo,» 29 Octubre 2014. [En línea]. Available: https://www.adictosaltrabajo.com/2014/10/29/spring-boot/#:~:text=Spring%20Boot%20Starter%20POMs%20es,necesita%20el%20proyecto%20para%20funcionar.. [Último acceso: Abril 2022].

[35] «ebooksonline.es,» 23 Marzo 2021. [En línea]. Available: https://ebooksonline.es/que-es-una-prueba-de-caja-negra-tecnicas-muestras-y-tipos/#:~:text=Prueba%20de%20caja%20negra%20es,implementaci%C3%B3n%20y%20las%20rutas%20internas..

Anexos

En este apartado, vamos a detallar el funcionamiento de la aplicación para que todo usuario pueda hacer uso de ella y a la vez sea fácil e intuitiva. También se va a detallar el manual de instalación, mostrando cómo cualquier usuario puede descargarse ambos códigos para hacer uso de ellos.

Anexo 1: Manual de Instalación

En este anexo vamos a explicar cómo instalar la aplicación para que pueda ser utilizada en cualquier ordenador. Para ello, vamos a dividir la instalación en dos partes, por un lado, la instalación del servidor, por otro, la instalación del cliente.

Instalación del servidor

Para instalar el servidor, lo primero que hay que hacer es descargar el software Spring Tools desde el navegador (ilustración 118) y descargar según el sistema operativo.

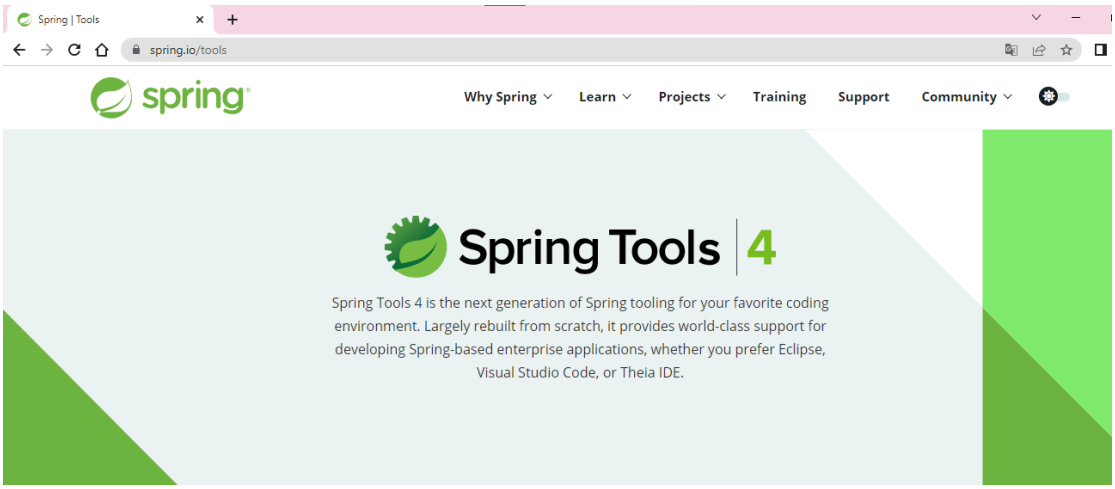

Ilustración 118: Página para descargar la herramienta Spring Tools Suite

Cuando la descarga haya finalizado descomprimimos la carpeta y ejecutamos la aplicación de SpringToolsSuite4, como vemos en la ilustración 119.

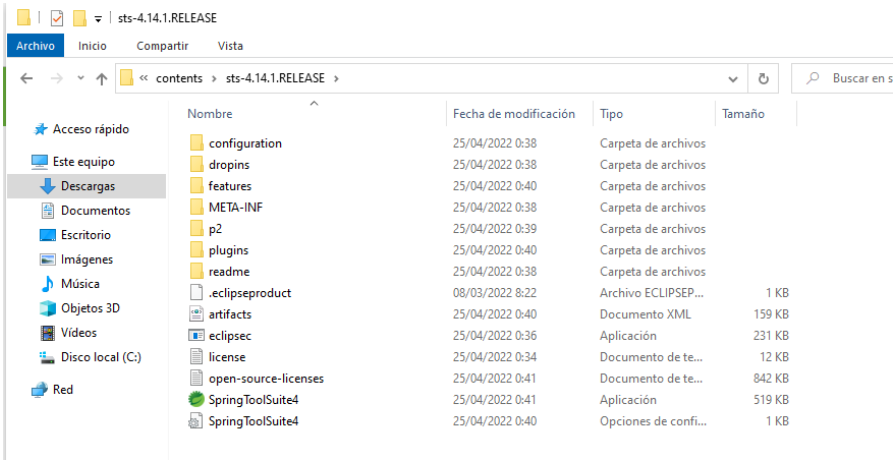

Ilustración 119: Carpeta de la herramienta para ejecutar la aplicación

Cuando la aplicación está arrancando, elegimos el espacio de trabajo, que será donde se guarden los proyectos. Y automáticamente el programa estará instalado y se abrirá en nuestro ordenador. En la ilustración 120, es donde elegimos el espacio de trabajo.

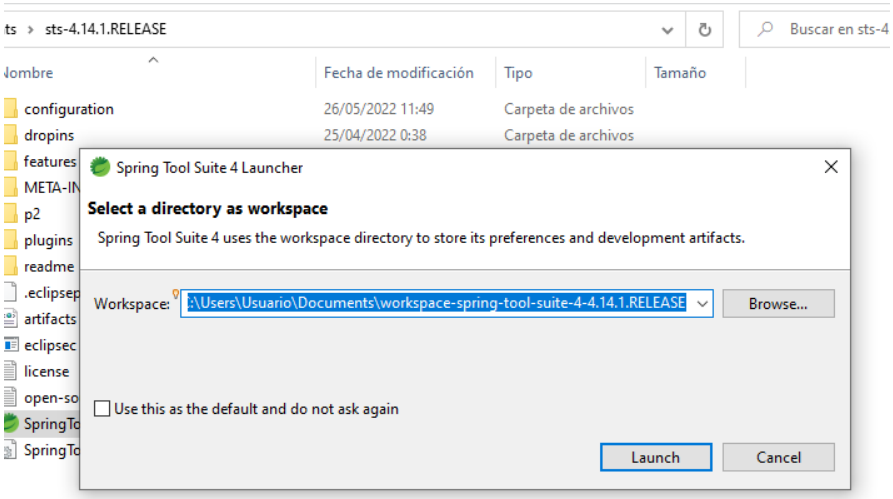

Ilustración 120: Elegir espacio de trabajo de Spring Tools Suite

El siguiente paso es importar el código (ilustración 121), para ello entramos en la aplicación SpringToolSuite y seleccionamos en la pestaña “Git Repositories” → “Clone Git Repository”.

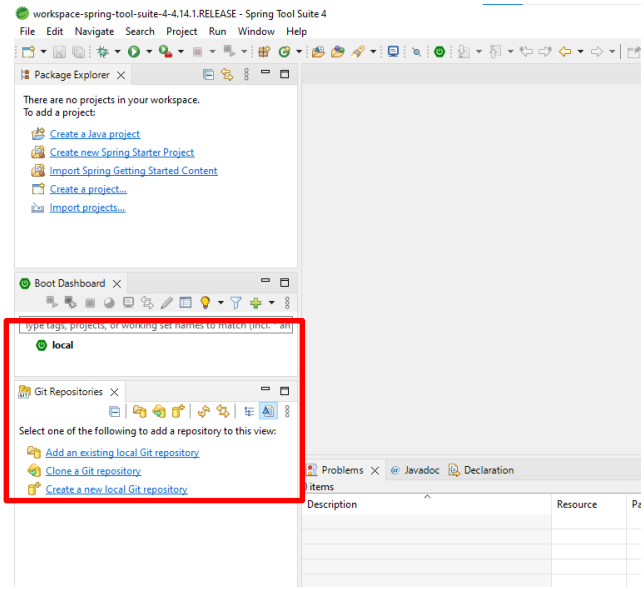

Ilustración 121: Ilustración para clonar el código del servidor

Para realizar la importación (ilustración 122) tenemos que dirigirnos a la siguiente url → https://gitlab.com/irm00040/tfg_backend_webclient de gitlab, que es donde se encuentra subido el código necesario para arrancar el servidor. Para ello, nos dirigimos a “Clone” y copiamos la dirección para clonar con HTTPS.

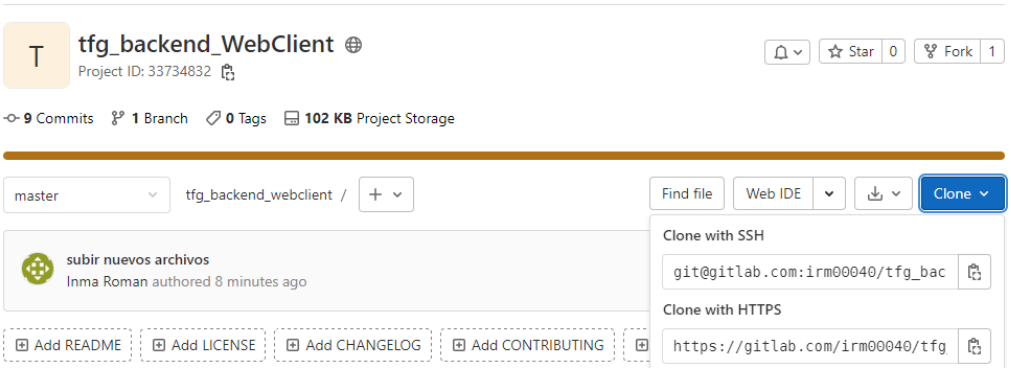

Ilustración 122: Ilustración para copiar el enlace de gitlab

Una vez copiada la dirección https, volvemos al programa SpringTools y dentro de "Clone Git Repository" copiamos la dirección. Esto lo podemos ver en la ilustración 123.

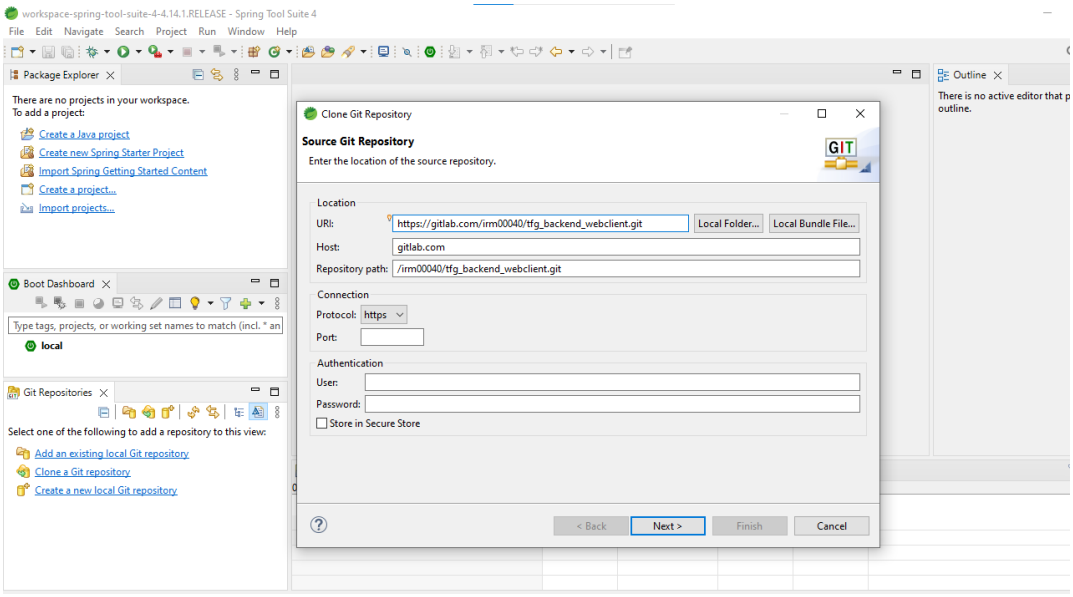

Ilustración 123: Ilustración para clonar la url en SpringTools

Continuamos los pasos del manual de importación y cuando este haya finalizado, solo tendremos que dirigirnos a la pestaña "File" → "Open Proyects From File System" seleccionamos "Directory" y buscamos la carpeta "git" donde se ha importado el proyecto.

En este caso se encuentra en "C:\Usuario\Usuarios\git\tfg_backend_webclient". Estos pasos se pueden ver en las ilustraciones 124 y 125.

