

PLAN DE PROYECTO Y GESTIÓN DE RIESGOS



EROLE

Escuela Técnica Superior de Ingeniería Informática
Introducción a Ingeniería del Software

Índice

1. Miembros del equipo	4
2. Introducción.....	5
3. Roles	6
3.1. Explicación de roles	7
4. Planificación.....	8
4.1. Trello.....	9
4.2. Power-Ups	10
4.3. Sprint 2	13
4.4. Ejemplo de power-ups teamgank y planyway.....	15
4.5. Proceso de elección para el nombre de la app.....	16
4.6. Product Backlog	17
4.7. Sprint 3 + product backlog	17
4.8. Sprint 4.....	20
4.9. Sprint 5	21
4.10 Sprint 6 y 7	21
5. Gestión de Riesgos.....	22
6. Requisitos.....	23
6.1. Requisitos en MagicDraw:	29
7. Casos de uso.....	30
7.1. Casos de uso en MagicDraw	30
7.2. Documentación casos de uso.....	31
7.2.1. Registro.....	31
7.2.2. Iniciar sesión.....	32
7.2.3. Visualizar perfiles.....	33
7.2.4. Editar del perfil.....	34
7.2.5. Buscar información multimedia	35
7.2.6. Interactuar con otros usuarios.....	36
7.2.7. Etiquetado del contenido	37
7.2.8. Listado del contenido.....	38
7.2.9. Valorar el contenido con comentarios	39
7.2.10. Interactuar con perfiles	40
8. Modelo de Dominio	41

9. Pruebas.....	42
9.1. ActorTest.....	42
9.2. UserTest.....	44
9.3. CommentTest	47
10. Diagramas de secuencia.....	48
10.1. Búsqueda multimedia.....	48
10.2. Editar perfil.....	49
10.3. Iniciar sesión.....	50
10.4. Interactuar.....	51
10.5. Listado del contenido	52
10.6. Registrarse.....	53
11. Herramientas Software.....	54
11.1. Enlace a GitHub	54

1. Miembros del equipo

- Pablo Astudillo Fraga: pastudillof02@uma.es
- Jorge Camacho García: jorgecamachog@uma.es
- Marina Sayago Gutiérrez: marsay2002@uma.es
- José Fco. Artacho Martín: pepe.artacho.martin@uma.es
- Ignacio Alba Avilés: nachoalav@uma.es
- Antonio Fernández Rodríguez: afr012240@uma.es
- Diego López Reduello: diegolr02@uma.es
- Iván Delgado Alba: ivandelgadoalba@uma.es
- Manuel Jesús Jerez Sánchez: manujs@uma.es
- Mario Merino Zapata: mariomerzap@uma.es

2. Introducción

Nuestro proyecto consiste en una aplicación web enfocada al entretenimiento audiovisual (películas, series, libros, etc), donde poder consultar información (como el género, fecha de salida, director/autor, etc.) sobre diferentes títulos, así como llevar un seguimiento personal de los contenidos que ya han sido vistos o leídos.

Nuestra aplicación da la oportunidad a los usuarios de pertenecer a una comunidad donde poder interactuar entre sí y conocer a otras personas que comparten sus mismos gustos.

Además, siendo usuario de nuestra plataforma podrás llevar a cabo desafíos personales, así como los retos de la comunidad, ideales como incentivo a descubrir nuevos géneros.

3. Roles

Analista: Jorge, Nacho, Artacho
Diseñador gráfico: Marina, Manu, Antonio
Programador: Diego, Antonio, Nacho, Artacho
Tester: Iván, Mario
Documentadores: Astudillo, Mario
Scrum Master: Manu, Iván
Product Owner: Marina, Diego
Modelador: Jorge, Astudillo

Miembros	Rol 1	Rol 2
Jorge	Analista	Modelador
Mario	Tester	Documentador
Antonio	Programador	Diseñador gráfico
Marina	Product Owner	Diseñador gráfico
Manu	Scrum Master	Diseñador gráfico
Nacho	Analista	Programador
Artacho	Analista	Programador
Diego	Product Owner	Programador
Pablo	Documentador	Modelador
Iván	Tester	Scrum Master

3.1. Explicación de roles

Analista:

El analista se asegurará de reunir los requisitos y especificaciones que necesite el producto encargado por el cliente.

Documentador:

Se encarga de redactar memorias actualizadas con los avances que se van realizando en el proyecto y de las reuniones que tengan lugar, así como de actualizar y comprobar que los distintos ficheros se encuentren en el repositorio.

Diseñador Gráfico:

Da un aspecto atractivo y amigable para el usuario a la aplicación usando lenguajes de diseño tales como HTML o CSS.

Tester:

Se encarga de asegurar que los códigos no presenten fallos y funcionen de forma correcta en todas las situaciones.

Scrum Master:

Encargado de realizar la lista de tareas a realizar por el equipo de desarrolladores y organizar la fase de sprint.

Programador:

Implementa las distintas funcionalidades de la aplicación usando lenguajes de programación.

Product Owner:

Se encarga de representar los intereses del cliente dentro del equipo de desarrolladores.

Modelador:

Su papel es desarrollar los diagramas y modelaje para la visualización global del proyecto, así como el diagrama de casos de uso o historias de usuario.

4. Planificación

Tras consultar información sobre diferentes modelos de procesos software finalmente hemos decidido utilizar la metodología Scrum.

Scrum es una metodología ágil de gestión de proyectos, la cual se basa en un proceso iterativo en el que en cada iteración el producto se ve modificado, obteniendo de esta manera una versión reducida del producto final.

Scrum divide el proceso en diferentes fases y asigna a cada miembro del equipo un rol.

Los roles asignados en Scrum son los siguientes:

- **Scrum master:** Se encarga de organizar y dirigir las reuniones, además de asegurarse de que todos los miembros dispongan de las herramientas necesarias para realizar sus tareas. Lidera y dirige al equipo.
- **Product owner:** Es el que tiene la visión más clara de lo que el equipo va a construir. Se encarga de indicar que debe ir al Backlog y representa a los usuarios y clientes del producto.
- **Equipo de desarrollo:** Son los profesionales que se encargan del desarrollo del producto.

El proceso se divide en las siguientes fases:

- **Visión:** Fase de recogida de ideas.
- **Sprint:** Fase de desarrollo.
- **Incremento del producto:** Fase de implementación y modificación del producto.

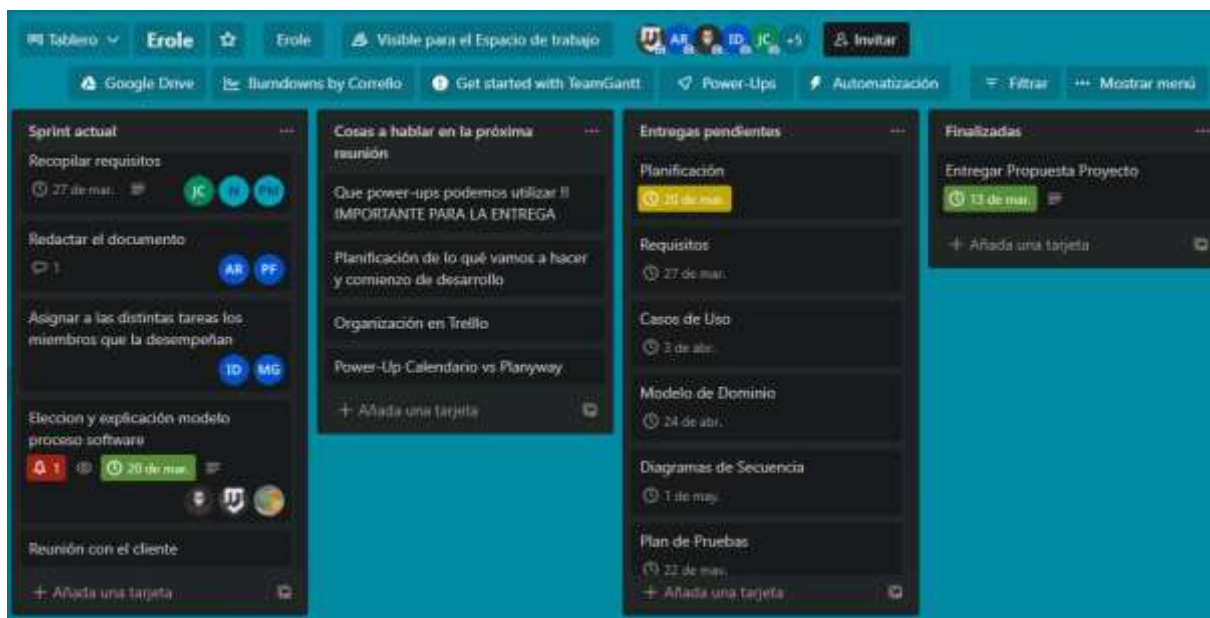
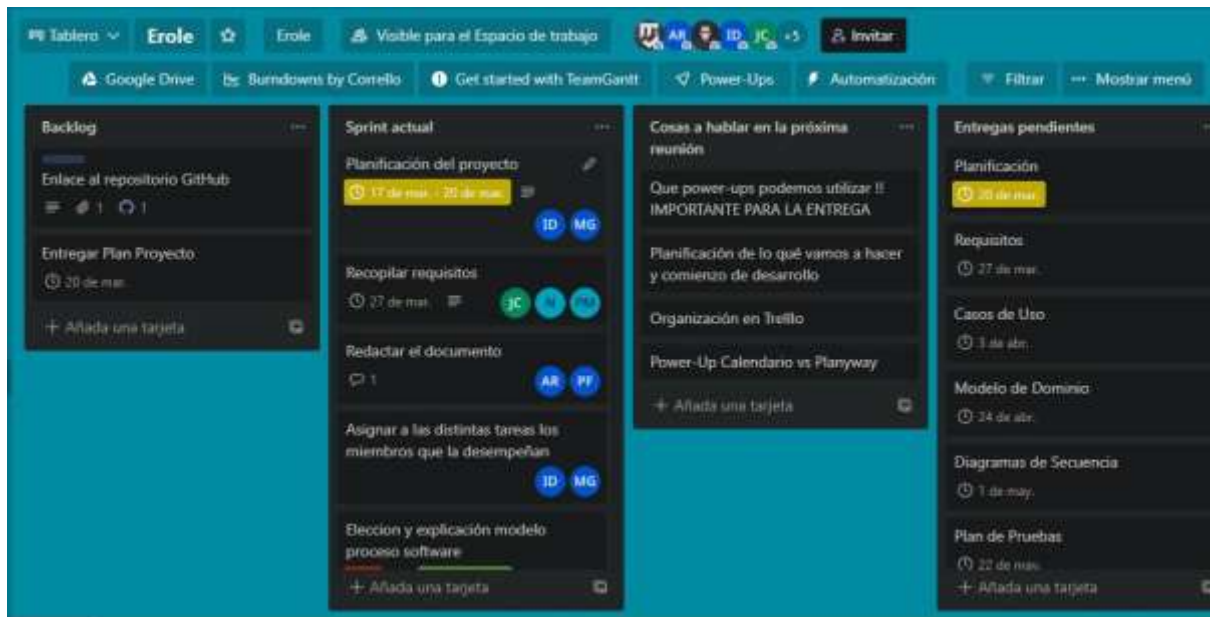
En Scrum, se suelen usar unos documentos concretos para facilitar el entendimiento y la organización de tareas, estos son: el product backlog, el sprint backlog y el burndown chart.

Hemos escogido Scrum como metodología para nuestro proyecto software principalmente por la implicación que este modelo otorga al cliente durante todo el proceso, de esta manera que nuestro cliente está informado en todo momento de la fase en la que se encuentra el producto y de los cambios que ha sufrido a lo largo del tiempo.

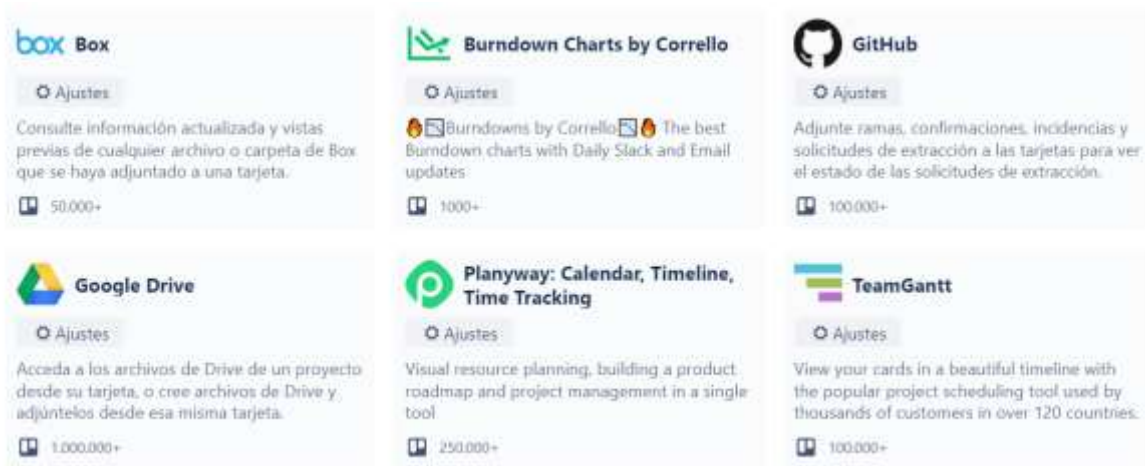
De esta manera recibimos un feedback de nuestro cliente, lo cual nos es muy útil para conseguir desarrollar un producto final lo más adecuado posible a las expectativas del cliente.