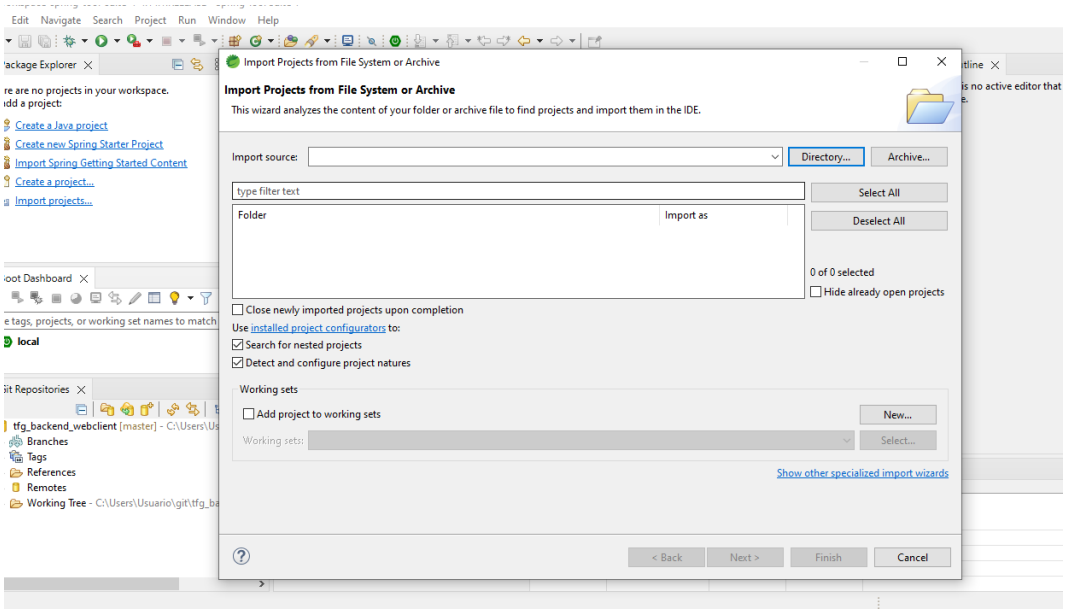

Ilustración 124: Importar proyecto Git en SpringTools

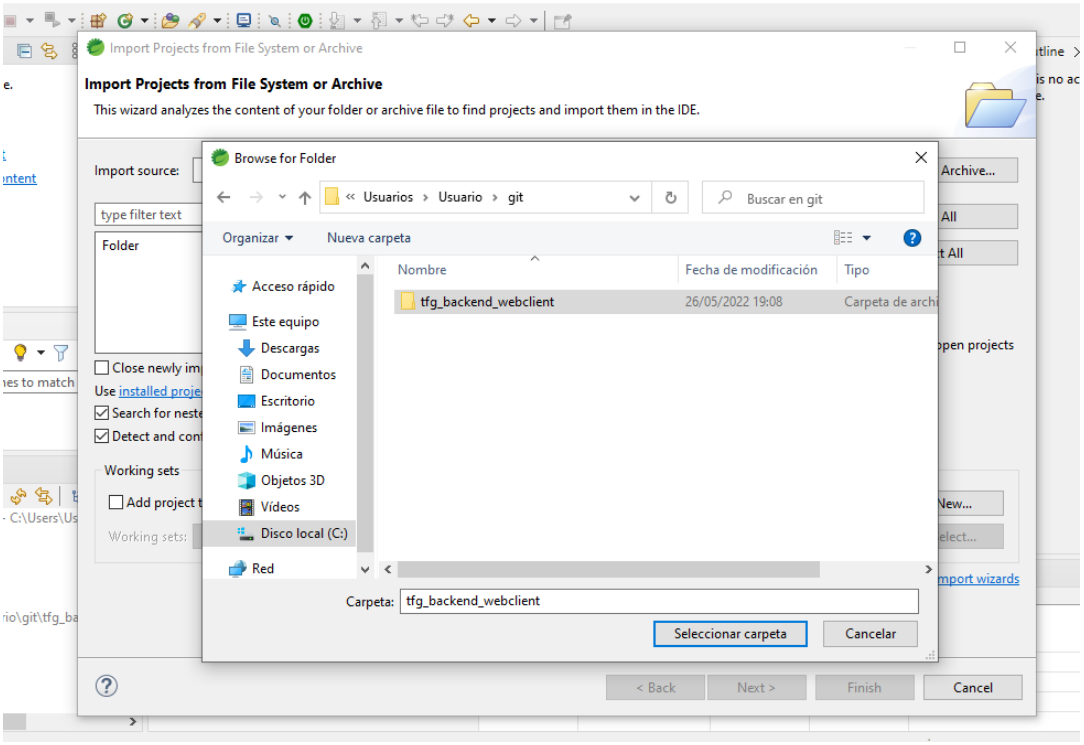

Ilustración 125: Seleccionar proyecto Git para terminar la importación en SpringTools

Una vez terminada la importación, sólo nos queda ejecutar el proyecto. Para ello pulsamos sobre el proyecto, botón derecho y seleccionamos “Run As” → “Spring Boot App” y arrancará el servidor. En la consola nos aparecerá un identificador de tablero como vemos en la ilustración 126, esto nos indica que el proyecto ha arrancado con éxito.

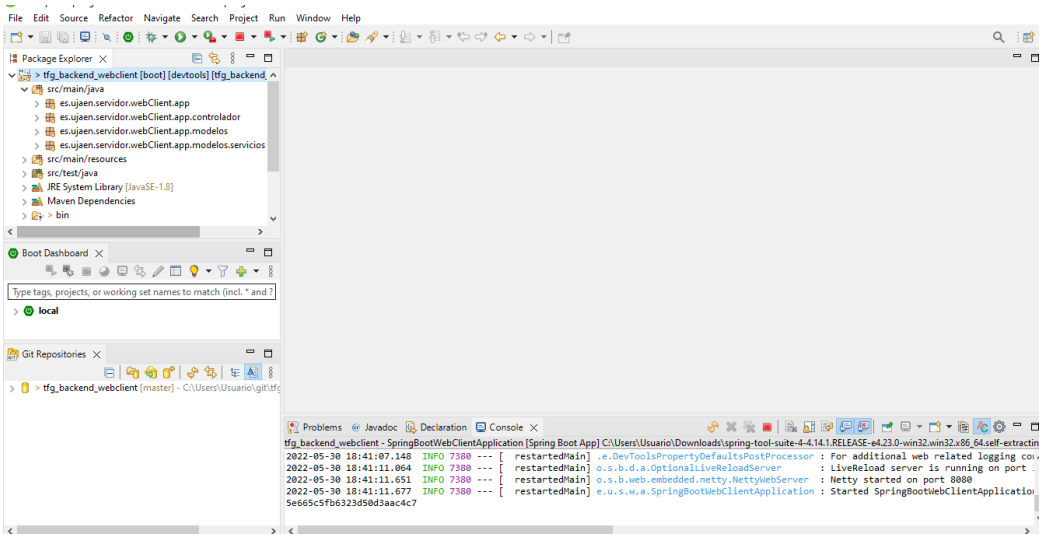

Ilustración 126: Proyecto Git importado y arrancado

Anotación: Es necesario tener instalado previamente JDK 8.

Instalación del cliente

Para la instalación del cliente será necesario descargar la herramienta “Visual Studio Code” dependiendo del sistema operativo desde la página oficial, como vemos en la ilustración 127.

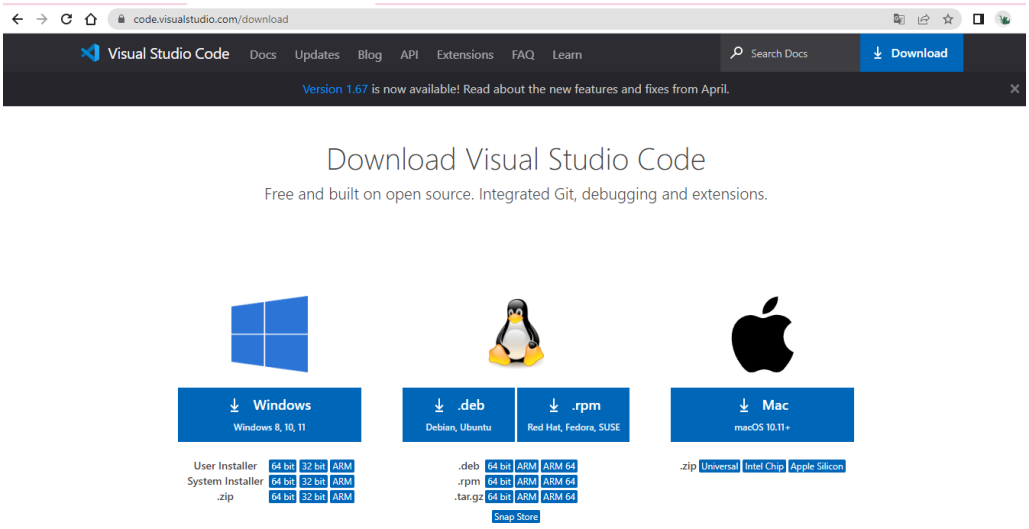

Ilustración 127: Ilustración para descargar Visual Studio Code

Al ejecutar la aplicación, solo hay que seguir los pasos de instalación y automáticamente se instalará la aplicación y automáticamente se abrirá en el ordenador. (ilustración 128).

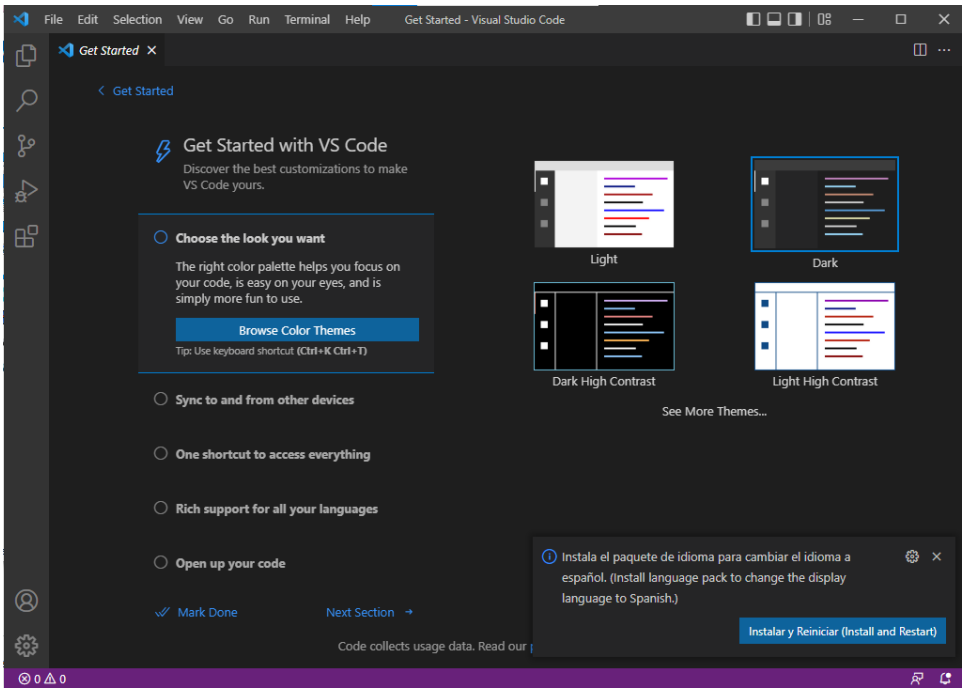

Ilustración 128: Instalación Visual Studio Code finalizada

Para poder hacer uso del código, es necesario realizar la instalación de node.js y se instalará también npm. Escribimos en el navegador https://nodejs.org/es/download/ y descargamos el instalador dependiendo del sistema operativo (ilustración 129).

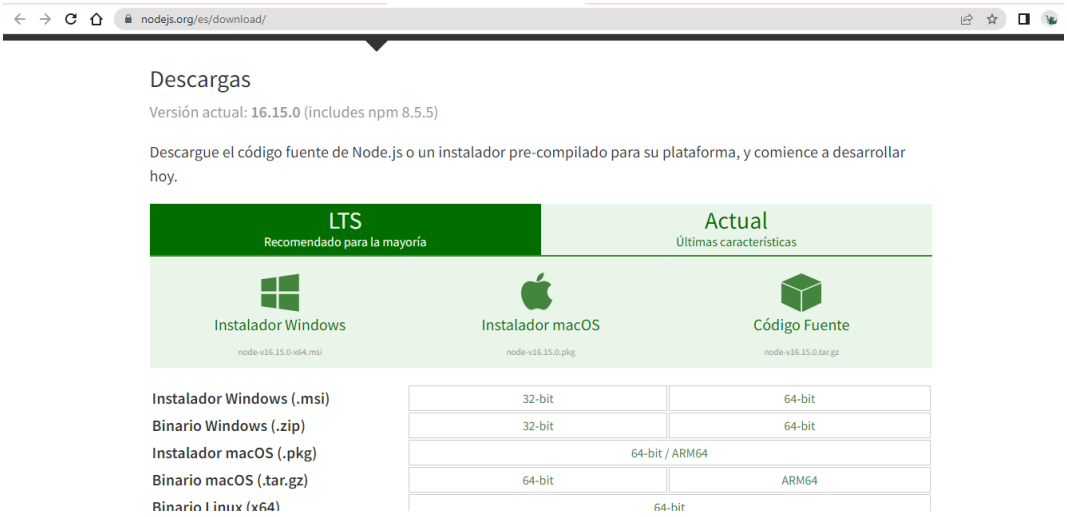

Ilustración 129: Ilustración para descargar node.js

Iniciamos la instalación siguiendo los pasos, y una vez instalado vamos a comprobar que se ha instalado correctamente. Es necesario abrir el Símbolo del sistema (cmd) y ejecutar los comandos que vemos en la ilustración 130. Esto nos indicará la versión que se ha descargado e instalado tanto de node.js como de npm.

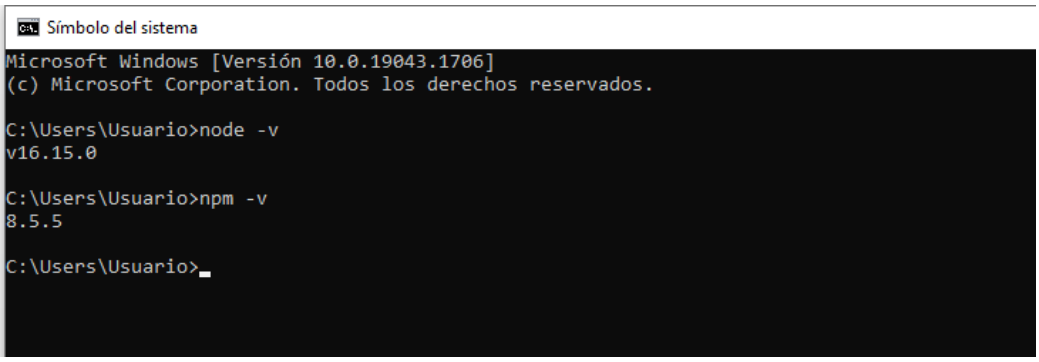

Ilustración 130: Ilustración para comprobar la instalación de node.js

Para terminar con la configuración de node.js, nos dirigimos a "Equipo" → "Propiedades" → "Configuración Avanzada del sistema" → "Variables de entorno" y editamos el "path" de las variables de usuario. Añadiendo así, la ruta donde se ha instalado node.js, que suele ser en la carpeta de "Program Files". Ver ilustración 131.

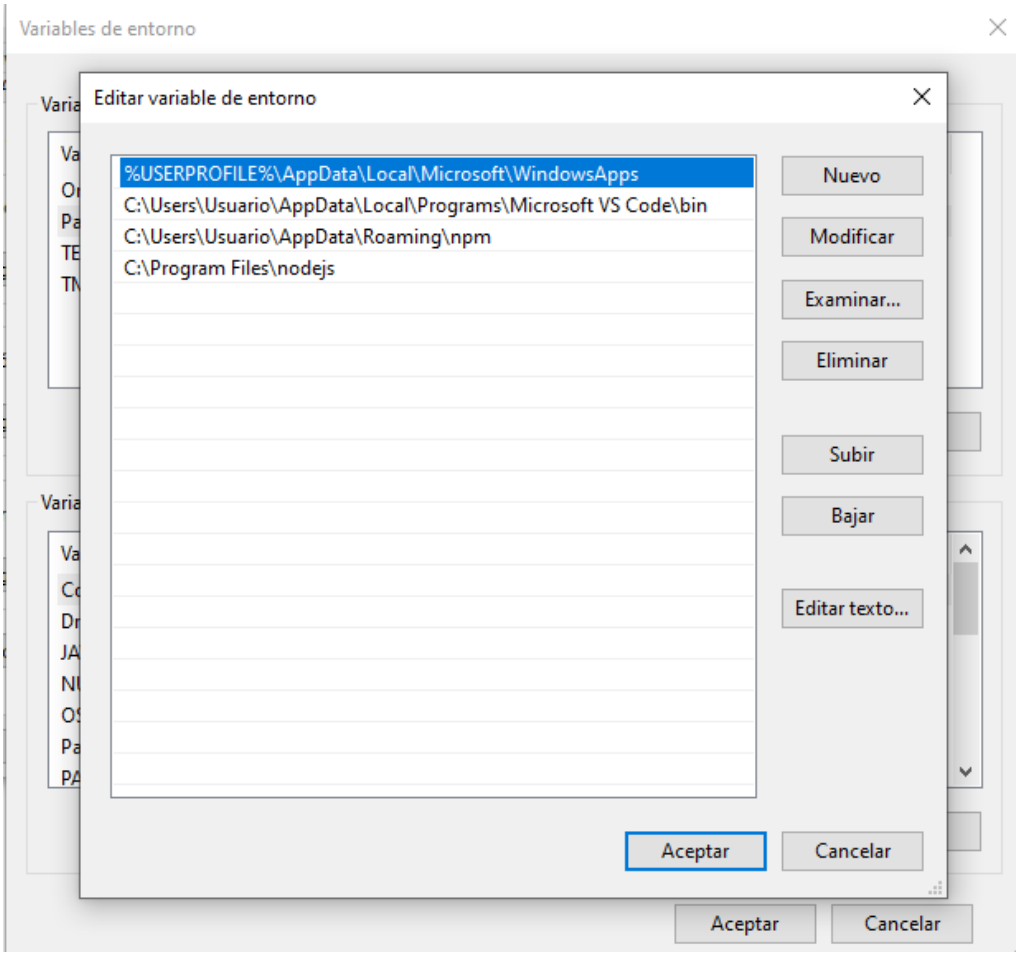

Ilustración 131: Ilustración para añadir el path de node.js a las variables de entorno

Ahora procedemos a descargar el código correspondiente al cliente, el cual se encuentra en gitlab, en la siguiente url → https://gitlab.com/irm00040/tfg_frontend y descargamos el proyecto en formato zip, para ello ver ilustración 132.

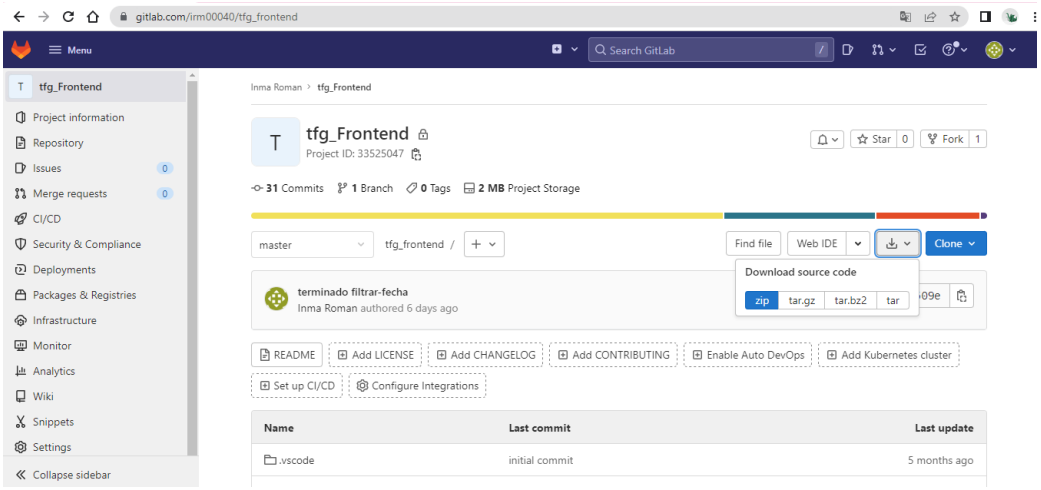

Ilustración 132: Como descargar el código del cliente

Para abrir el proyecto (ilustración 133), abrimos Visual Studio Code y vamos a "Archivo" → "Abrir carpeta" y seleccionamos la carpeta previamente descargada y descomprimida.

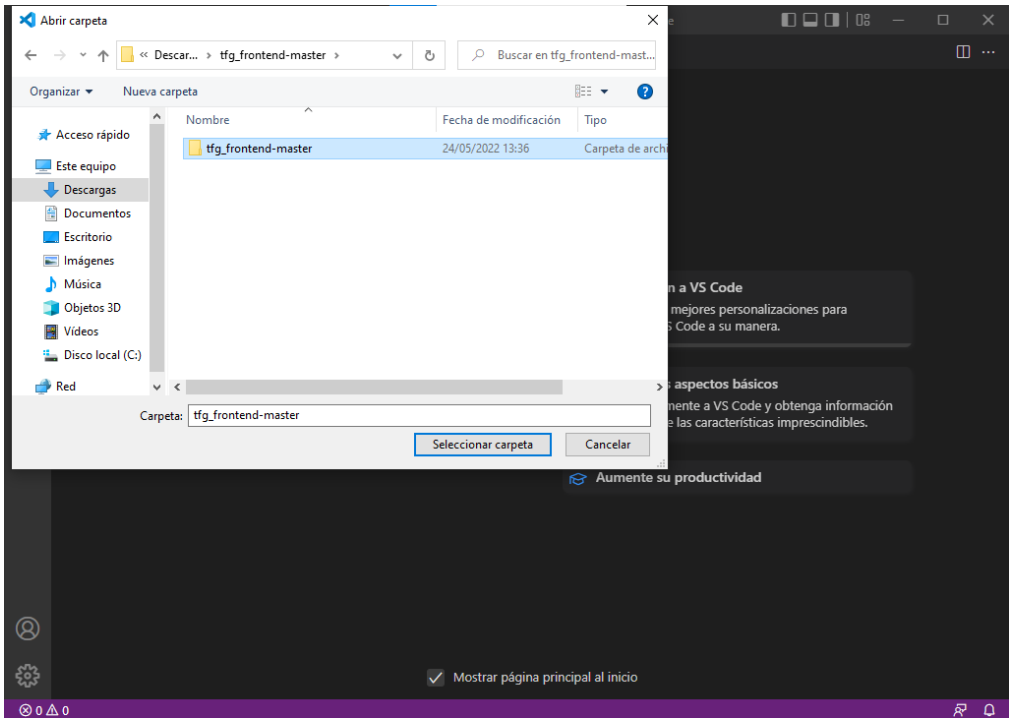

Ilustración 133: Abrir el proyecto en Visual Studio Code

Automáticamente se abrirá la carpeta, y solo nos quedaría realizar la instalación de Angular Cli, para ello, nos vamos a “Terminal” → “Nuevo terminal” y ejecutamos el siguiente comando → “npm install –g @angular/cli”, para que quede más claro se puede ver en la ilustración 134.

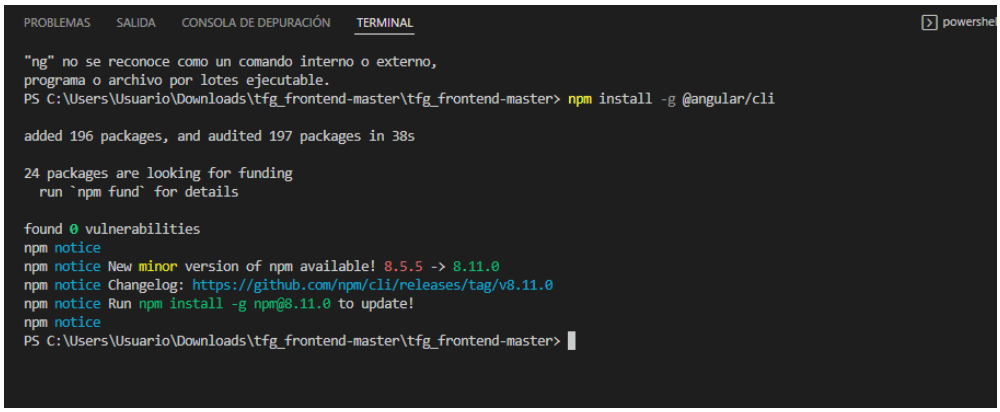

Ilustración 134: Instalar angular/cli

El último paso, sería ejecutar el comando → “npm start” para ejecutar la aplicación (ilustración 135).