Además, Scrum es una metodología ideal para proyectos pequeños con bastante incertidumbre puesto que es muy flexible a cambios durante el proceso, lo que nos permite reducir riesgos en el proyecto.

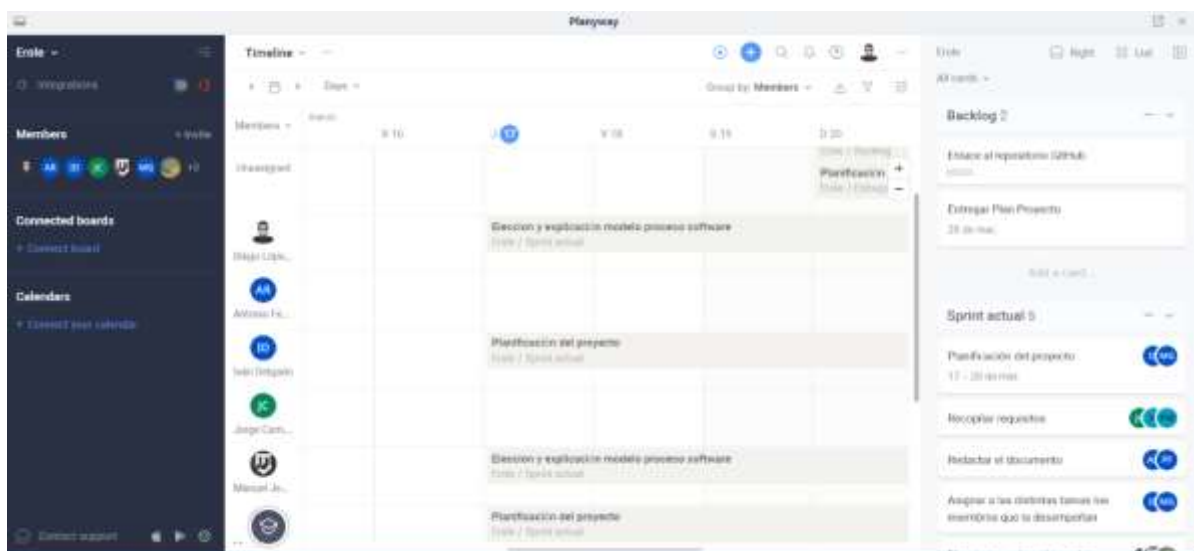
4.1. Trello

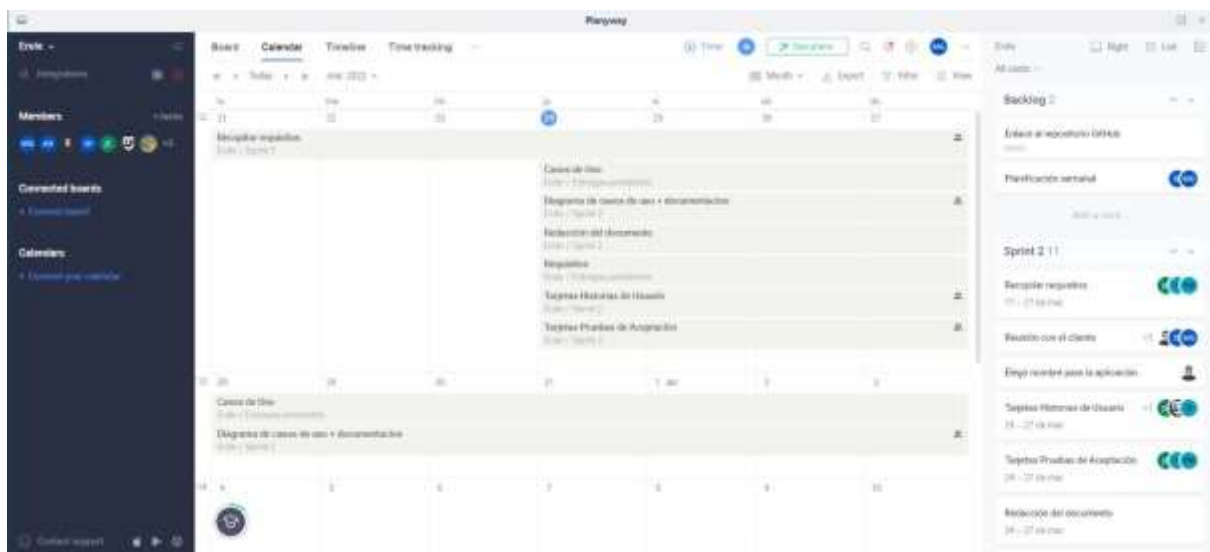
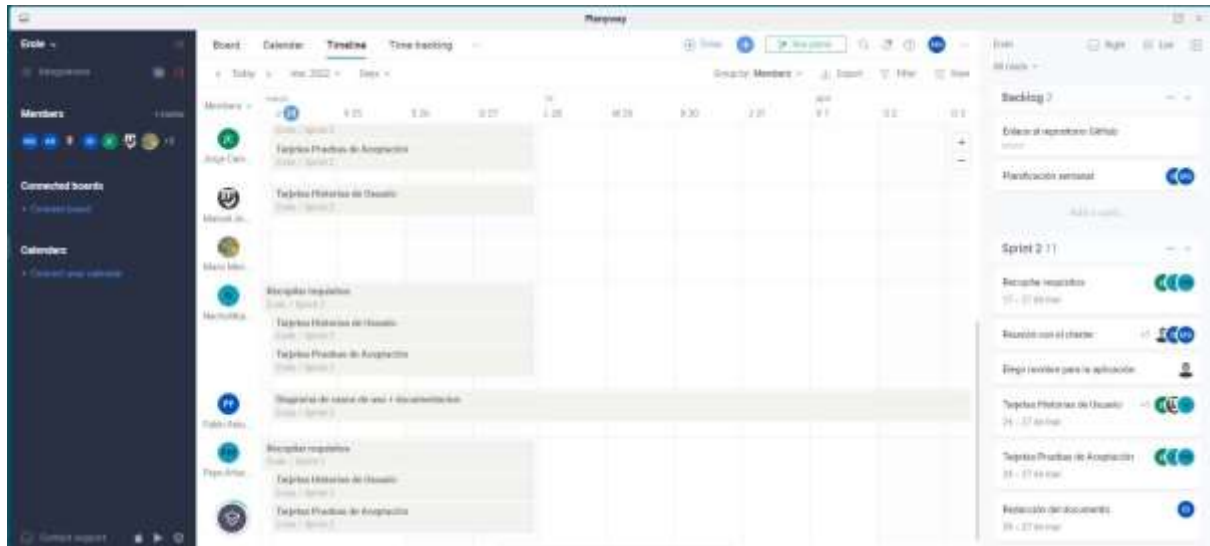
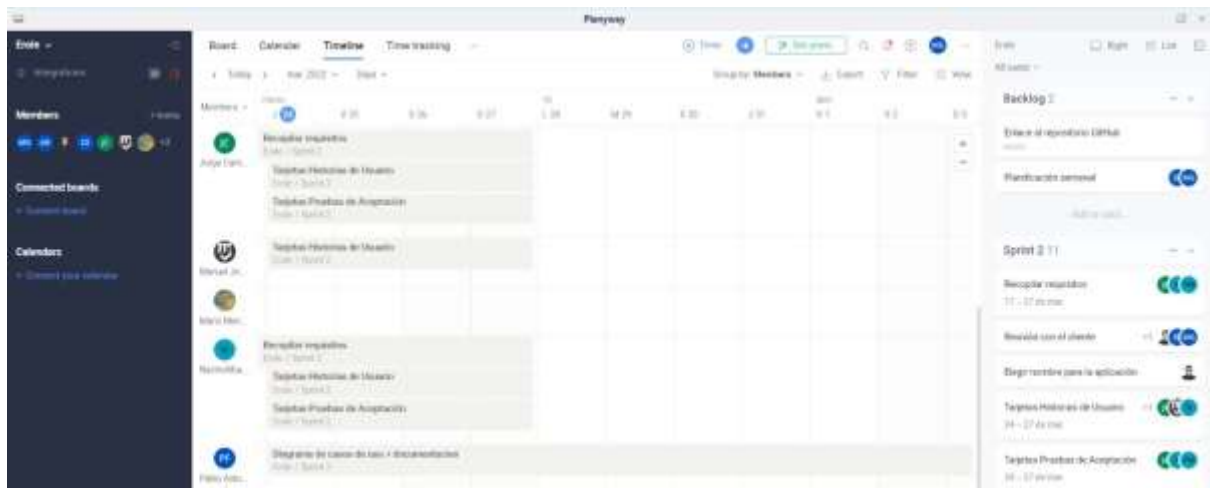


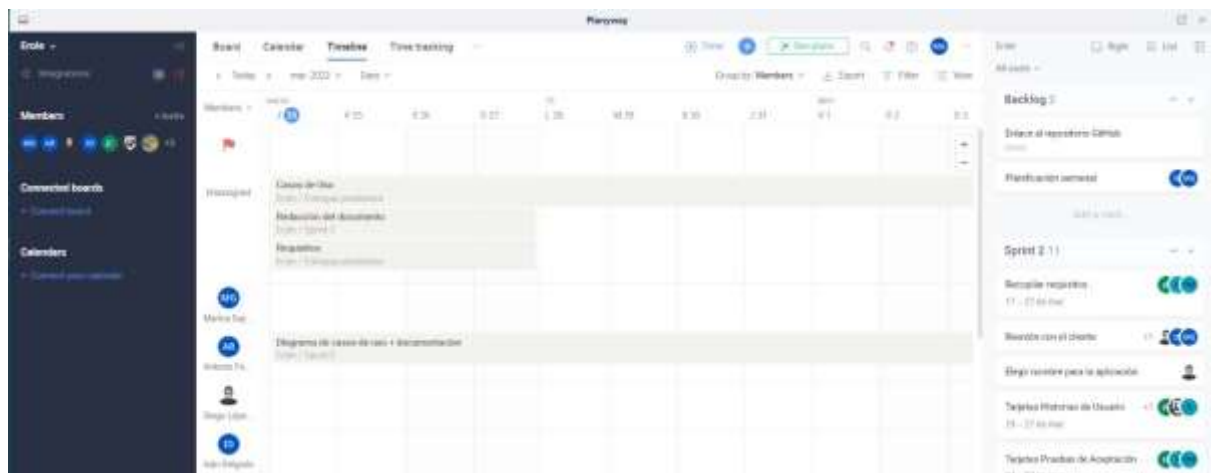
4.2. Power-Ups



- **Google drive:** Vinculación de la cuenta de google drive para facilitar la subida de archivos.
- **TeamGantt:** Elaboración de líneas de tiempo.
- **GitHub:** Supervisión de todo lo que ocurre en GitHub.
- **Box:** Integra un servicio de nube sin tener que salir de Trello.
- **Planyway:** calendario alternativo al predeterminado de Trello con más opciones y funciones que nos permitirá organizarnos mejor.







4.3. Sprint 2

NachoAlbaUMA 3 cards

Add a card...

Recopilar requisitos 17 – 27 de mar.

Tarjetas Historias de Usuario +1 24 – 27 de mar.

Tarjetas Pruebas de Aceptación 24 – 27 de mar.

Add a card...

Pablo Astudillo Fraga 3 cards

Add a card...

Diagrama de casos de uso + documentación 24 de mar. – 3 de abr.

Documentación casos de uso

Add a card...

Redactar el documento

Add a card...

Pepe Artacho Martín 3 cards

Add a card...

Recopilar requisitos 17 – 27 de mar.

Tarjetas Historias de Usuario +1 24 – 27 de mar.

Tarjetas Pruebas de Aceptación 24 – 27 de mar.

Diego López Reduelo 3 cards

Add a card...

Reunión con el cliente +1

Elegir nombre para la aplicación

Add a card...

✓ Reunión y explicación del modelo propuesto al cliente 17 – 20 de mar.

Add a card...

Wén Delgado 5 cards

Planificación semanal

Reunión con el cliente +1

Añade rol moderador

✓ Planificación del proyecto 17 - 20 de mar

Asignar a los distintos tareas los miembros que la desempeñan

Add a card...

Add a card...

Add a card...

Add a card...

Add a card...

Jorge Carrasco 3 cards

Recopilar requisitos 17 - 27 de mar

Tarjetas Historias de Usuario 24 - 27 de mar

Tarjetas Pruebas de Aceptación 24 - 27 de mar

Add a card...

Add a card...

Add a card...

Add a card...

Maria Saegui Gutiérrez 3 cards

Planificación semanal

Reunión con el cliente +1

Captura gráfica Business charts

✓ Planificación del proyecto 17 - 20 de mar

Asignar a los distintos tareas los miembros que la desempeñan

Add a card...

Add a card...

Add a card...

Add a card...

Add a card...

Antonio Fernandez Rodriguez 3 cards

Diagrama de casos de uso + documentación 24 de mar - 3 de abr

Documentación casos de uso

Redactar el documento

Add a card...

Add a card...

Add a card...

Add a card...

Manuel Jesús Jerez Sánchez 3 cards

Elegir nombre para la aplicación

✓ Gestión y explotación modelo proceso software 17 - 20 de mar

Add a card...

Add a card...

Add a card...

Mario Merino Zapata 3 cards

Planificación semanal

Reunión con el cliente +1

✓ Gestión y explotación modelo proceso software 17 - 20 de mar

Add a card...

Add a card...

Add a card...

Add a card...

Unicogit 14 cards

Enlace al repositorio GitHub

Redacción del documento 24 - 27 de mar

Requisitos 24 - 27 de mar

✓ Entrega Plan Proyecto 20 de mar

✓ Entrega Proyecto Proyecto 13 de mar

Añade enlace del repositorio git en el documento

Casos de Uso 24 de mar - 3 de abr

✓ Planificación 20 de mar

Modelo de Datos 24 de mar

Que poner uso podemos utilizar 1 IMPORTANTE PARA LA ENTREGA

Diagramas de Secuencia 1 de mar

Power-Up Calendar vs Playway

Plan de Pruebas 22 de mar

Proyecto 5 de jun

Add a card...

Add a card...

Add a card...

Add a card...

Add a card...

Add a card...

Add a card...

Add a card...

Add a card...

Add a card...

Add a card...

4.4. Ejemplo de power-ups teamgank y planyway

✕

Recopilar requisitos

en la lista [Sprint 2](#)

Miembros

JC

N

PM

+

Vencimiento

27 de mar. a las 13:00

▼

Descripción

Editar

Sacando los requisitos funcionales y no Funcionales

TeamGantt Task

View in [TeamGantt](#)

✓ 100%

Erole

First Group (click to rename)

Set Start Date

Set End Date

Mar 27, 2022

Mar 27, 2022

Dependencies

Assigned Resources

Depends on

Dependent on this task

Planyway

Dates

17 – 27 de mar.

Sugerencias

Unirse

Añadir a la tarjeta

Miembros

Etiquetas

Checklist

Fechas

Adjunto

Portada

Campos personali...

Añade listas desplegables, campos de texto y fechas, entre otras cosas, a sus tarjetas.

Comenzar prueba gratis

Power-Ups

Box

GitHub

Google Drive

Story Points

TeamGantt

View on timeline


4.5. Proceso de elección para el nombre de la app

Actividad

Mostrar detalles


MG

Escriba un comentario...

 **Mario Merino Zapata** hace 2 minutos


CompactFilm (usandose las siglas CF)
Lo siento me hacía gracia la broma con el CD 🤔

😊 - Responder

 **NachoAlbaUMA** hace 14 minutos


Metaerole

😊 - Responder

 **Jorge Camacho** hace 15 minutos


Movierole // Cinerole // Filmerole // Pelicurole

😊 - Responder

 **NachoAlbaUMA** hace 16 minutos


Filmerole

😊 - Responder

 **NachoAlbaUMA** hace 17 minutos

Movierole

😊 - Responder

 **Manuel Jesús Jerez Sánchez** hace 18 minutos

Añadir aquí alguna sugerencia adicional

😊 - Responder

Acciones

Mover

Copiar

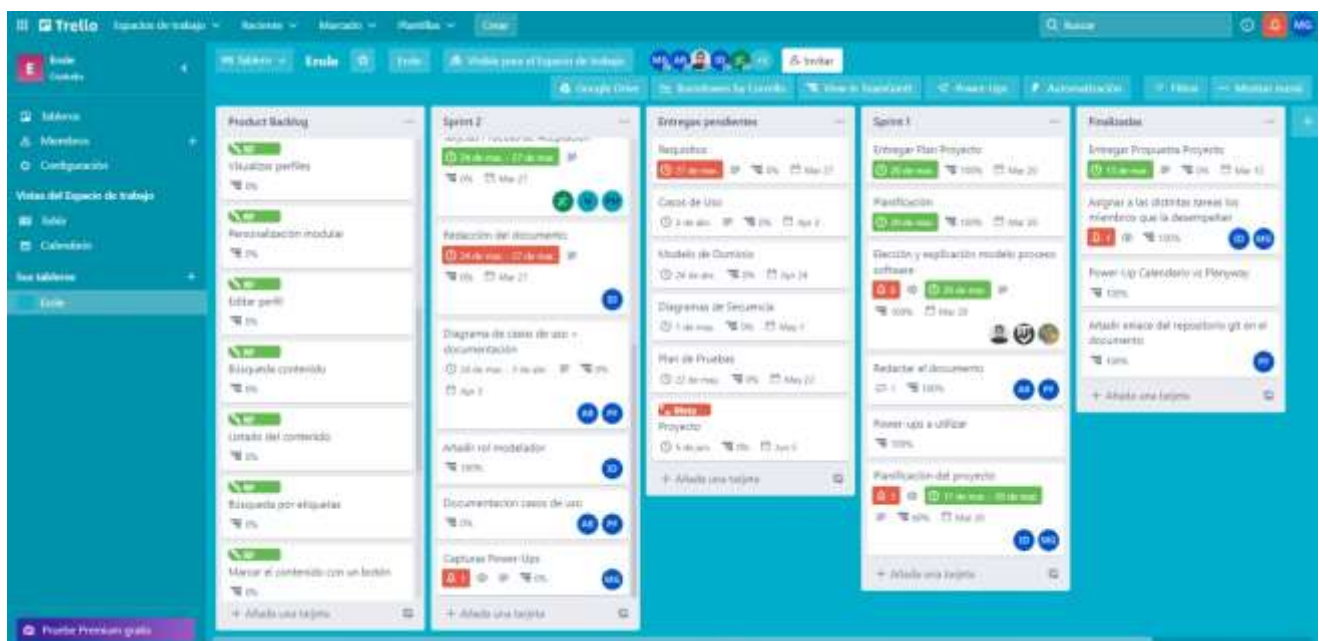
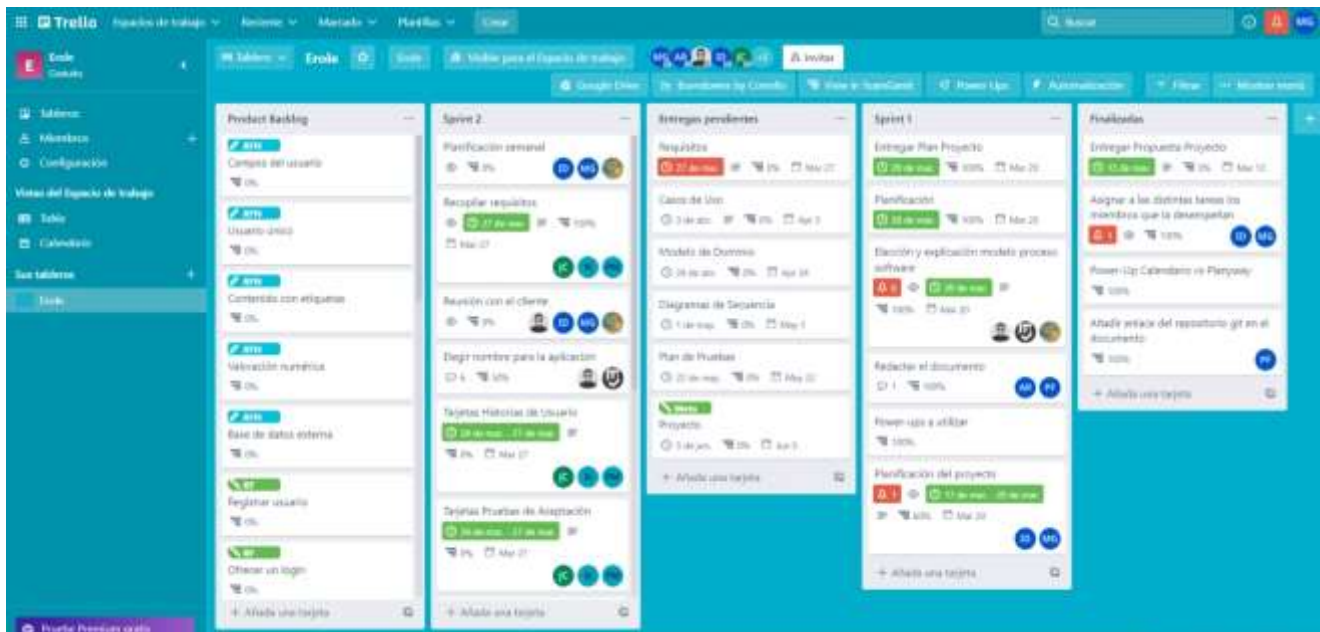
Crear plantilla