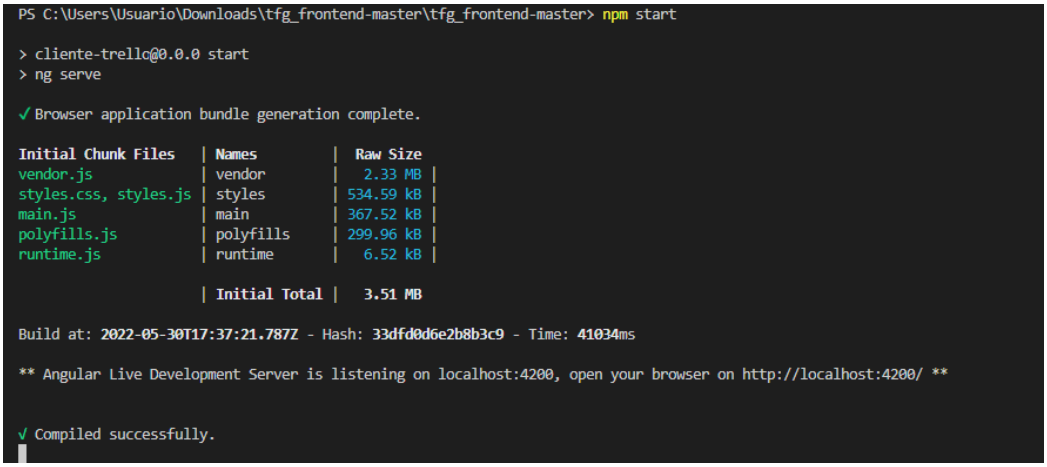

Ilustración 135: Ejecutar el cliente

Como vemos en la ilustración 135, si se ha compilado con éxito, y previamente hemos arrancado el servidor, solo nos queda abrir el navegador y escribir → “localhost:4200” y veremos la página principal de la aplicación.

Ilustración 136: Página de inicio

Anexo 2: Manual de Usuario

El manual de usuario describe el funcionamiento de las diferentes pantallas de la aplicación. Lo primero que se tiene que hacer es, una vez instalado el servidor y el cliente y arrancado la aplicación, tendremos que entrar en el navegador, en este caso se ha utilizado Google Chrome y escribir la siguiente dirección → "http://localhost:4200".

Una vez tecleada la dirección, automáticamente aparecerá la siguiente pantalla (ilustración 137).

Ilustración 137: Pantalla de inicio

En esta pantalla aparece una breve descripción del funcionamiento de la aplicación. En la parte superior aparece un menú principal con las siguientes opciones:

1. **Inicio.** Muestra la página de inicio de la aplicación. (Ilustración 137)
2. **Tableros.** Dirige a la búsqueda de un tablero (ver ilustración 138) en el cual es necesario introducir un identificador.

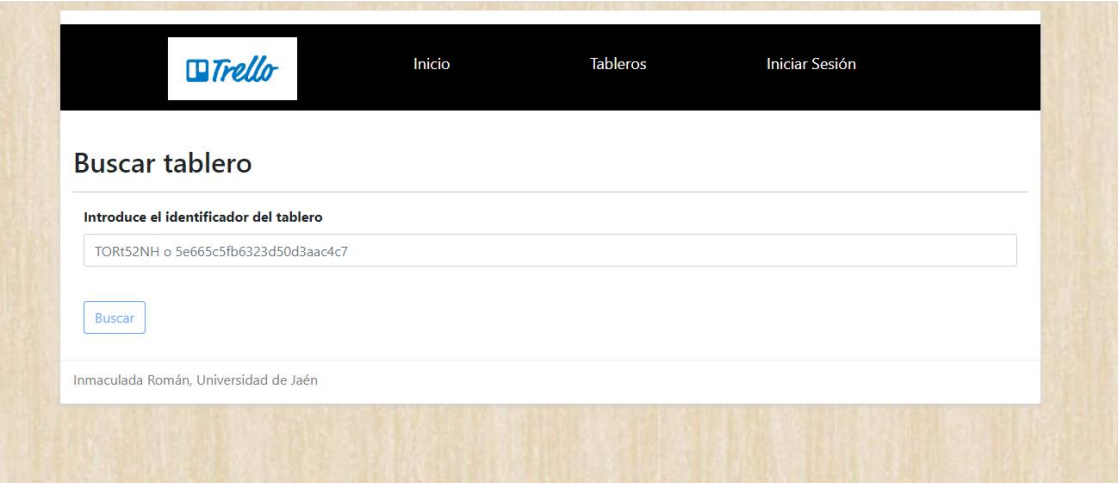

Ilustración 138: Pantalla para realizar la búsqueda de un tablero

Tenemos dos opciones de identificadores para realizar la búsqueda:

- **Opción 1.** Situados en el tablero a buscar en la página de Trello, nos fijamos en la url, y aparece un identificador (ver ilustración 139). Este corresponde al nombre del fichero JSON del tablero. Con este identificador se puede realizar la búsqueda.

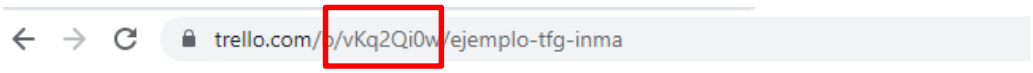

Ilustración 139: Identificador del tablero mediante la url

- **Opción 2.** Al exportar el tablero en formato JSON desde Trello, al inicio del documento aparece el identificador correspondiente al tablero, con el cual se puede realizar la búsqueda. Se puede ver en la ilustración 140.

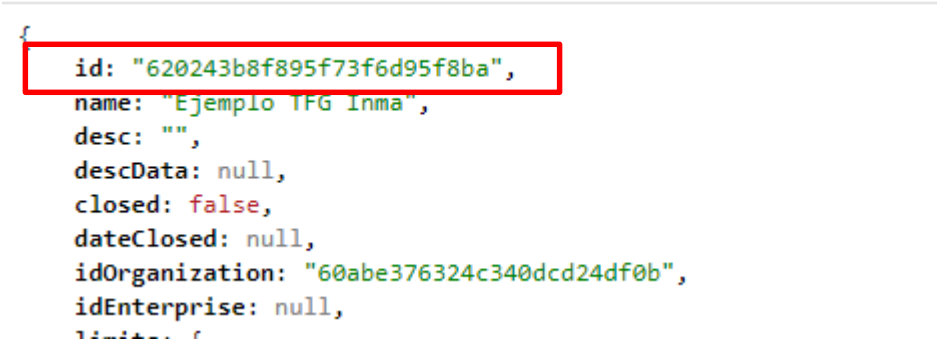

Ilustración 140: Identificador del tablero en el fichero JSON

Una vez introducido el identificador y realizada la búsqueda, nos aparece en la pantalla la información del tablero (ilustración 141).

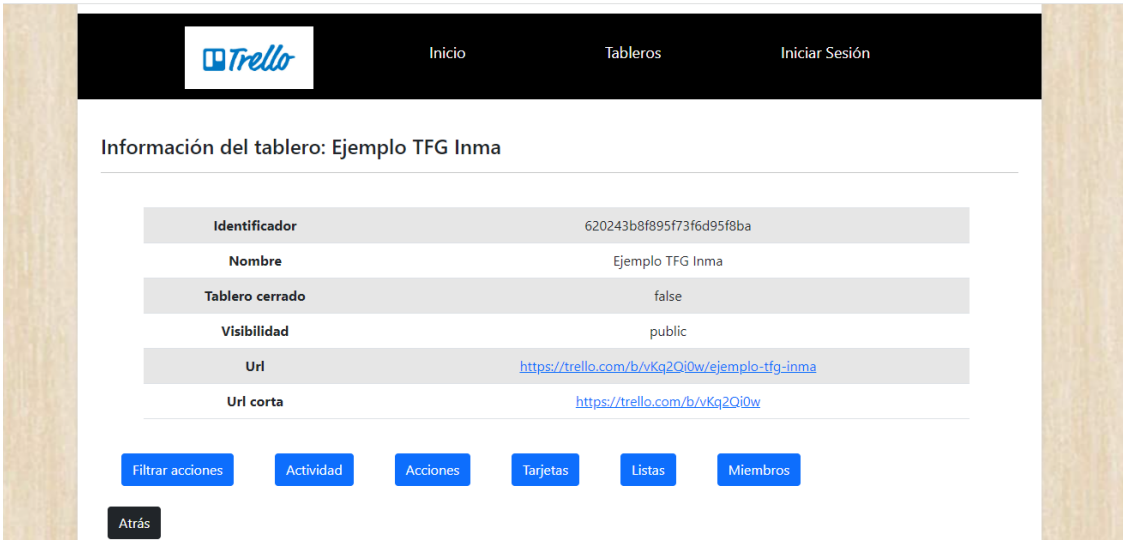

Ilustración 141: Pantalla para mostrar la información de un tablero

Como podemos observar en la ilustración 141, además de ver la información del tablero, tenemos varios botones con diferentes funcionalidades, entre ellas:

Filtrar acciones

Al pulsar sobre “Filtrar acciones” se muestra una pantalla en la cual aparece un desplegable para realizar una búsqueda dependiendo del tipo o del miembro que realiza la acción (Ilustración 142).

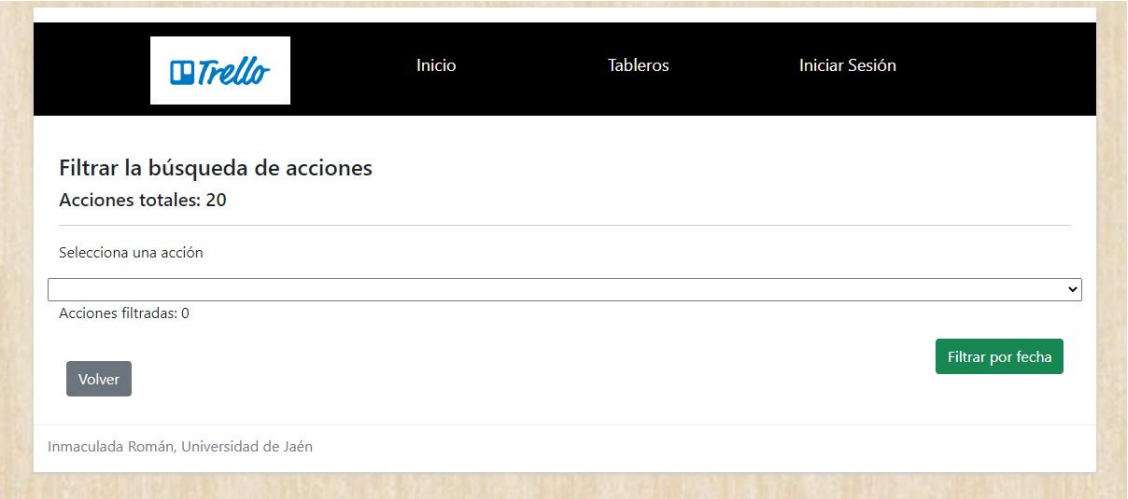

Ilustración 142: Pantalla para buscar acciones por tipo o miembro

Al realizar la búsqueda, automáticamente aparece la siguiente pantalla. (Ilustración 143).