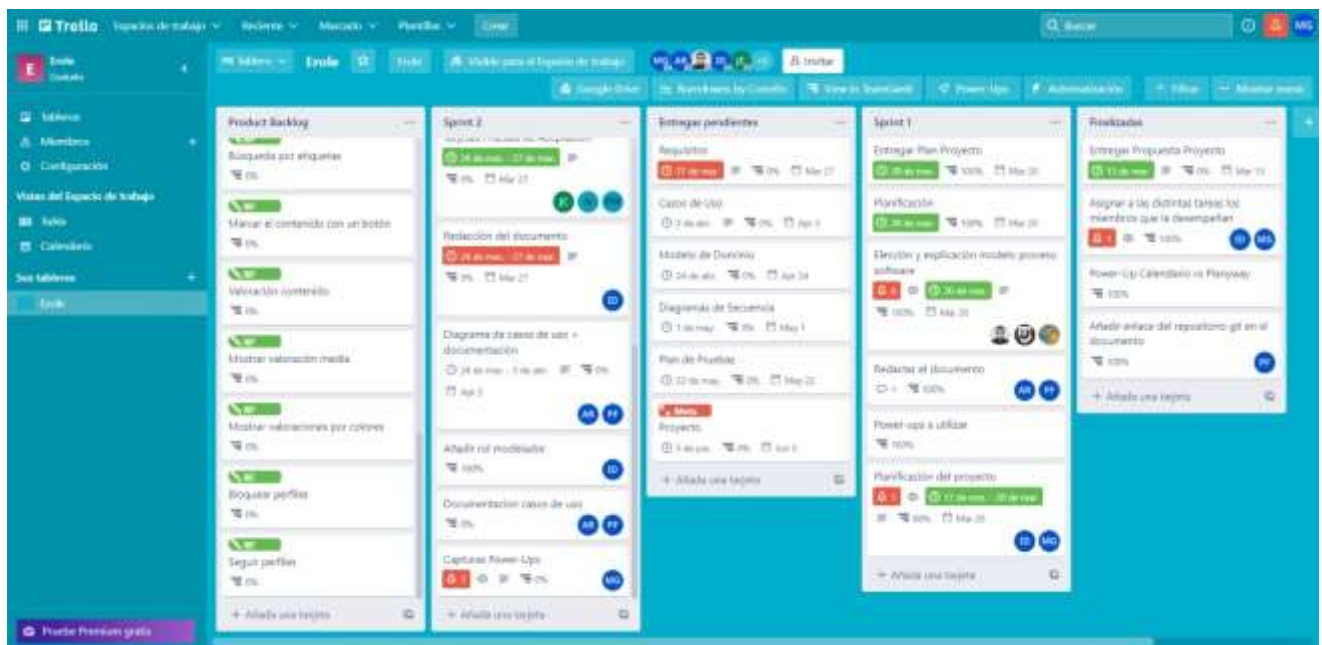
Seguir

Archivar

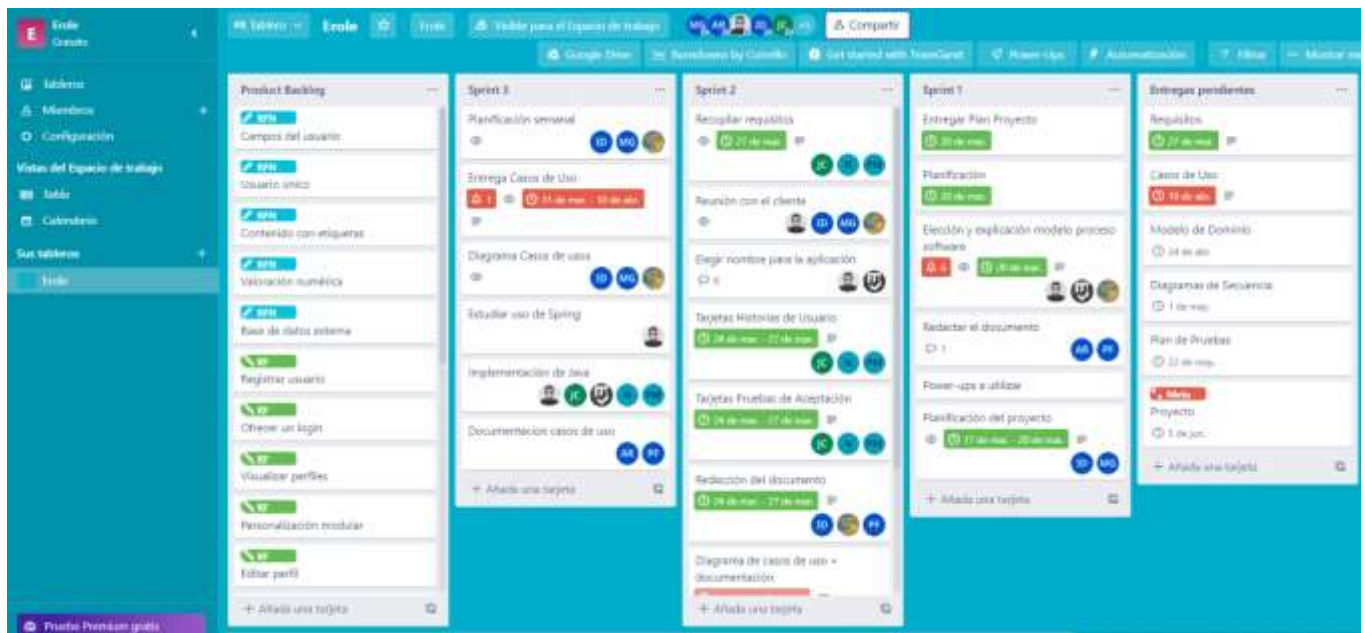
Compartir

4.6. Product Backlog

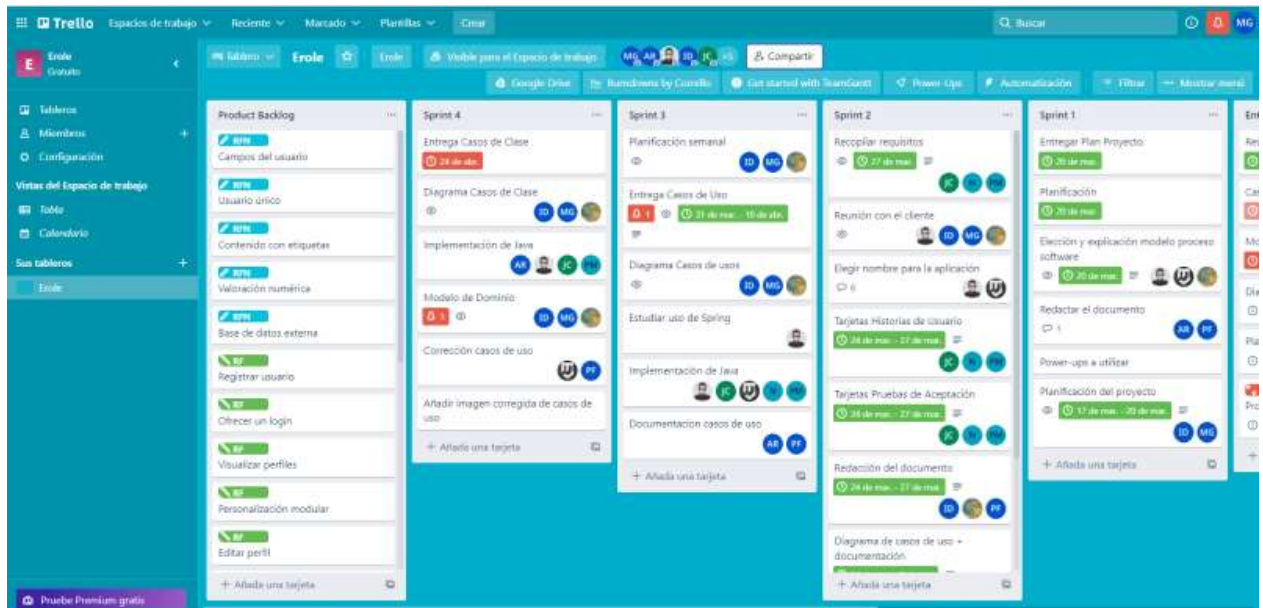




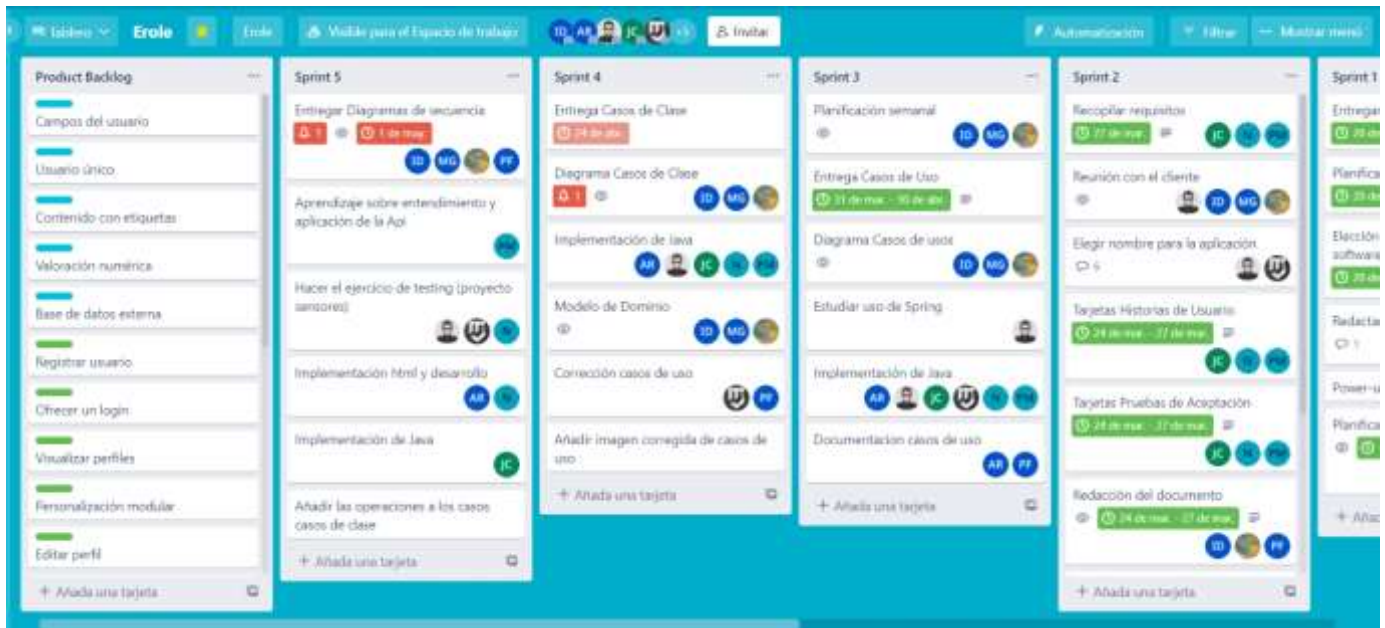
4.7. Sprint 3 + product backlog



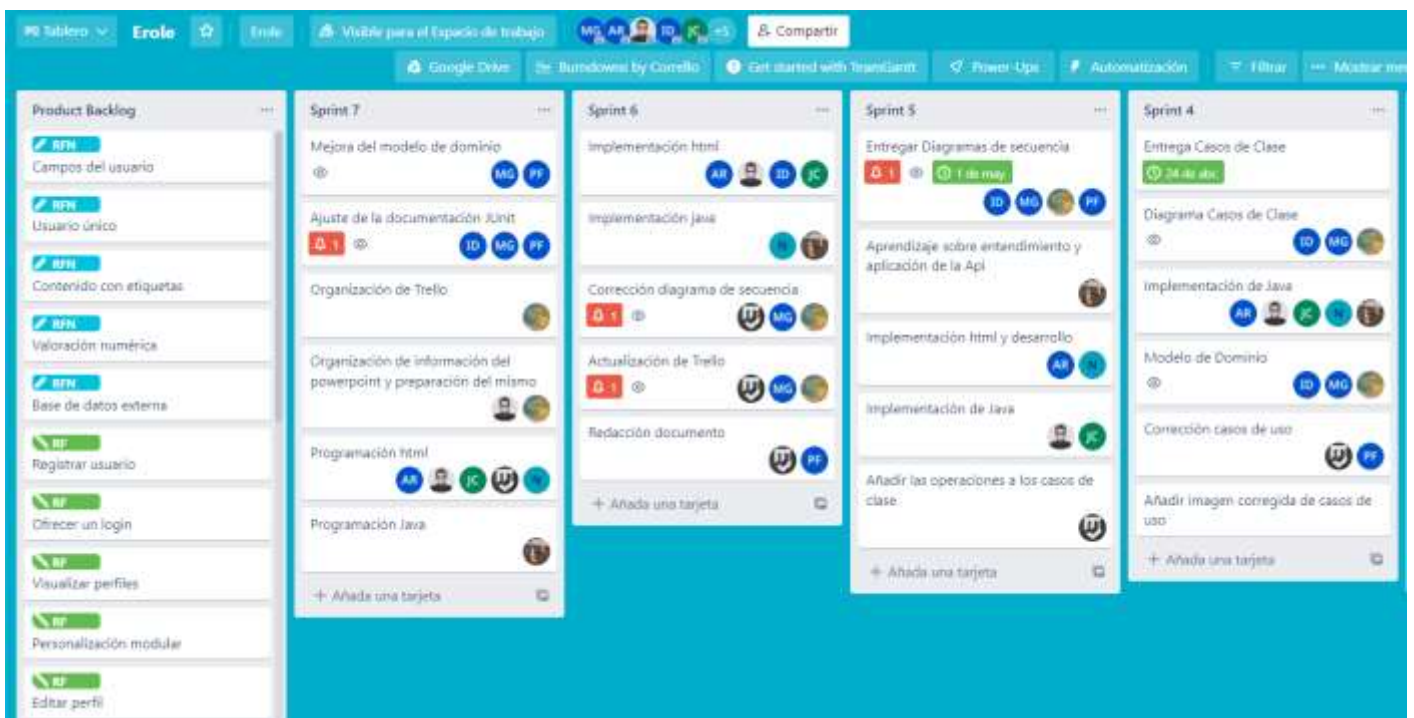
4.8. Sprint 4



4.9. Sprint 5



4.10 Sprint 6 y 7



5. Gestión de Riesgos

Riesgo	Descripción	Probabilidad	Impacto	Tipo	Estrategia
Pérdida de datos	Perder distintos datos de la base de datos del proyecto.	Baja	Alto	Proyecto y producto	Disponer de copias de seguridad e intentar encontrar datos que puedan suplir los que se hayan perdido
Falta de tiempo	Se ha asignado un tiempo insuficiente para realizar una tarea.	Media	Alto	Proyecto y producto	Reducir el producto final
Un compañero abandona el grupo	Un compañero del equipo decide abandonar el proyecto o la asignatura.	Baja	Leve/Medio	Proyecto	Repartir sus funciones entre el resto del equipo
Cambios bruscos en el producto	El cliente decide hacer un cambio drástico en el producto final.	Leve	Crítico	Proyecto	Reutilizar lo posible y alcanzar el mismo punto de desarrollo anterior
Contratiempos CASE	Incorrecto funcionamiento de las herramientas CASE.	Media/Leve	Leve	Producto	Buscar otras alternativas software
No encontrar solución a cierto problema	Desconocimiento de cualquier ámbito del proyecto.	Alta	Leve/Medio	Proyecto	Informarse según documentaciones oficiales y foros para encontrar la solución

6. Requisitos

RF1: Registrar usuario

Como interesado en usar Erole,

Quiero ser capaz de crearme un perfil con mi información personal

Para poder usar la aplicación.

Pruebas de aceptación

- Se registra un usuario nuevo y se crea un perfil con los campos rellenados por el usuario.
- Se intenta registrar un usuario nuevo con un nombre de usuario o correo ya registrado y se recibe un mensaje de error (no se crea por tanto el perfil).
- No se permite el registro de un nuevo usuario si no ha rellenado el nombre, el correo y la contraseña o no son válidos.

RF2: Campos del usuario

Como gestor de la base de datos

Quiero tener disponible las credenciales del usuario: nombre de usuario, correo, contraseña de forma obligatoria. Y de forma opcional nombre, apellido, foto de perfil, biografía escrita de máximo 200 caracteres, fecha de nacimiento.

Para gestionar eficientemente el inicio de sesión.

RNF1: Usuario único

Como gestor de la base de datos,

Quiero que cada cuenta tenga asociado un correo electrónico y nombre y estos a su vez sean únicos

Para evitar conflictos de correos/cuentas repetidas

RF3: Ofrecer un login

Como usuario de la aplicación,

Quiero poder iniciar sesión con mi propia clave y que me brinde seguridad de que otra persona no acceda a mi cuenta

Para poder acceder a la aplicación.

Pruebas de aceptación

- Se pide al usuario que introduzca su nombre de usuario y su contraseña.
- Si el usuario no se ha registrado previamente o si ha puesto algún campo erróneamente, se recibe un mensaje de error
- Si los campos son correctos y corresponden a un perfil registrado con anterioridad, se permite el acceso a la aplicación.

RF4: Visualizar perfiles

Como usuario de la aplicación,

Quiero ver los perfiles de otros usuarios de la aplicación y su información

Pruebas de aceptación

- El usuario clicla en el perfil de otro usuario y le lleva a ver la información de éste (nombre, foto, listado de películas...).

RF5: Personalización modular

Como usuario de la aplicación,

Quiero ser capaz de modificar la información que se muestra en mi perfil por módulos (listas de contenido, película favorita...)

Para poder personalizar mi perfil.

Pruebas de aceptación

- El usuario no modifica nada y, por tanto, el perfil permanece intacto.
- El usuario modifica los distintos módulos que conforman su perfil y los cambios se ven actualizados.

RF6: Editar perfil

Como usuario,

Quiero tener la posibilidad de modificar los datos de mi perfil

Para mantener mi información actualizada.

Pruebas de aceptación

- El usuario modifica los datos que crea convenientes, y dichos datos se quedan modificados.
- Ningún otro dato del perfil sufre ningún cambio.
- Si el usuario cambia a un nombre/correo ya existente, salta un aviso de error (igual que en el login).

RF7: Búsqueda contenido

Como usuario,

Quiero tener la posibilidad de realizar búsquedas de los elementos almacenados en la BD

Para poder encontrar los contenidos deseados de forma sencilla.

Pruebas de aceptación

- El usuario escribe en la barra de búsqueda el título de uno de los contenidos y se muestran por pantalla los coincidentes.
- El usuario filtra los resultados por etiquetas y como resultado aparecen sólo los datos que cumplan las restricciones impuestas.
- Se busca un título que no se encuentre en la base de datos y no muestra nada.

RF8: Listado del contenido

Como usuario,

Quiero tener la posibilidad de ver un listado con el contenido que podrían interesarme

Para sacarle mayor partido a la aplicación.

RF9: Búsqueda por etiquetas

Como usuario

Quiero poder filtrar el contenido según las etiquetas disponibles

Pruebas de aceptación

- Se seleccionan una o varias etiquetas (como por ejemplo el género) por la que filtrar el contenido.
- Se intenta buscar alguna película existente que no cumpla el filtro y no sale como resultado.
- Si no se selecciona ninguna etiqueta no se aplica ningún filtro.

RNF2: Contenido con etiquetas

Como gestor de la base de datos

Quiero que el contenido pueda estar etiquetado según género, duración, plataforma, reparto.

Para poder ofrecerlo de manera más eficiente al usuario.

RF10: Marcar el contenido con un botón

Como usuario

Quiero tener la opción de seleccionar qué contenido me ha gustado, no me ha gustado, quiero ver, visto.

Para que la aplicación pueda hacerme buenas recomendaciones y poder compartir mis gustos con otros usuarios (perfil)

Pruebas de aceptación

- El usuario marca el contenido con un botón de 'me gusta' y se ve actualizada la lista de películas que le han gustado en su perfil.
- De igual manera ocurre con el botón de 'no me gusta', 'quiero ver' y 'visto'.
- Si el usuario desmarca el botón, la lista correspondiente del perfil se ve modificada (se elimina el contenido correspondiente).
- Si el usuario visita una página pero no marca ninguna opción, no se ve modificado el perfil.

RF11: Valoración contenido

Como usuario

Quiero poder dejar una valoración en cada contenido

Pruebas de aceptación

- Si el usuario da una valoración en la página del contenido, ésta perdura y puede afectar a la valoración media (ver más abajo).
- Si el usuario decide además escribir un comentario, perdurará de igual manera y se mostrará con el color de fondo adecuado (como se explica más abajo).

RF12: Valoración numérica

Como gestor de la base de datos

Quiero que la valoración del contenido se numérica desde 0 hasta 5 con posibilidad de añadir un comentario de 300 caracteres como máximo.

Para poder gestionarla de forma más sencilla

RF13: Mostrar valoración media

Como usuario

Quiero que cuando acceda a la página del contenido se muestre la valoración media

Para conocer las opiniones de los demás usuarios

Pruebas de aceptación

- El usuario da una valoración numérica como reseña en la página de un contenido concreto y, si procede, se ve actualizada la valoración media general.
- Si el usuario no escribe ninguna valoración, no se ve afectada la valoración media general.

RF14: Mostrar valoraciones por colores

Como usuario

Quiero que los comentarios de cada contenido sean mostrados con un color de fondo en función de la valoración, que vaya desde el rojo hasta el verde, siendo el 0 un color rojo intenso y el 5 un verde intenso

Para que se visualmente podamos saber si le ha gustado o no

Pruebas de aceptación

- Se pone una reseña negativa (0) y, al publicarlo, aparece sobre un fondo rojo intenso.
- Se pone un reseña muy positiva (5) y, al publicarlo, aparece sobre un fondo verde intenso.
- Se pone un reseña neutra (3) y, al publicarlo, aparece sobre un fondo amarillo pálido.
- Si se ponen reseñas negativas o positivas no tan rotundas, aparecerán los colores correspondientes más pálidos.

RNF4: Base de datos externa

Como equipo de desarrollo

Queremos que los datos sean importados de una base de datos externa a través de una API

Para no tener que introducir los datos manualmente y tener disponibles una gran cantidad de datos actualizados

RF15: Bloquear perfiles

Como usuario

Quiero tener la capacidad de bloquear cualquier interacción con ciertos usuarios

Para tener una seguridad plena y una buena experiencia

RF16: Seguir perfiles

Como usuario

Quiero poder seguir a otros usuarios

Para poder ver el contenido que ellos comparten

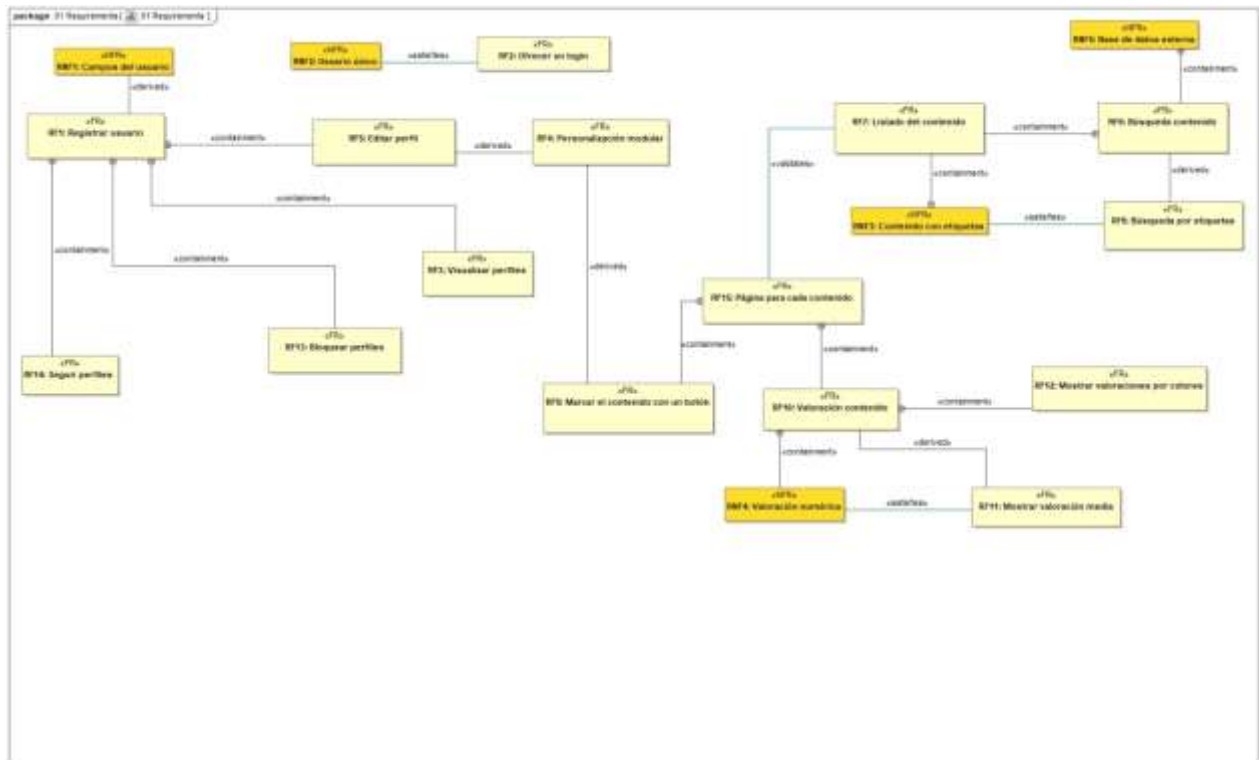
RF17: Página para cada contenido

Como usuario

Quiero que haya una página por cada contenido dentro de la base de datos

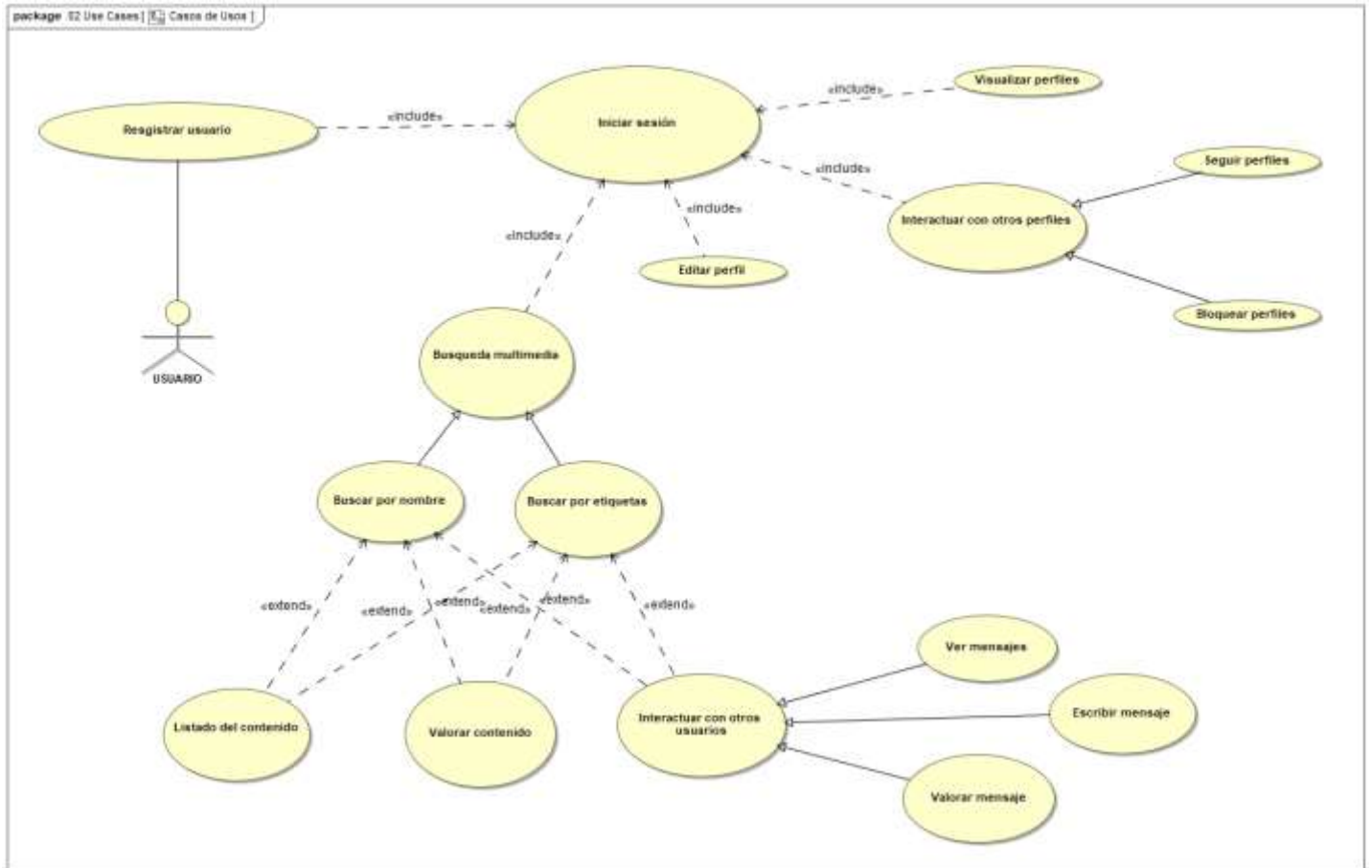
Para poder ver las valoraciones y marcar el contenido.

6.1. Requisitos en MagicDraw:



7. Casos de uso

7.1. Casos de uso en MagicDraw



7.2. Documentación casos de uso

7.2.1. Registro

CU1 → Registrarse en la aplicación

- **Contexto de uso:**

Cuando un usuario quiera disponer de perfil en la aplicación, se registrará.

- **Precondiciones y activación:**

El usuario está conectado a la aplicación y en el formulario de registro.

- **Garantías de éxito / Postcondición:**

El usuario se registra en el sistema y tiene acceso a su perfil.

- **Escenario principal:**

1. El usuario pulsa la opción para registrarse.
2. El sistema muestra el formulario para registrarse.
3. El usuario rellena la información para registrarse.
4. El usuario pulsa la opción de registrarse.
5. La cuenta se crea en el sistema y se muestra un mensaje de éxito al usuario.

- **Escenarios alternativos:**

La cuenta no se ha creado correctamente y se muestra un mensaje al usuario.

7.2.2. Iniciar sesión

CU1 → Iniciar sesión en la aplicación.

- **Contexto de uso:**

Cuando un usuario quiera acceder a su perfil en la aplicación, iniciará sesión.

- **Precondiciones y activación:**

El usuario está conectado a la aplicación y en el formulario de inicio de sesión.

- **Garantías de éxito / Postcondición:**

El usuario inicia sesión en el sistema y tiene acceso a su perfil.

- **Escenario principal:**

1. El usuario pulsa la opción para iniciar sesión.
2. El sistema muestra el formulario para iniciar sesión.
3. El usuario rellena la información para iniciar sesión (usuario y contraseña).
4. El usuario pulsa la opción de iniciar sesión.
5. Se inicia sesión en el sistema y se muestra el contenido de su perfil.

- **Escenarios alternativos:**

No se ha iniciado sesión correctamente y se muestra un mensaje al usuario.

7.2.3. Visualizar perfiles

CU1 → Visualizar perfiles de otros usuarios

- **Contexto de uso::**

Cuando un usuario quiera visualizar el perfil de otro usuario para ver su información pública.

- **Precondiciones y activación**

El usuario está conectado a la aplicación y al perfil de algún otro usuario.

- **Garantías de éxito / Postcondición:**

El usuario visualiza la información pública de otro usuario.

- **Escenario principal:**

1. El usuario se conecta al perfil de otro usuario.
2. El sistema muestra el perfil del otro usuario así como su información marcada como pública.

- **Escenarios alternativos:**

No se muestra el perfil del otro usuario ya sea por un error de la aplicación o porque este no exista y se muestra un mensaje de error al usuario.

7.2.4. Editar del perfil

CU1 → Editar nuestro perfil de forma modular

- **Contexto de uso:**

Cuando un usuario quiera editar la información de su perfil.

- **Precondiciones y activación:**

El usuario está conectado a la aplicación y al editor de su perfil.

- **Garantías de éxito / Postcondición:**

El usuario visualiza los cambios en su perfil.

- **Escenario principal:**

1. El usuario se conecta a su perfil.
2. Pulsa el botón de editar perfil.
3. Seleccionar los campos a editar
4. Editar los campos seleccionados
5. Pulsar en el botón de guardar los cambios

- **Escenarios alternativos:**

No se muestra el perfil del usuario con los cambios realizados y se muestra un mensaje de error al usuario.

7.2.5. Buscar información multimedia

CU1 → Buscar información sobre películas o series

- **Contexto de uso:**

Cuando un usuario quiera buscar información acerca de películas o series en base a distintos filtros.

- **Precondiciones y activación:**

El usuario está conectado a la aplicación y en el formulario de búsqueda.