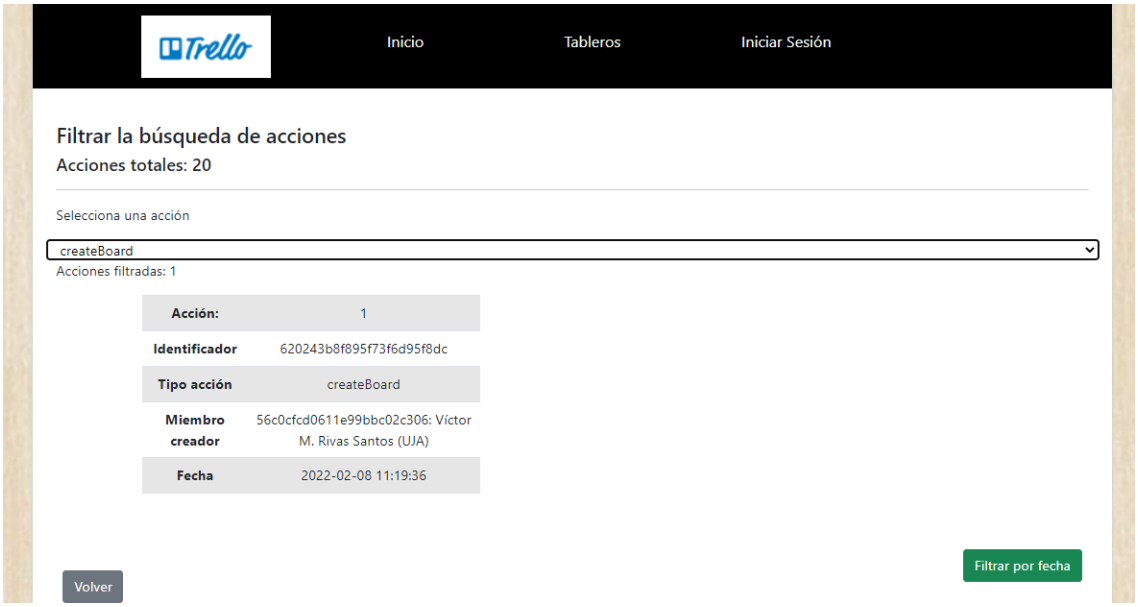

Ilustración 143: Pantalla mostrando el resultado de la búsqueda

Si nos fijamos bien en la ilustración 143, aparece un botón, el cual nos va a permitir realizar una búsqueda, pero esta vez por fechas. (Ilustración 144)

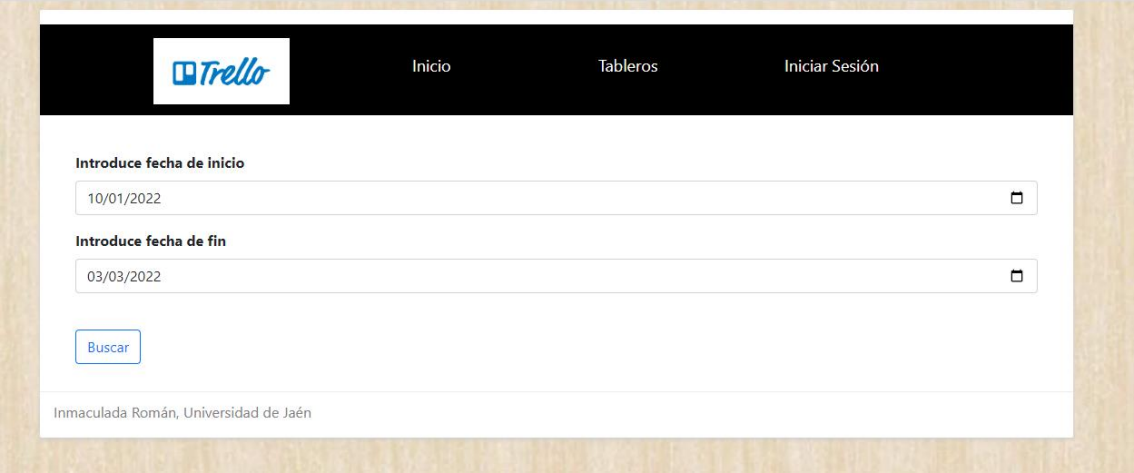

Ilustración 144: Pantalla para buscar acciones por rango de fechas

Al introducir el rango de fechas a buscar, automáticamente aparecen las acciones que se encuentran entre ese rango. (Ilustración 145).

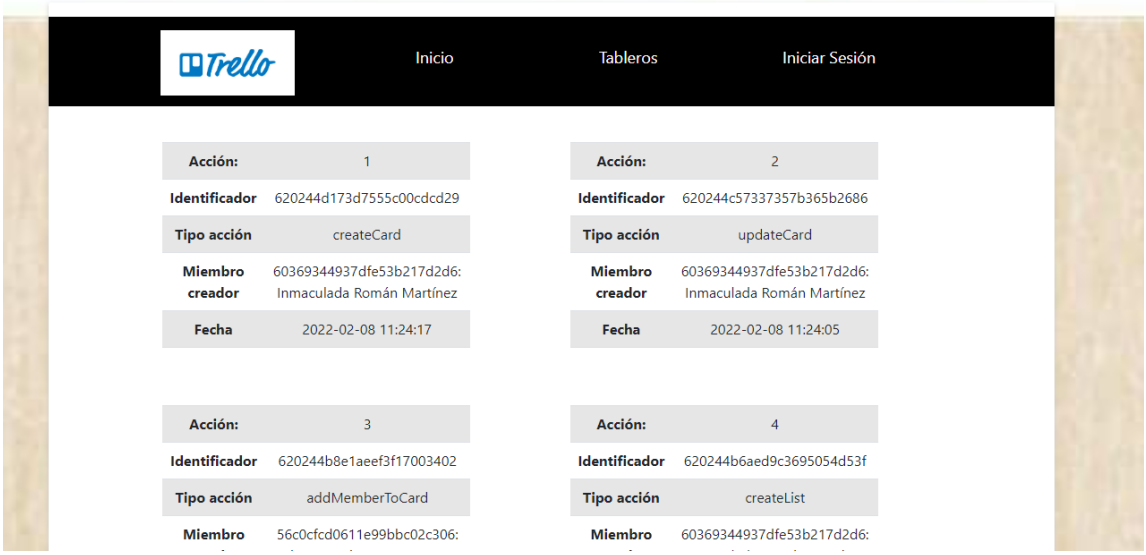

Ilustración 145: Pantalla mostrando las acciones buscadas entre dos fechas

En caso de introducir dos fechas y no haber realizado ninguna acción, se muestra una pantalla con un mensaje indicando que no hay acciones entre ese rango. (Ilustración 146).

Ilustración 146: Pantalla si no hay acciones entre el rango de fechas buscadas

Actividad

Este apartado tiene la funcionalidad principal de la aplicación. Para ver la actividad, el usuario tiene que volver a la pantalla donde se muestra la información del tablero (ilustración 141) y pulsar sobre el botón “Actividad”. Entonces se muestra una pantalla en la que se van a clasificar las acciones realizadas (Ilustración 147).

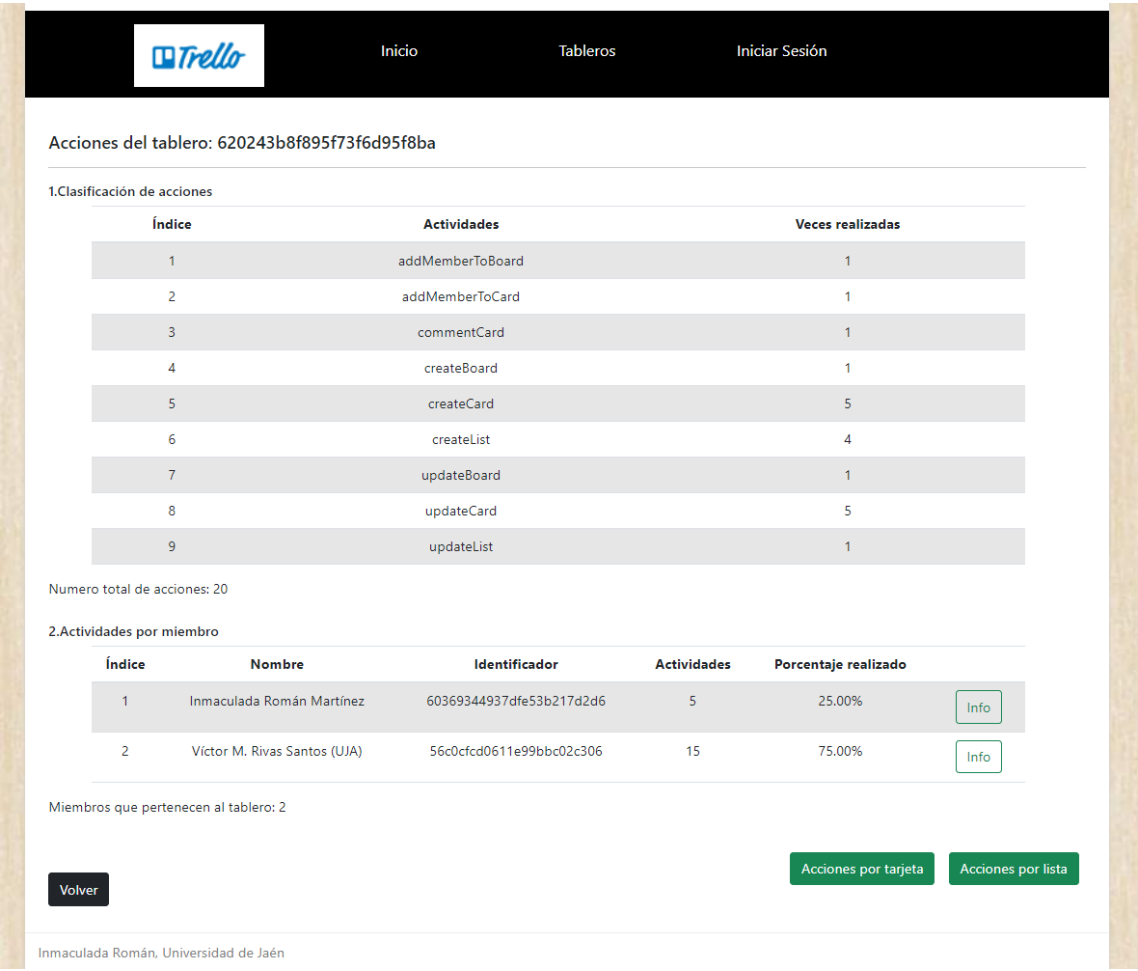

Ilustración 147: Pantalla para clasificar y cuantificar las acciones

En la parte superior, aparece una tabla que contiene el tipo de acciones y cuantas se han realizado de cada una. En la parte inferior, se encuentra otra tabla en la que se cuenta la actividad que ha realizado cada miembro del tablero indicando el nombre e identificador de cada miembro, junto con el número y porcentaje realizado. (Ilustración 147).

Para ver información más detallada sobre la actividad realizada por cada miembro, solo hay que pulsar sobre el botón “info” de la tabla “Actividades por miembro” y se mostrará la siguiente pantalla (ilustración 148).

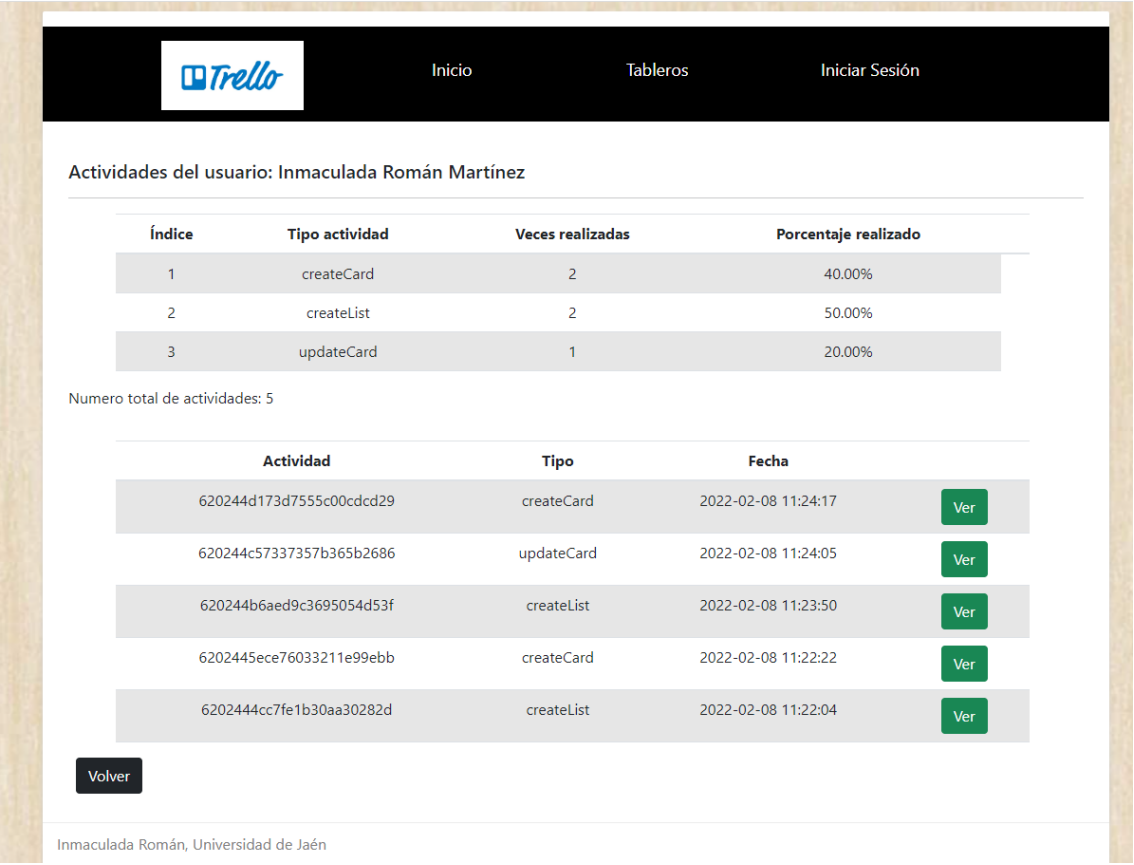

Ilustración 148: Pantalla que muestra la actividad de un miembro

En esta pantalla se desglosa la actividad realizada por un miembro, indicando en una primera tabla el tipo de acción que ha realizado, junto con el número de veces que ha realizado cada una y el porcentaje de actividad realizado sobre cada acción.