- **Garantías de éxito / Postcondición:**

El usuario realiza la búsqueda en el sistema y obtiene los resultados.

- **Escenario principal:**

1. El usuario pulsa la opción para buscar.
2. El sistema muestra el formulario para buscar.
3. El usuario introduce los filtros de búsqueda.
4. El usuario pulsa la opción de buscar.
5. Se realiza la búsqueda y se muestra el resultado.

- **Escenarios alternativos:**

No se ha realizado la búsqueda correctamente y se muestra un mensaje al usuario.

7.2.6. Interactuar con otros usuarios

CU1 → Interactuar con otros usuarios mediante foros específicos que se pueden encontrar en cada película y serie o mediante el foro general de la aplicación.

- **Contexto de uso:**

Cuando un usuario quiera interactuar con otros usuarios interesados en cualquier película o serie.

- **Precondiciones y activación:**

El usuario está conectado a la aplicación y en alguno de los foros.

- **Garantías de éxito / Postcondición:**

El usuario puede ver las discusiones activas en los foros o enviar un mensaje participar en ellas.

- **Escenario principal:**

1. El usuario se conecta a alguno de los foros.
2. El sistema muestra el foro con los mensajes de los usuarios.
3. El usuario pulsa en la opción de escribir mensaje.
4. El usuario rellena el formulario de envío de mensaje.
5. El usuario pulsa la opción de enviar mensaje.
6. Se envía el mensaje y se muestra el mensaje en el foro.

- **Escenarios alternativos:**

No se ha cargado el foro correctamente o no se ha enviado el mensaje correctamente y se muestra un mensaje al usuario.

7.2.7. Etiquetado del contenido

CU1 → Marcar una película o serie como contenido me ha gustado, no me ha gustado, he visto o no he visto

- **Contexto de uso:**

Cuando un usuario quiera etiquetar una película de alguna manera.

- **Precondiciones y activación:**

El usuario está conectado a la aplicación y en alguna entrada de película o serie.

- **Garantías de éxito / Postcondición:**

El usuario etiqueta la película o serie deseada y se añade a su perfil.

- **Escenario principal:**

1. El usuario se conecta a la entrada de alguna serie o película.
2. El usuario pulsa en la opción de etiquetado deseada.
3. Se etiqueta el contenido.

- **Escenarios alternativos:**

No se ha etiquetado el contenido correctamente y se muestra un mensaje al usuario.

7.2.8. Listado del contenido

CU1 → Incluir una película o serie a una lista creada por el usuario

- **Contexto de uso:**

Cuando un usuario quiera crear una lista con un nombre identificativo y varias referencias a distinto contenido en su interior.

- **Precondiciones y activación:**

El usuario está conectado a la aplicación y a su perfil.

- **Garantías de éxito / Postcondición:**

El usuario añade un contenido a una lista creada previamente por el mismo.

- **Escenario principal:**

1. El usuario se conecta a su perfil.
2. En caso de no existir, pulsa en el botón de añadir lista.
 - 2.1. Le da un nombre a la lista creada.
 - 2.2. Le da al botón de guardar lista
3. El usuario añade las películas o series que desee a la lista creada.
4. Le da al botón de guardar cambios.

- **Escenarios alternativos:**

No se crea la lista o no se añaden los cambios de forma correcta y se le muestra un mensaje de error al usuario.

7.2.9. Valorar el contenido con comentarios

CU1 → Añadir una valoración a una película o serie

- **Contexto de uso:**

Cuando un usuario quiera dar su opinión acerca de una película o serie.

- **Precondiciones y activación:**

El usuario está conectado a la aplicación y a la entrada de la película o serie a valorar.

- **Garantías de éxito / Postcondición:**

El usuario añade una valoración al contenido y esta se muestra en la entrada de la película o serie.

- **Escenario principal:**

1. El usuario se conecta a la entrada de un contenido.
2. Le da al botón de añadir valoración.
3. Rellena el formulario de la valoración.
4. Le da al botón de guardar valoración.

- **Escenarios alternativos:**

No se añade la valoración a la entrada de la película o serie y se muestra un mensaje de error al usuario.

7.2.10. Interactuar con perfiles

CU1 → Interactuar con perfiles de otros usuarios de forma que puedas seguirlos o bloquearlos

- **Contexto de uso:**

Cuando un usuario quiera seguir o bloquear a otro usuario.

- **Precondiciones y activación:**

El usuario está conectado a la aplicación y al perfil de otro usuario.

- **Garantías de éxito / Postcondición:**

El usuario añade al otro usuario a su lista de seguidos o bloqueados.

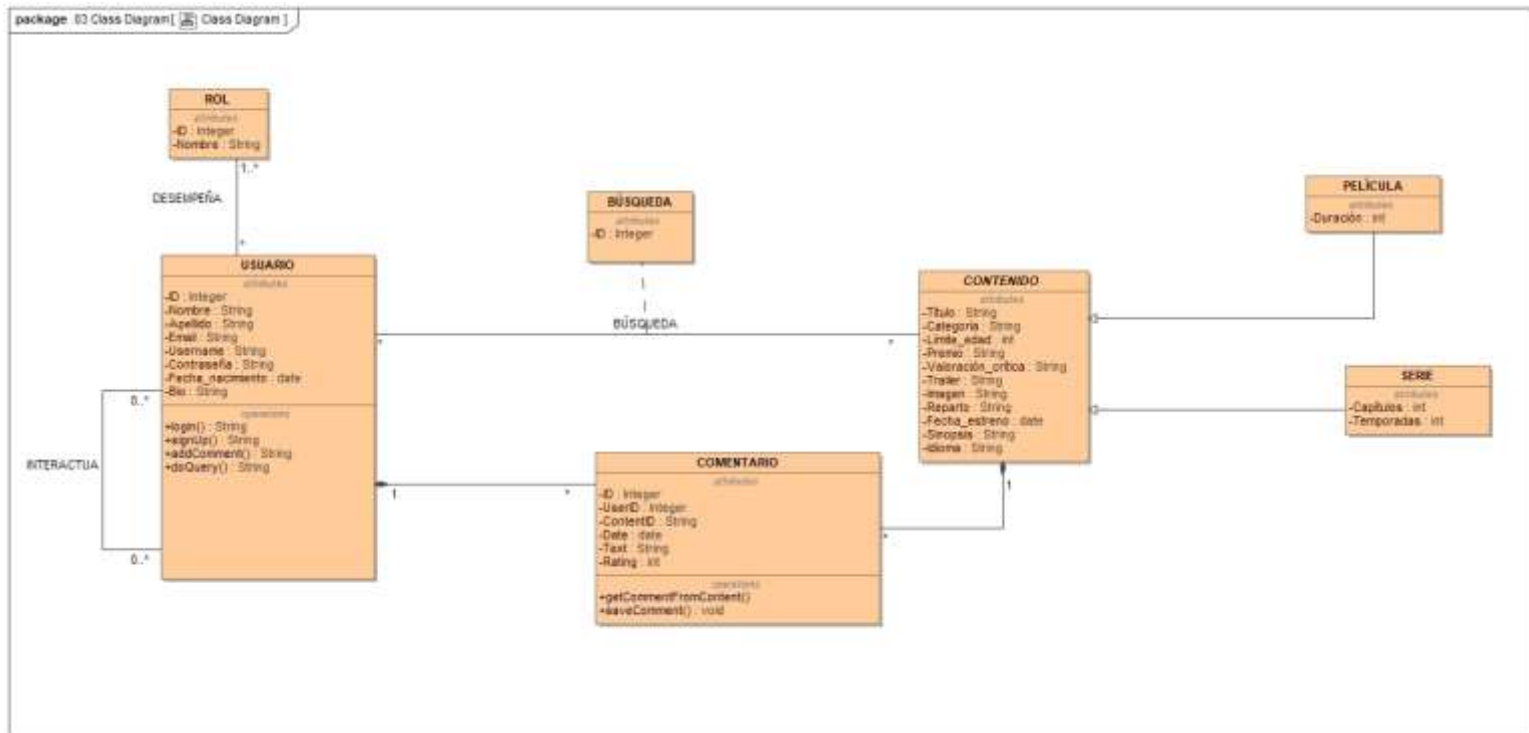
- **Escenario principal:**

1. El usuario se conecta al perfil de otro usuario.
2. Le da al botón de seguir o bloquear.

- **Escenarios alternativos:**

No se añade al otro usuario a la lista de usuarios seguidos o bloqueados del propio usuario y se muestra un mensaje de error.

8. Modelo de Dominio



9. Pruebas

Para hacer las pruebas hemos utilizado tanto Junit como Mockito. Hemos hecho ocho pruebas, tres para la clase Actor y otras tres para la clase User y dos para la clase Comment, seis de ellas de JUnit y dos pruebas de Mockito.

9.1. ActorTest

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import static org.mockito.Mockito.*;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import com.erole.moviErole.APIQuery.model.contentQuery.Actor;

class ActorTest {

    Actor actor;

    @BeforeEach
    public void inic() {
        actor = new Actor();
    }

    @AfterEach
    public void terminate() {
        actor = null;
    }

    @Test //Un actor sin argumentos no contiene información
    void inicialmenteEstaVacio() {
        assertEquals(actor.getName(), null);
    }

    @Test //Tras insertar datos, el usuario contiene información
    void contieneInformacion() {
        actor = new Actor("id", "image", "name", "asCharacter");
        assertEquals(false, actor.getId().isEmpty());
        assertEquals(false, actor.getImage().isEmpty());
        assertEquals(false, actor.getName().isEmpty());
        assertEquals(false, actor.getName().isEmpty());
    }

    @Test //Tras cambiar los atributos a unos diferentes, debería ser distintos
    al actor original
    void setTest() {
        Actor actor1 = mock(Actor.class); //Creamos los mock de dos objetos
        Actor actor2 = mock(Actor.class);

        String id1 = "1";
        String id2 = "2";

        when(actor1.getId()).thenReturn(id1); //Asignamos los id de los
    actores
}
```

```

when(actor2.getId()).thenReturn(id2);
actor1.setId(id2); //Cambiamos el id del primer actor

String img1 = "1";
String img2 = "2";

when(actor1.getImage()).thenReturn(img1); //Asignamos las imagenes
when(actor2.getImage()).thenReturn(img2);
actor1.setImage(img2); //Cambiamos la imagen del primer actor

String name1 = "1";
String name2 = "2";

when(actor1.getName()).thenReturn(name1); //Asignamos los nombres
when(actor2.getName()).thenReturn(name2);
actor1.setName(name2); //Cambiamos el nombre del primer actor

String char1 = "1";
String char2 = "2";

when(actor1.getAsCharacter()).thenReturn(char1); //Asigna los papeles
when(actor2.getAsCharacter()).thenReturn(char2);
actor1.setAsCharacter(char2); //Cambia el papel del primer actor

assertEquals(false, actor1.getId().equals(actor2.getId()));
//Comprueba el id
assertEquals(false, actor1.getImage().equals(actor2.getImage()));
//Comprueba la imagen
assertEquals(false, actor1.getName().equals(actor2.getName()));
//Comprueba el nombre
assertEquals(false,
actor1.getAsCharacter().equals(actor2.getAsCharacter())); //Comprueba el personaje
    }

}

```

9.2. UserTest

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import static org.mockito.Mockito.*;
import java.util.Collection;
import java.util.Date;
import java.util.LinkedList;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import com.erole.moviErole.model.Role;
import com.erole.moviErole.model.User;

class UserTest {

    User user;

    @BeforeEach
    public void inic() {
        user = new User();
    }

    @AfterEach
    public void terminate() {
        user = null;
    }

    @Test //Un usuario sin argumentos no contiene información
    void inicialmenteEstaVacio() {
        assertEquals(user.getName(), null);
    }

    @Test //Tras insertar datos, el usuario contiene información
    void contieneInformacion() {
        @SuppressWarnings("deprecation")
        Date date = new Date(1,1,1);
        Collection<Role> roles;
        Role rol = new Role("Rol");
        roles = new LinkedList<Role>();
        roles.add(rol);
        user = new User("email", "userName", "password", "name", "surname",
date,
                        "bio", roles);
        /*public User(String email, String userName, String password, String
name, String surname, Date birthdate,
                        String bio, Collection<Role> roles)*/
        assertEquals(false, user.getEmail().isEmpty());
        assertEquals(false, user.getName().isEmpty());
        assertEquals(false, user.getPassword().isEmpty());
        assertEquals(false, user.getUserName().isEmpty());
        assertEquals(false, user.getSurname().isEmpty());
        assertEquals(false, user.getBirthdate().equals(null));
        assertEquals(false, user.getBio().isEmpty());
        assertEquals(false, user.getRoles().isEmpty());
    }

    @Test
```

```

public void setTest(){
    User user1 = mock(User.class); //Creamos mock de dos objetos usuario
    User user2 = mock(User.class);

    Integer id1 = 0;
    Integer id2 = 1;

    when(user1.getId()).thenReturn(id1); //Asignamos un identificador
    when(user2.getId()).thenReturn(id2);
    user1.setId(id2); //Cambiamos el identificador del primer usuario al
del segundo

    Collection<Role> roles1;
    Role rol1 = new Role("Rol1");
    roles1 = new LinkedList<Role>();
    roles1.add(rol1);

    Collection<Role> roles2;
    Role rol2 = new Role("Rol2");
    roles2 = new LinkedList<Role>();
    roles2.add(rol2);

    when(user1.getRoles()).thenReturn(roles1); //Asignamos los roles
    when(user2.getRoles()).thenReturn(roles2);
    user1.setRoles(roles2); //Cambiamos el rol del primer usuario

    @SuppressWarnings("deprecation")
    Date date1 = new Date(1,1,1);
    @SuppressWarnings("deprecation")
    Date date2 = new Date(2,2,2);

    when(user1.getBirthdate()).thenReturn(date1); //Asignamos las fechas
    when(user2.getBirthdate()).thenReturn(date2);
    user1.setBirthdate(date2); //Cambiamos la fecha de nacimiento del
primer usuario

    String email1 = "1";
    String email2 = "2";

    when(user1.getEmail()).thenReturn(email1); //Asignamos los emails
    when(user2.getEmail()).thenReturn(email2);
    user2.setEmail(email2); //Cambiamos el email del primer usuario

    String uName1 = "1";
    String uName2 = "2";

    when(user1.getUserName()).thenReturn(uName1); //Asignamos los nombres
de usuario
    when(user2.getUserName()).thenReturn(uName2);
    user2.setUserName(uName2); //Cambiamos el nombre de usuario del
primer usuario

    String pass1 = "1";
    String pass2 = "2";

    when(user1.getPassword()).thenReturn(pass1); //Asignamos la
contrasena
    when(user2.getPassword()).thenReturn(pass2);

```

```

        user2.setPassword(pass2); //Cambiamos la contraseña del primer
usuario

        String name1 = "1";
        String name2 = "2";

        when(user1.getName()).thenReturn(name1); //Asignamos los nombres
        when(user2.getName()).thenReturn(name2);
        user2.setPassword(name2); //Cambiamos el nombre del primer usuario

        String sur1 = "1";
        String sur2 = "2";

        when(user1.getSurname()).thenReturn(sur1); //Asignamos el apellido
        when(user2.getSurname()).thenReturn(sur2);
        user2.setSurname(sur2); //Cambiamos el apellido del primer usuario

        String bio1 = "1";
        String bio2 = "2";

        when(user1.getBio()).thenReturn(bio1); //Asignamos la bio
        when(user2.getBio()).thenReturn(bio2);
        user2.setPassword(bio2); //Cambiamos la bio del primer usuario

        assertEquals(false, user1.getId().equals(user2.getId())); //Comprueba
el id
        assertEquals(false, user1.getRoles().equals(user2.getRoles()));
//Comprueba los roles
        assertEquals(false,
user1.getBirthdate().equals(user2.getBirthdate())); //Comprueba la fecha
        assertEquals(false, user1.getEmail().equals(user2.getEmail()));
//Comprueba el email
        assertEquals(false, user1.getUserName().equals(user2.getUserName()));
//Comprueba el nombre de usuario
        assertEquals(false, user1.getPassword().equals(user2.getPassword()));
//Comprueba la contraseña
        assertEquals(false, user1.getName().equals(user2.getName()));
//Comprueba el nombre
        assertEquals(false, user1.getSurname().equals(user2.getSurname()));
//Comprueba el usuario
        assertEquals(false, user1.getBio().equals(user2.getBio()));
//Comprueba la bio

    }

}

```

9.3. CommentTest

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import com.erole.moviErole.model.Comment;
import com.erole.moviErole.model.User;

public class CommentTest {

    Comment comment;

    @BeforeEach
    public void inic() {
        comment = new Comment();
    }

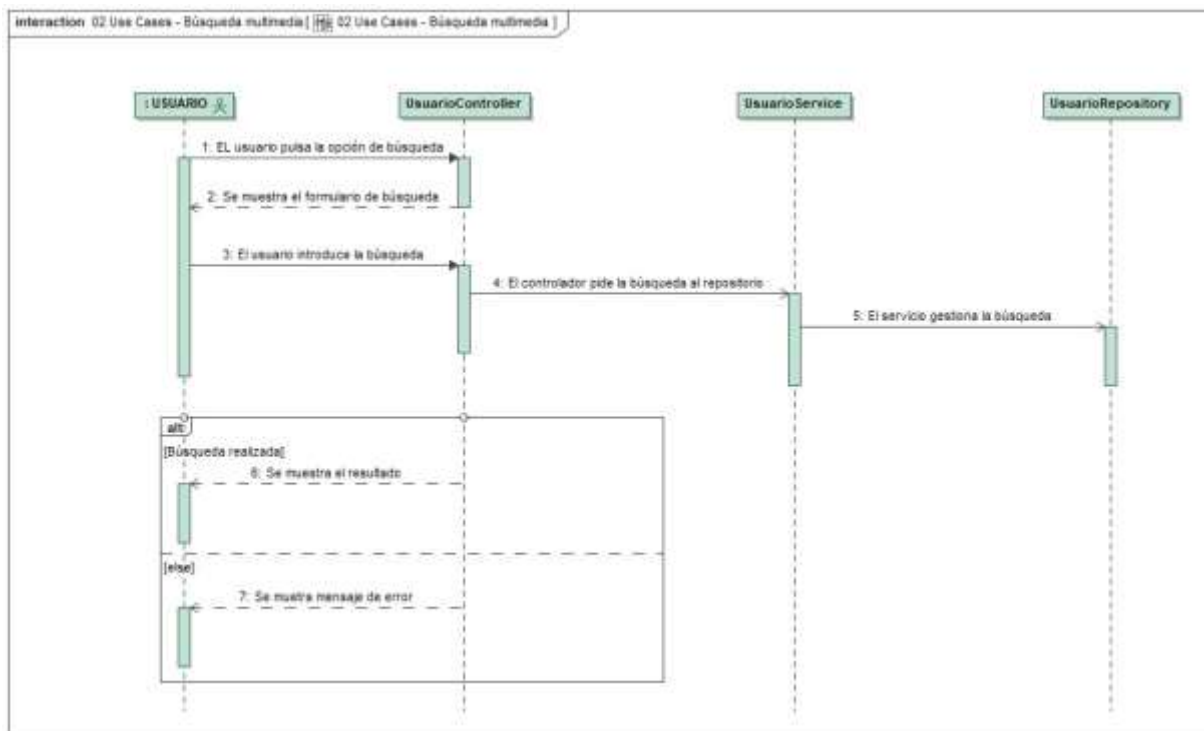
    @AfterEach
    public void terminate() {
        comment = null;
    }

    @Test //Un actor sin argumentos no contiene información
    void inicialmenteEstaVacio() {
        assertEquals(comment.getId(), null);
    }