En la parte inferior, se encuentra otra tabla con la información de cada acción junto con la fecha de realización.

Si clicamos en “Ver”, se puede ver información más detallada de una acción (Ilustración 149).

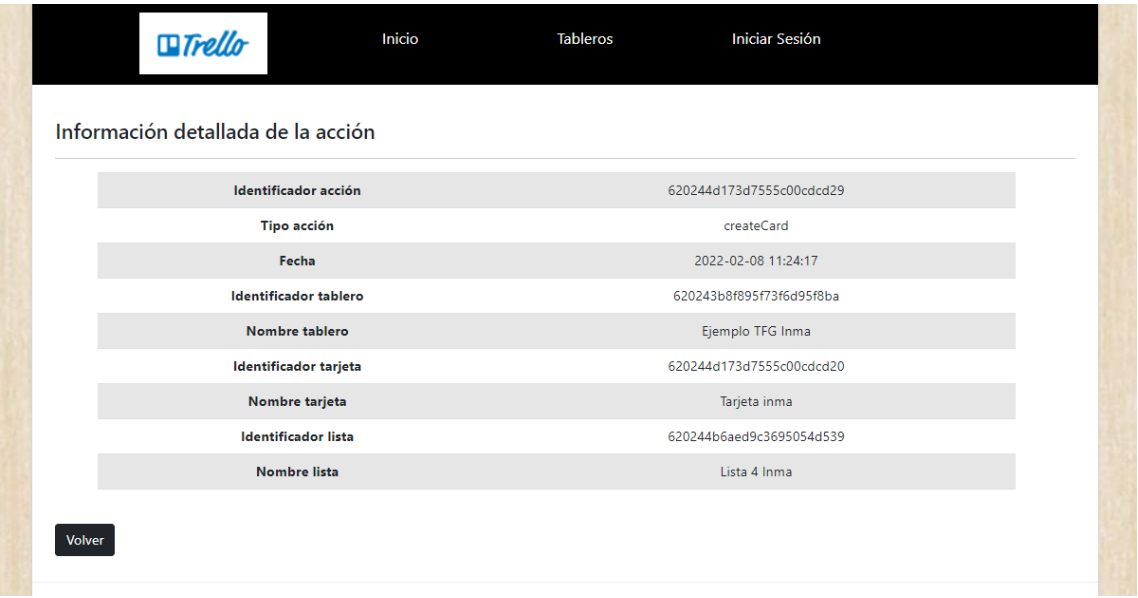

Ilustración 149: Pantalla para mostrar la información de una acción

Volviendo a la pantalla en la que se clasifican y cuantifican las acciones (Ilustración 147), aparecen dos botones que vamos a desglosar.

Acciones por tarjeta

En esta pantalla encontramos la actividad realizada sobre las tarjetas del tablero, como vemos en la ilustración 150.

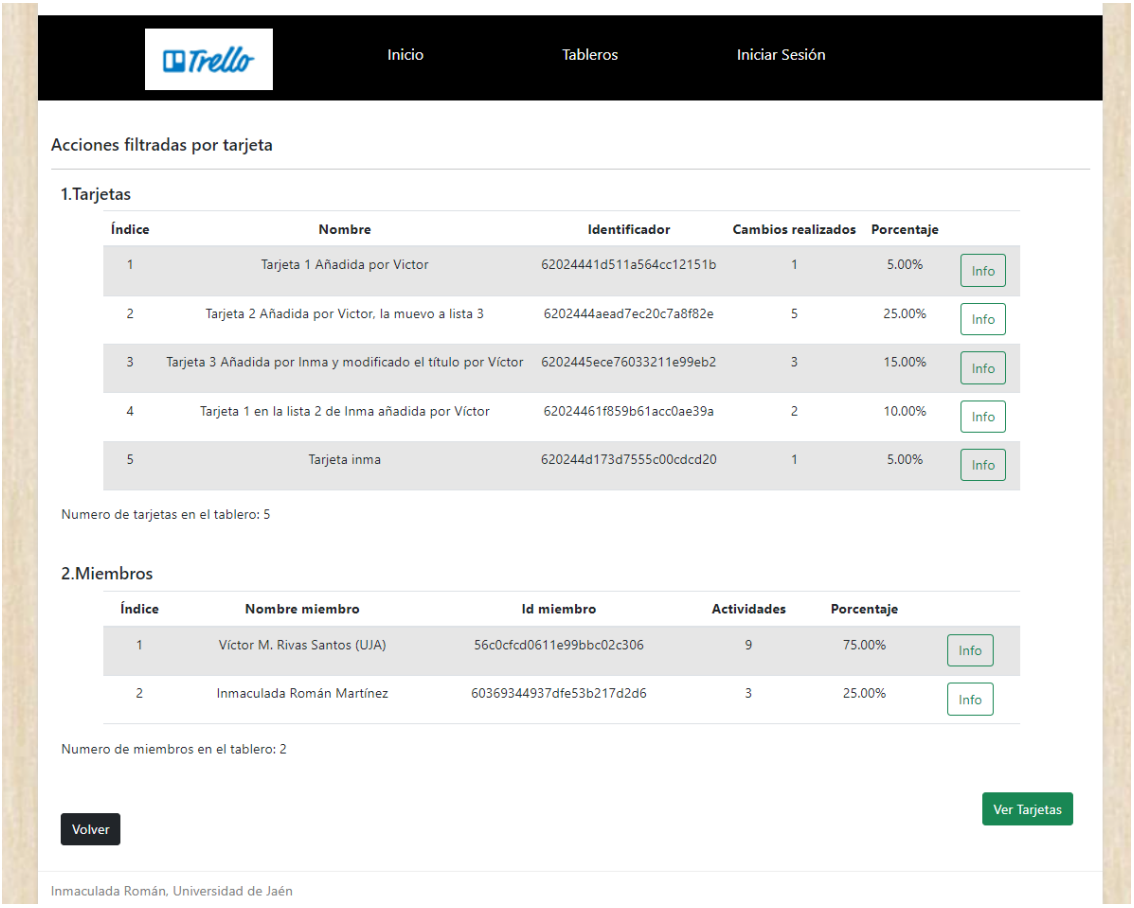

Ilustración 150: Pantalla para mostrar las acciones realizadas en tarjetas

En la primera tabla aparecen las tarjetas que han sido creadas en el tablero, y junto a ellas, aparece el total de acciones que se han realizado en cada una. Esto hace referencia a los cambios realizados y también aparece el porcentaje de actividad.

En la tabla de abajo, encontramos las acciones que ha realizado cada miembro, pero esta vez sobre las tarjetas. La información es igual que se ha explicado anteriormente.

Si queremos ver los cambios por los que ha pasado una tarjeta, simplemente pulsando sobre el botón “Info” de la primera tabla nos lleva a la pantalla donde se desglosan los cambios (Ilustración 151).

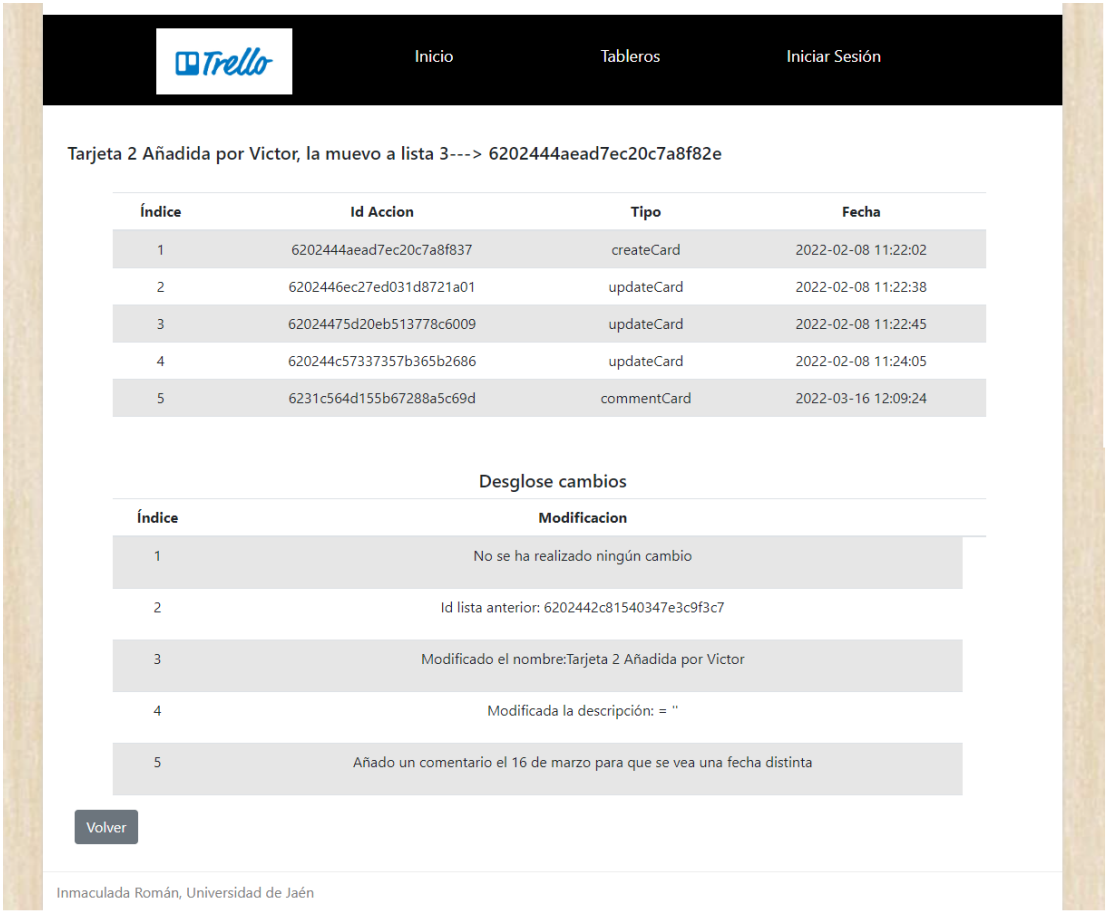

Ilustración 151: Pantalla que muestra los cambios de una tarjeta

En esta pantalla (ilustración 151), encontramos por las diferentes fases que ha pasado una tarjeta indicando el identificador, el tipo y la fecha en la que se realizó la acción. Como vemos viene indicado con un índice haciendo referencia, el número uno a lo primero que se hizo y el número 5 la última acción que se realizó. Vamos a desglosar los cambios para que quede más claro (tabla de abajo).

Volviendo a hacer hincapié al índice, el número 1 indica que se creó la tarjeta, por tanto, en la tabla de abajo como bien indica, no se ha realizado ningún cambio. El número 2 indica que se actualizó la tarjeta, en la tabla de abajo vemos que pone “Id de la lista anterior”, esto indica que la tarjeta se movió de lista. El número 3 vuelve a indicar que se actualizó la tarjeta, pero esta vez muestra el nombre anterior que tenía la tarjeta. El número 4 indica que la descripción de la tarjeta estaba en blanco, y por tanto se añadió una descripción. Y, por último, el número 5 muestra que se añadió un comentario a la tarjeta.

Acciones por lista

Volviendo a la pantalla en la que se clasifican y cuantifican las acciones (Ilustración 147), aparecen el botón “Acciones por lista”. La información que muestra es igual que lo comentado en el punto anterior (Acciones por tarjeta).

En la ilustración 152, aparece en una primera tabla las listas creadas en el tablero junto con el número de acciones que se han realizado en cada una y el porcentaje de actividad.

En la segunda tabla se vuelve a contabilizar la actividad realizada por cada miembro sobre las listas.