    @Test //Tras insertar datos, el usuario contiene información
    void contieneInformacion() {
        User user = new User();
        comment = new Comment(user, "1", "comment", 0); //Creamos un objeto
        //de tipo comentario
        assertEquals(false, comment.getUser() == null);
        assertEquals(false, comment.getContentId() == null);
        assertEquals(false, comment.getText() == null);
    }
}
```

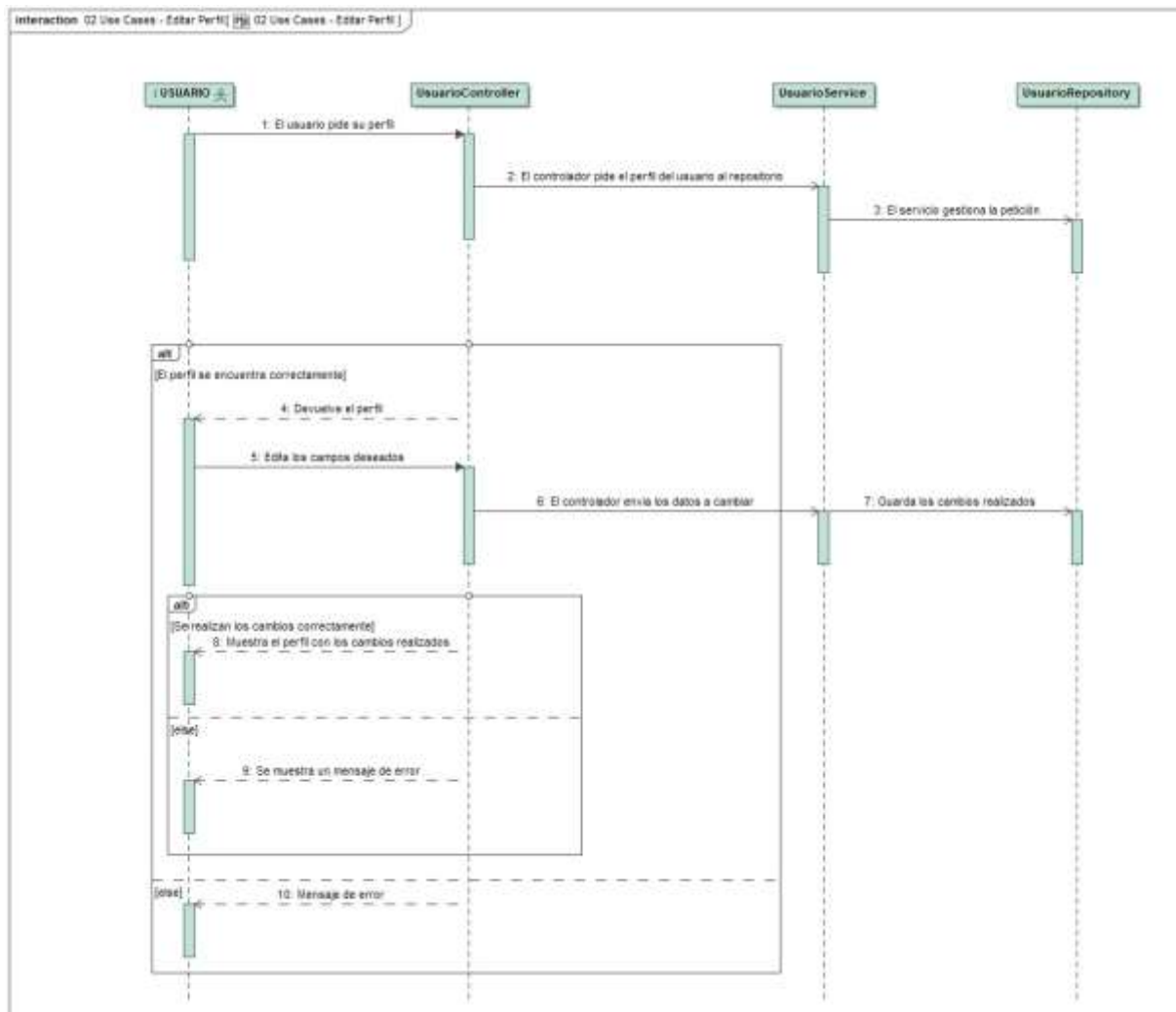

10. Diagramas de secuencia

10.1. Búsqueda multimedia



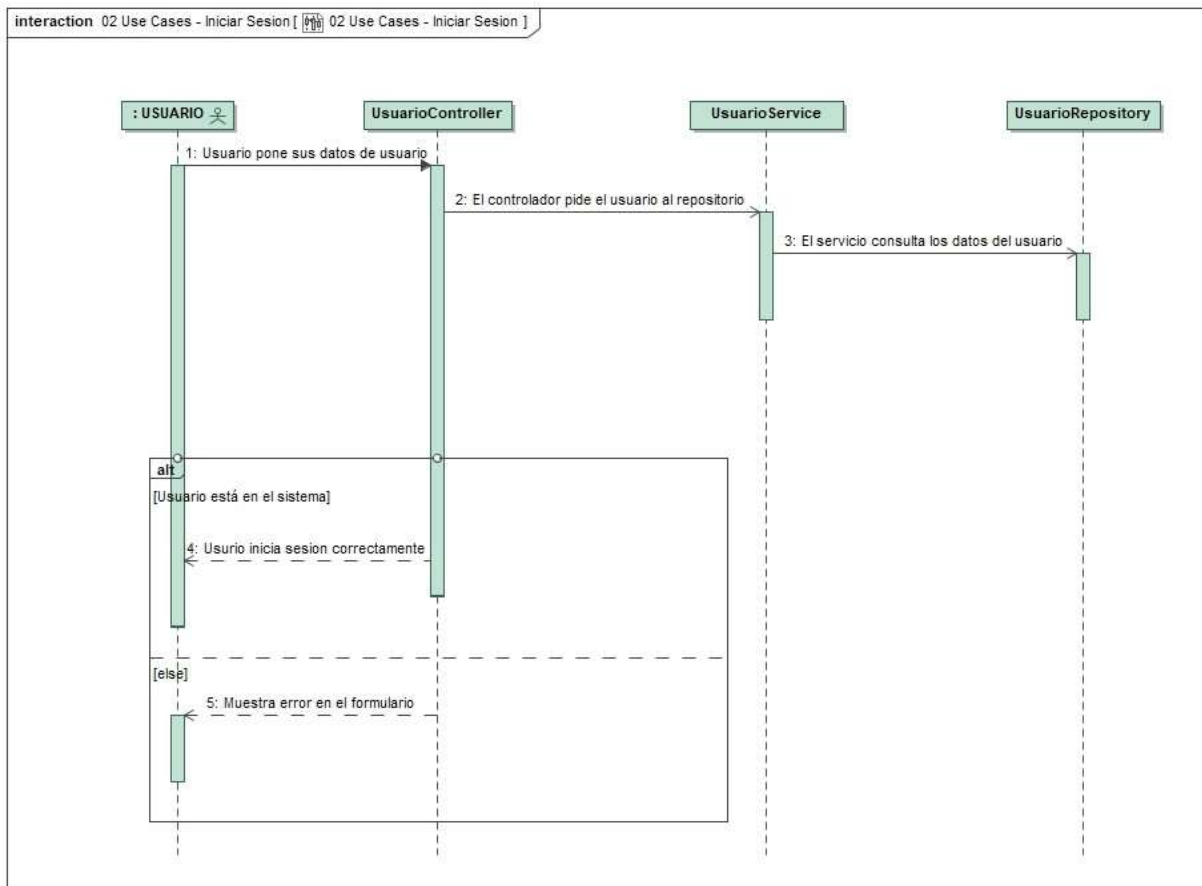
Este es el diagrama de secuencia de la búsqueda multimedia, el primer paso es que el usuario pulse la sección de búsqueda, a continuación, se mostrará el formulario de búsqueda, una vez introducida la búsqueda, el controlador pedirá la búsqueda al repositorio. Una vez realizado, el servicio gestionará la búsqueda. Si no se realizase la búsqueda con éxito se mostraría un mensaje de error.

10.2. Editar perfil



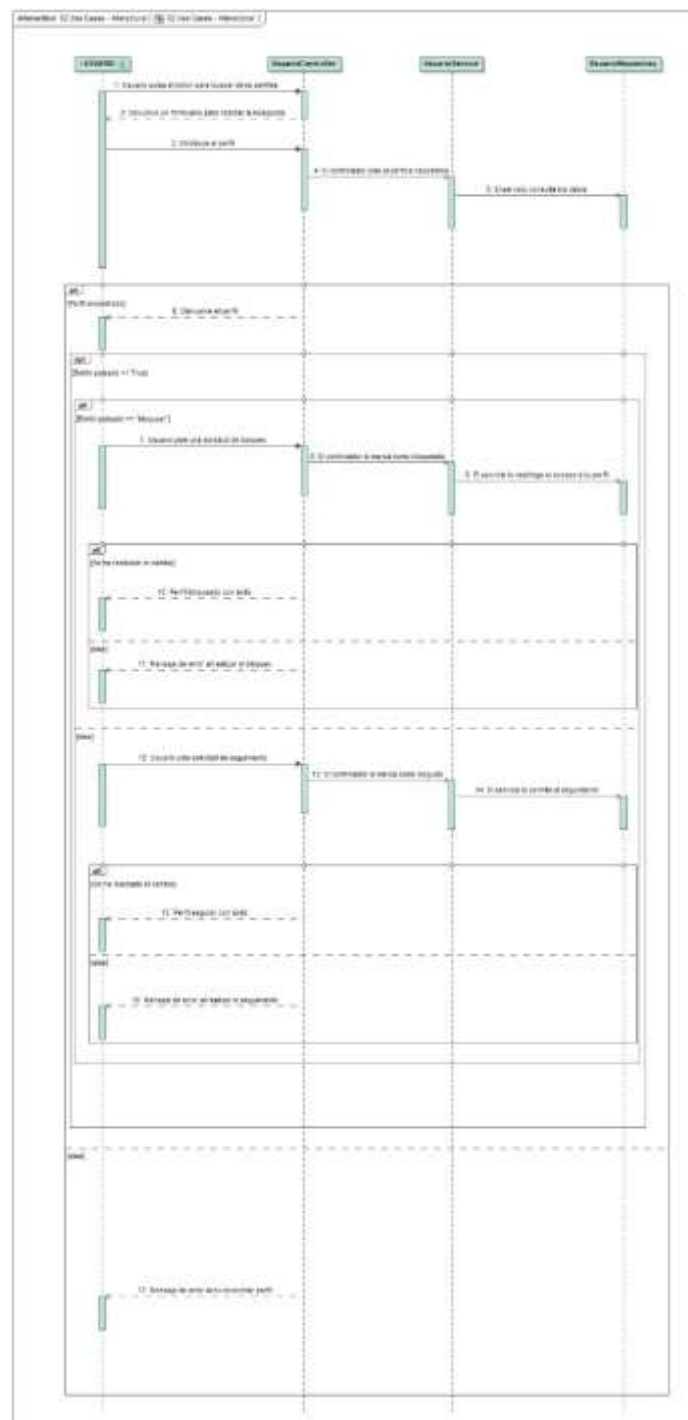
En este caso lo primero que se hará es pedir al controlador el su perfil, este, pedirá el perfil al repositorio, el servicio gestionará la petición. En el caso de que el perfil se encuentre se devolverá el perfil para editar los campos correspondientes y deseados, luego el controlador enviara los datos a cambiar y guardar los cambios ya realizados, en el caso de que no salga bien se mostrará un mensaje de error, tanto como cuando no se guardan los cambios como cuando no se encuentra el perfil.

10.3. Iniciar sesión



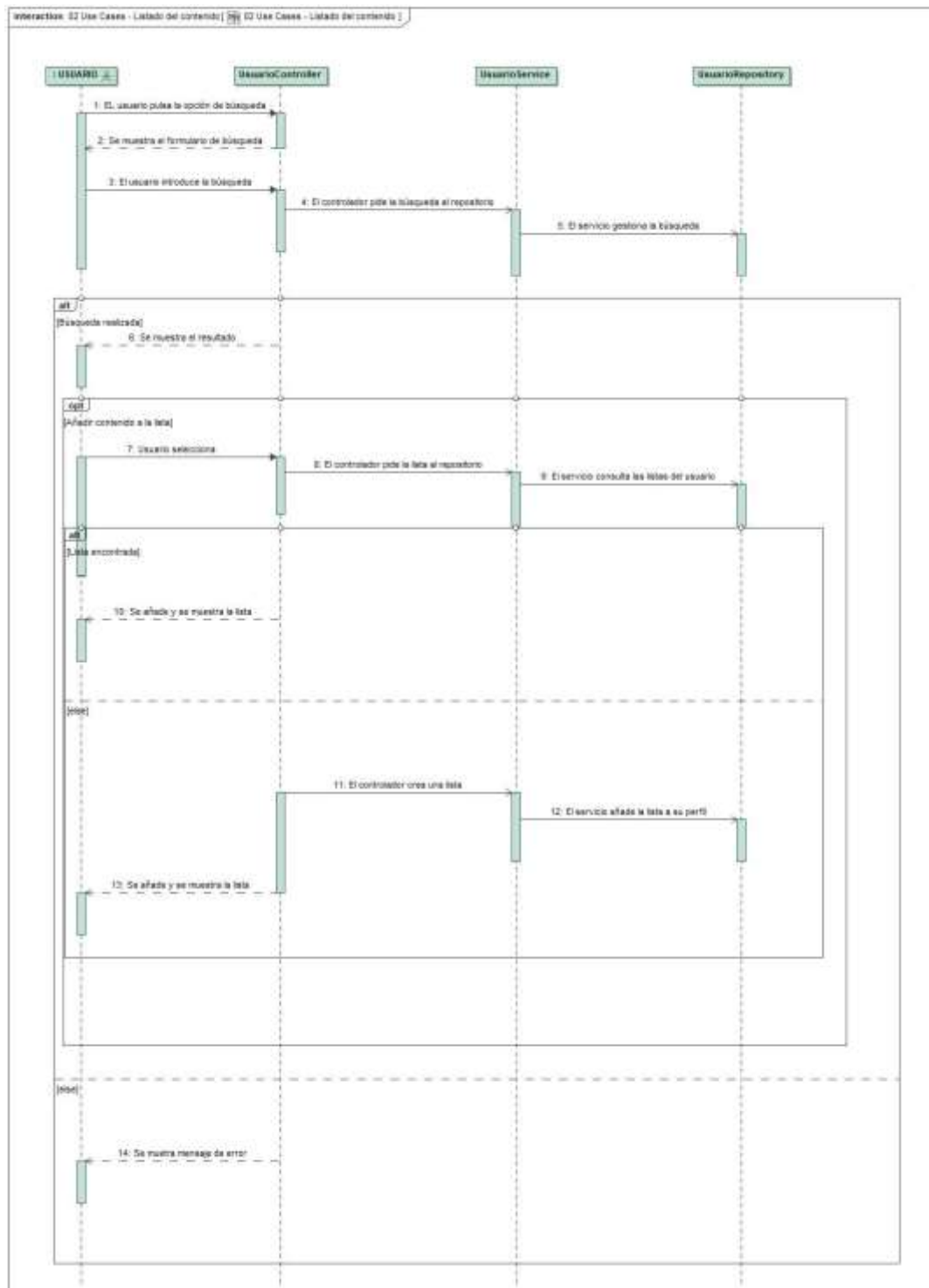
Primeramente, el usuario introduce sus datos en el formulario de inicio de sesión, seguidamente, el controlador pedirá el usuario al repositorio. El servicio consultará los datos del usuario, si no está bien se mostrará un error en el formulario.

10.4. Interactuar



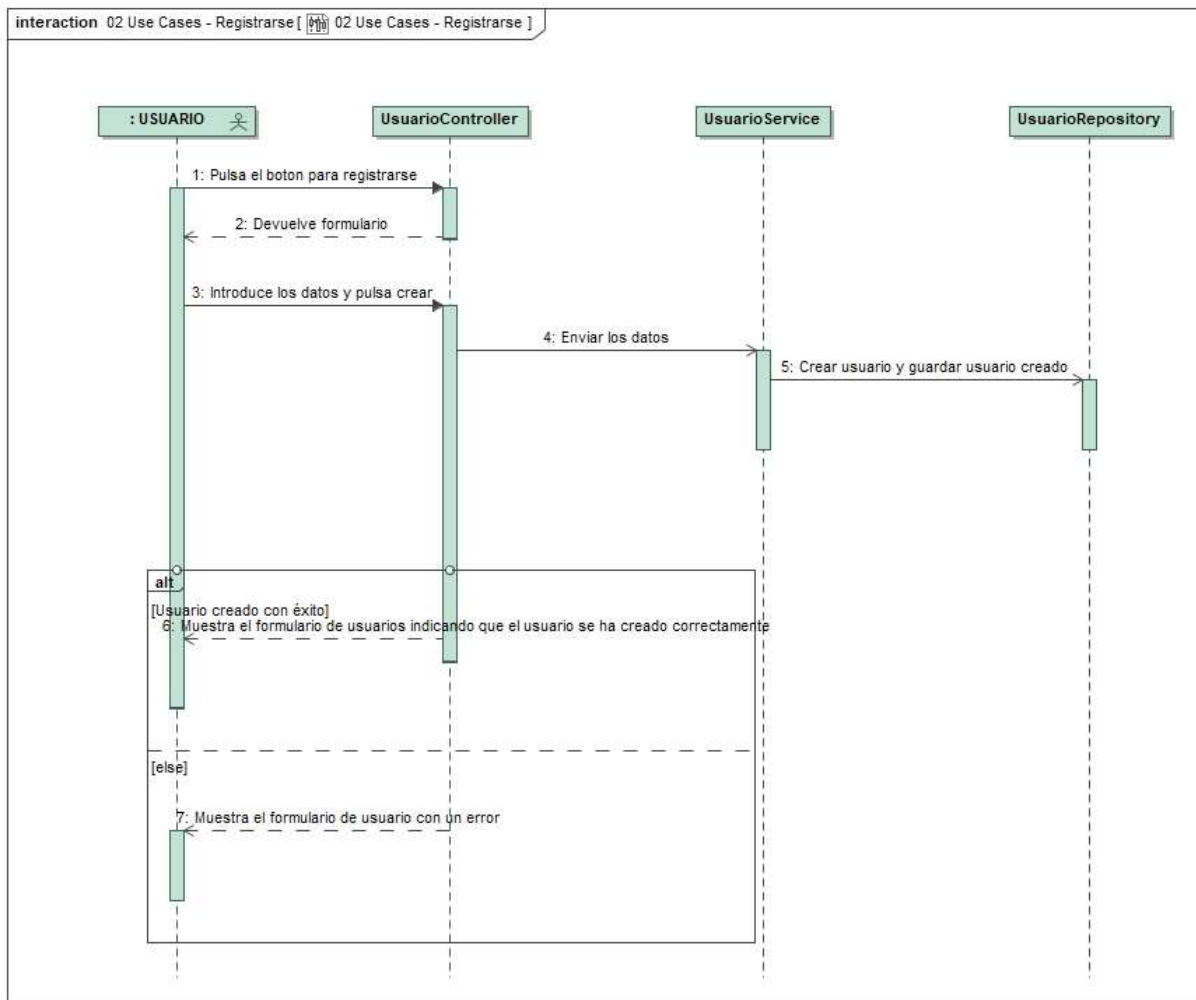
Lo primero de todo es que el usuario realice el formulario para buscar perfiles y que este se envíe al controlador. Una vez el controlador lo reciba este pedirá el perfil al repositorio y el servicio consultará los datos. En el caso de que el perfil exista se devolverá el perfil y se darán las opciones de visualizar, seguir y bloquear al perfil. En el caso de bloquear y seguir se mostrará un mensaje de éxito a los usuarios para saber que se ha bloqueado o seguido con éxito, si no, se mostrará un error.

10.5. Listado del contenido



Antes de poder listar habrá que realizar la búsqueda como ya hemos explicado anteriormente, una vez realizada con éxito el usuario podrá seleccionar el contenido y la opción de añadir a una lista, este escribirá el nombre de la lista, si esta existe se añadirá a la lista correspondiente, si no existe se creará primero y luego se añadirá. En el caso de que haya algún error se mostrará un mensaje correspondiente.

10.6. Registrarse



El usuario pulsará la opción para registrarse, a continuación, se le enviará un formulario a rellenar con sus datos y credenciales, una vez realizado este se enviará al repositorio, pasando antes por el servicio, que lo creará y lo guardará. Si todo ha salido correctamente se mostrará el perfil ya creado, si no se mostrará un mensaje de error.

11. Herramientas Software

- Whatsapp
- Discord
- GitHub
- Trello
- Herramientas de Google (Drive, Documentos, etc.)
- VSCode
- Eclipse IDE
- SQL Developer
- Data Modeler
- Herramientas de Office 365 (Microsoft Word)
- MagicDraw

11.1. Enlace a GitHub

<https://github.com/marinasayago/Erole.git>