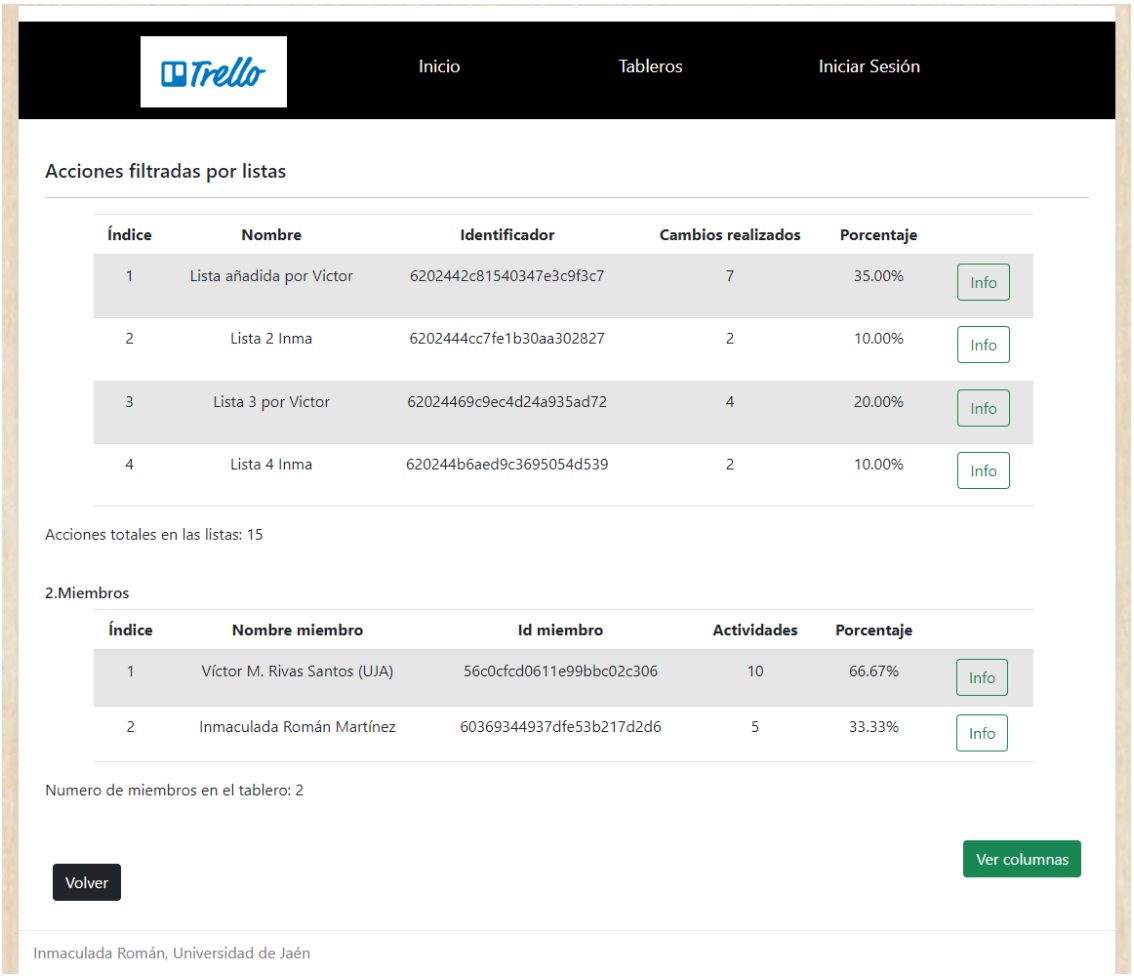

Ilustración 152: Pantalla para mostrar las acciones realizadas en las listas

Para obtener información más concreta de una lista, sólo hay que hacer click sobre el botón “info” de la primera tabla y automáticamente nos dirige a la información de una lista como vemos en la ilustración 153.

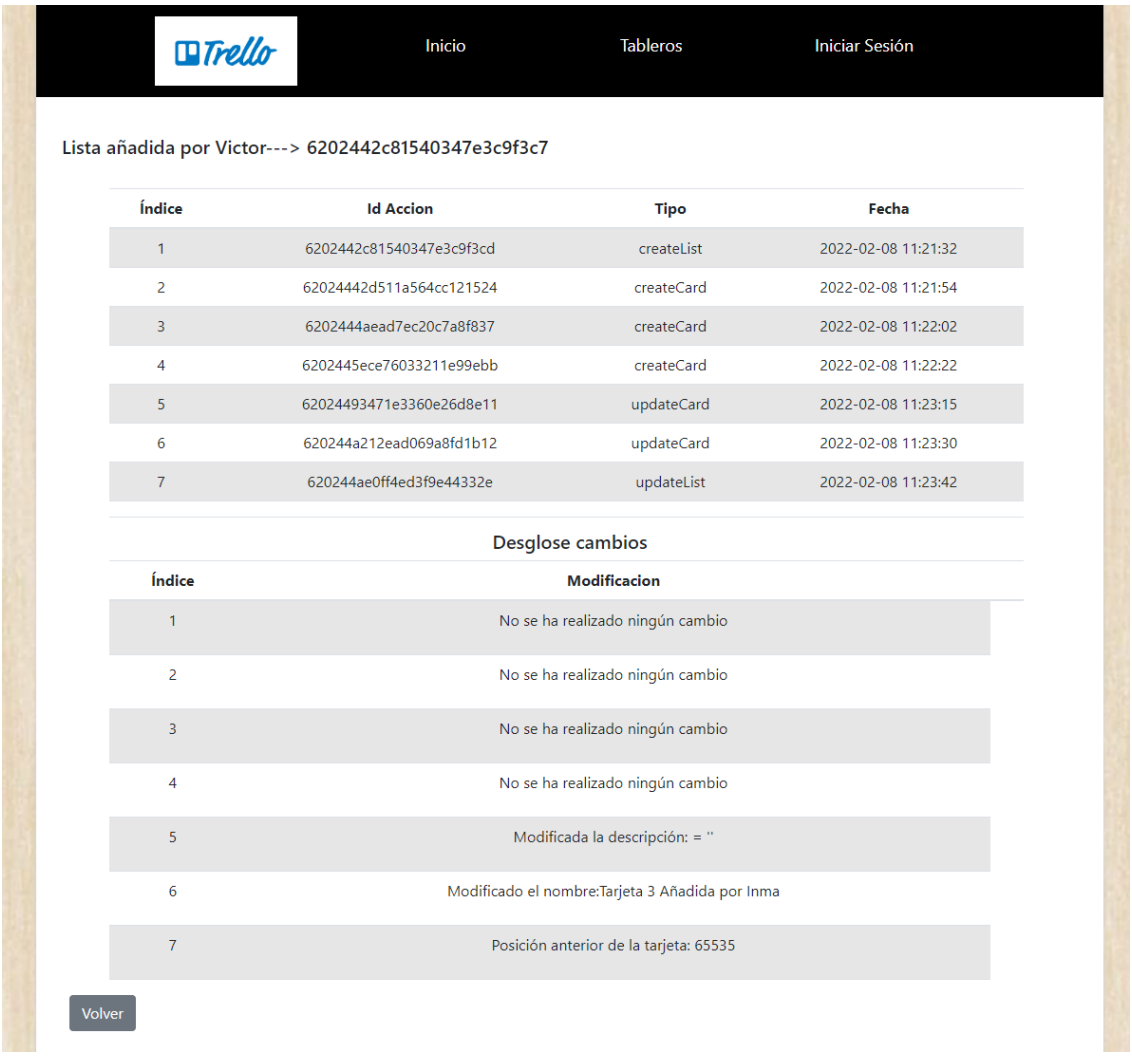

Ilustración 153: Pantalla que muestra los cambios por los que ha pasado una lista

Acciones

Volviendo a la pantalla en la que se muestra la información del tablero (Ilustración 141), si pulsamos sobre “Acciones”, se mostrará una pantalla con una tabla en la que aparecen las acciones que se han realizado, indicando su identificador, el tipo de acción y la fecha en la que se realizó. Esto se puede ver en la ilustración 154.

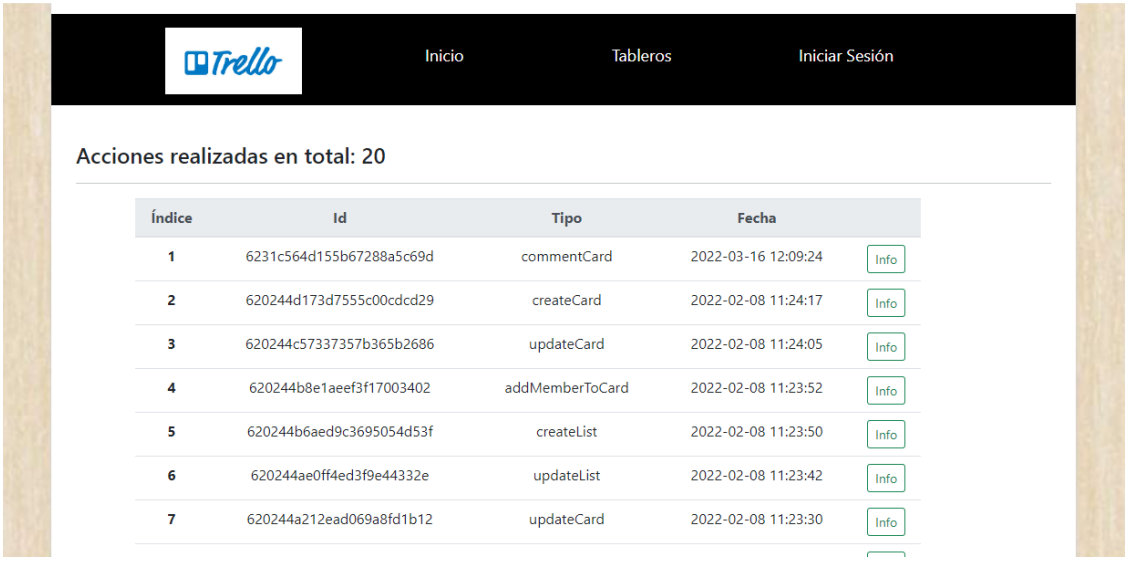

Ilustración 154: Pantalla que muestra las acciones realizadas

Al pulsar sobre el botón “info”, automáticamente nos dirige a una pantalla que nos muestra en una tabla la información más detallada de la acción (ilustración 155).

Esta acción nos indica que se añadió un comentario a una tarjeta, indicando el nombre e identificador de la tarjeta, la fecha en la que se realizó la acción, el identificador y el nombre de la lista en la que se encuentra la tarjeta, entre otros.

Ilustración 155: Pantalla que muestra una acción en detalle

Tarjetas

Para poder ver las tarjetas que hay en el tablero, tenemos que situarnos en la pantalla que nos muestra la información del tablero (ilustración 141). Al pulsar el botón “Tarjetas” nos dirige a una pantalla que nos muestra un listado completo de todas las tarjetas que se han creado en el tablero (ilustración 156).

Cada tarjeta se muestra en una tabla indicando el nombre de la tarjeta, si ha sido terminada, el identificador del tablero y la lista a la que pertenece.

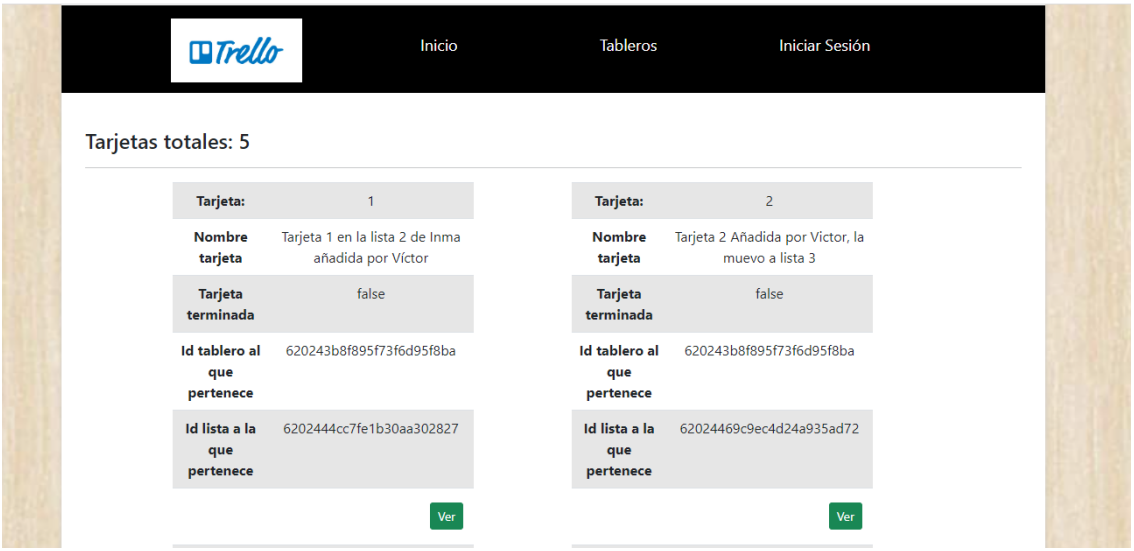

Ilustración 156: Pantalla que muestra las tarjetas del tablero

Para ver información más detallada de una tarjeta, es necesario pulsar sobre el botón “Ver” de la ilustración 156, y aparecerá una tabla con todos los detalles de una tarjeta (ilustración 157).

Ilustración 157: Pantalla que muestra en detalle una tarjeta

En esta tabla ya se muestra información más concreta de la tarjeta, como puede ser la descripción, la fecha de la última actividad, una url para acceder a ella en Trello, etc.

Listas

Al igual que antes, tenemos que situarnos en la pantalla que nos muestra la información del tablero (ilustración 141). Al pulsar sobre el botón “Listas”, aparece una pantalla indicando las listas creadas en el tablero, con el nombre e identificador de la lista y el identificador del tablero al que pertenece (ilustración 158).

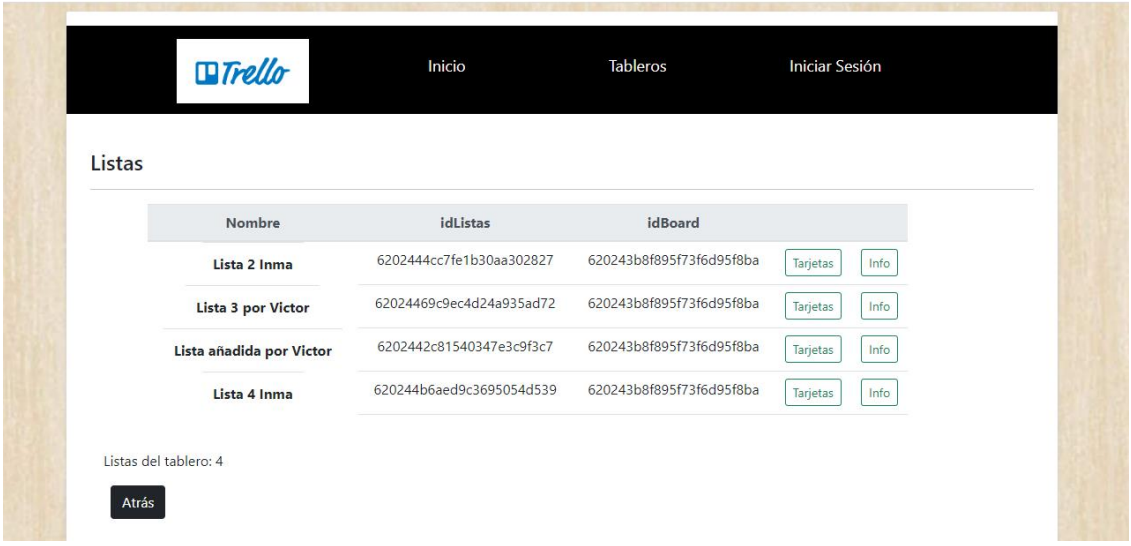

Ilustración 158: Pantalla que muestra las listas de un tablero

Como podemos observar en la ilustración 158, podemos realizar dos consultas en cada lista.

- **Consulta 1:** Obtener información más detallada de una lista (Ilustración 159)

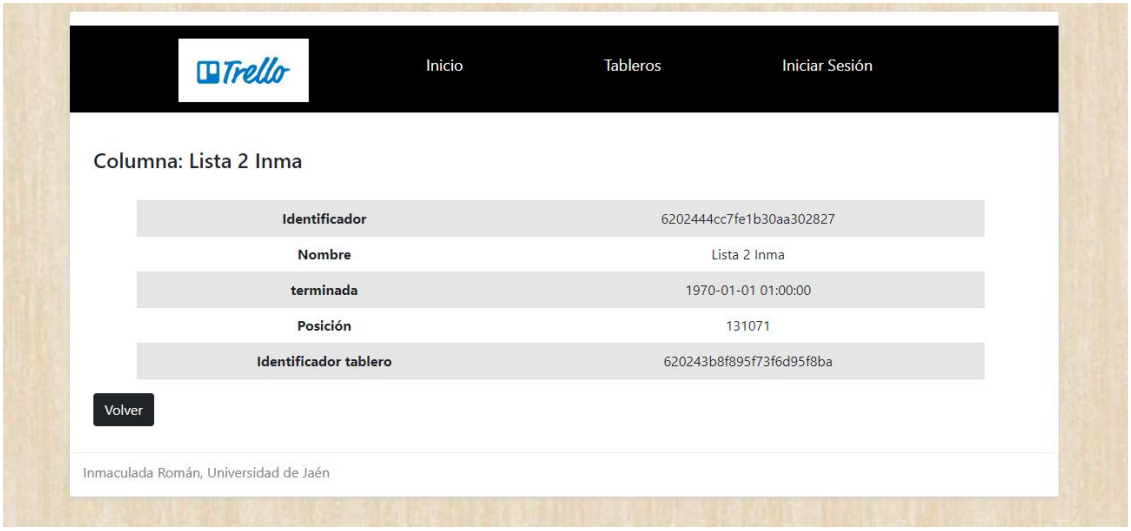

Ilustración 159: Pantalla que muestra información detallada de una lista

- **Consulta 2:** Visualizar cuáles son las tarjetas que tiene una lista, se puede ver en la siguiente ilustración 160.

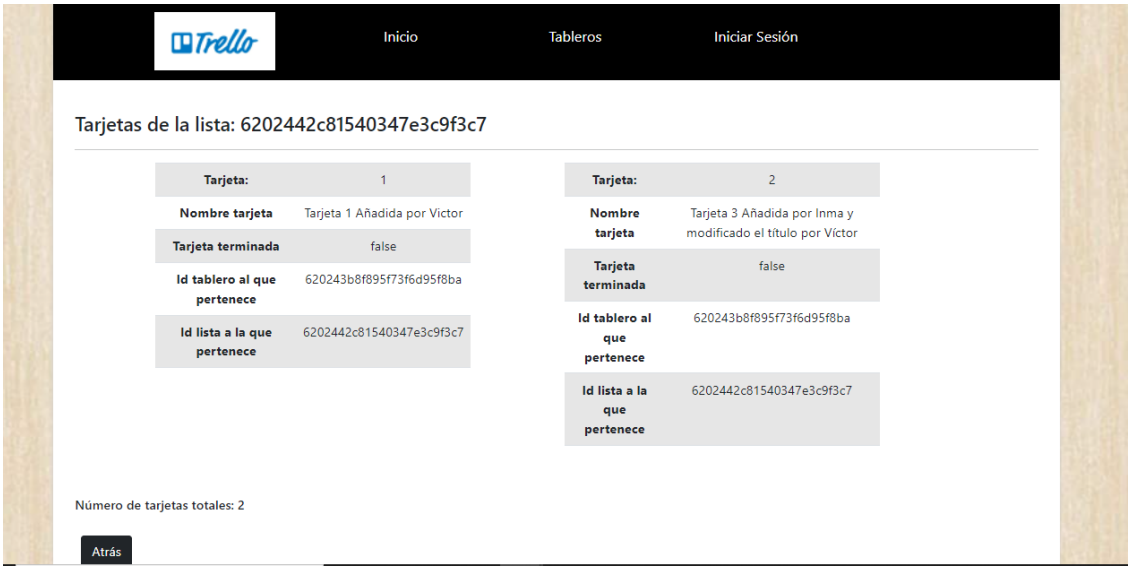

Ilustración 160: Pantalla que muestra las tarjetas que contiene una lista

Miembros

Como anteriormente, tenemos que situarnos en la pantalla que nos muestra la información del tablero (ilustración 141). Al pulsar sobre “Miembros” aparece una pantalla que muestra la información de los miembros que componen el tablero, con su identificador correspondiente y su nombre. (Ver ilustración 161).

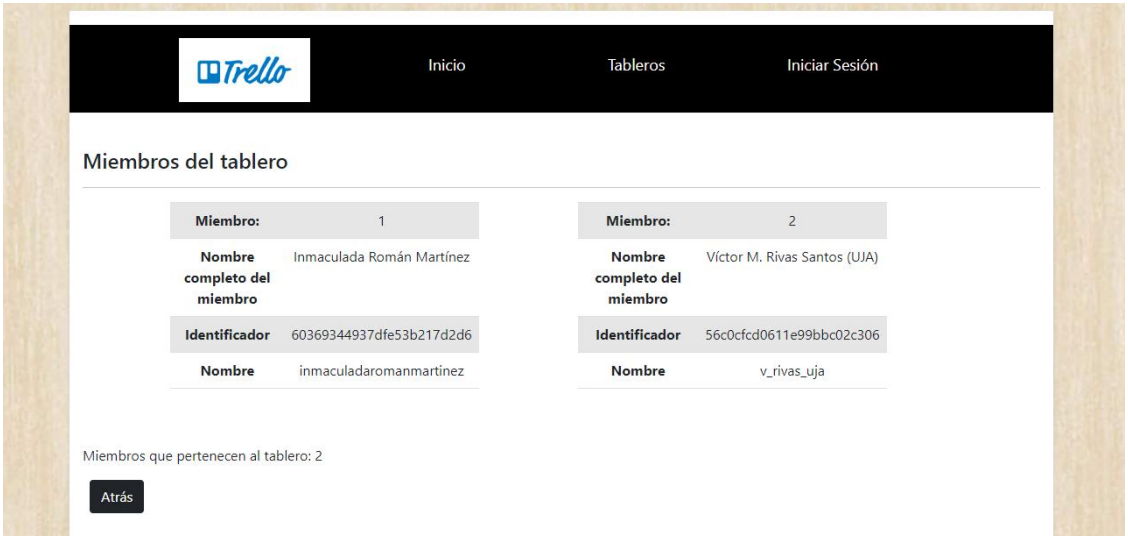

Ilustración 161: Pantalla que muestra los miembros de un tablero

Volviendo a la pantalla de inicio (ilustración 137) nos queda por explicar la tercera opción del menú principal.

3. **Inicio de Sesión.** Si el usuario está registrado en la página de Trello, puede iniciar sesión y ver su contenido igual que en Trello (ilustración 162).

Ilustración 162: Pantalla de Inicio de Sesión

Al iniciar sesión, aparecerá la siguiente pantalla (ilustración 163), la cual muestra el listado de tableros a los que el miembro pertenece. Pulsando sobre “Ver” en algún tablero, podemos realizar toda la gestión comentada anteriormente.

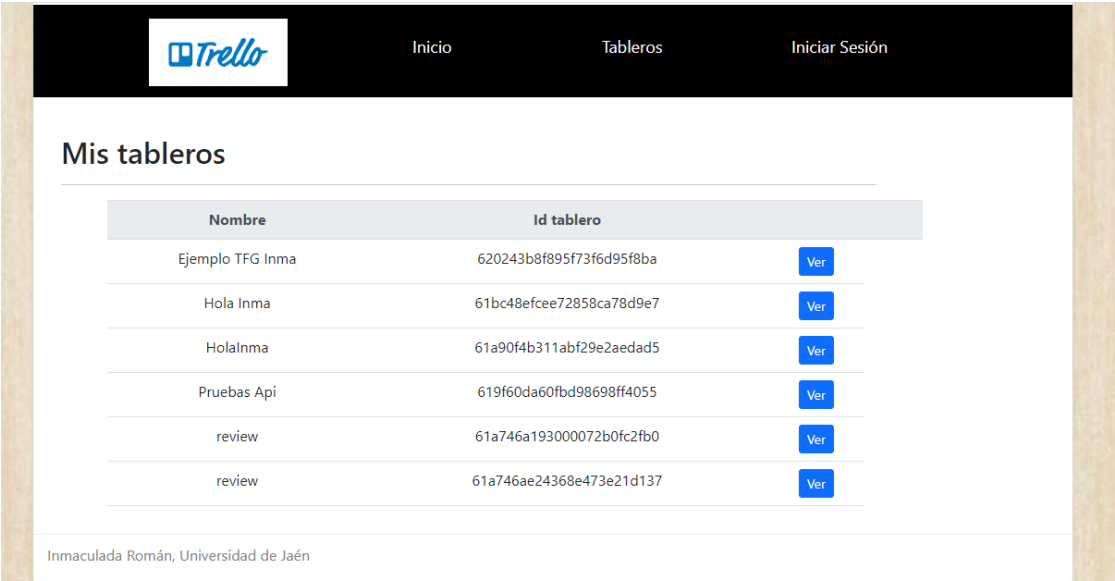

Ilustración 163: Pantalla que muestra los tableros de un usuario