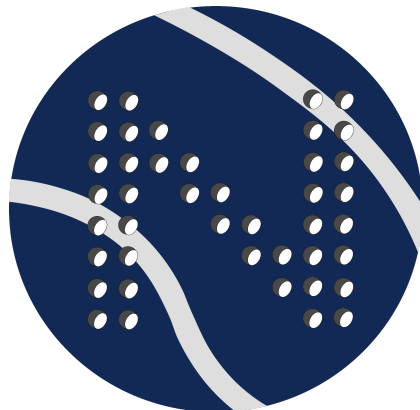
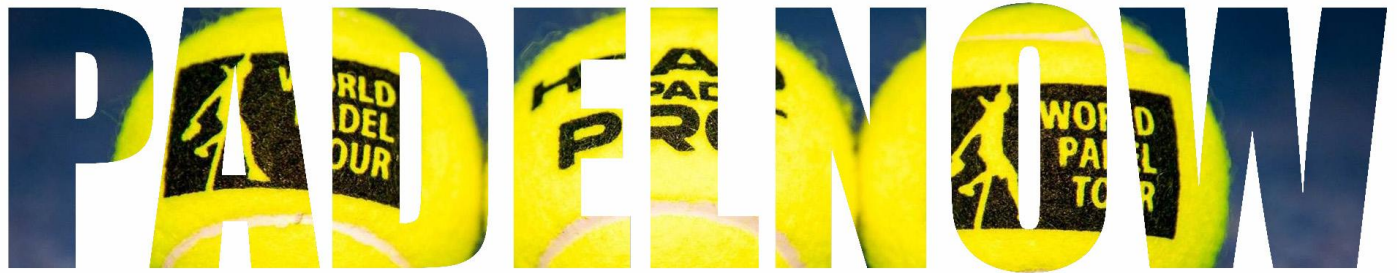


**Grupo 2 - Doble Grado Matemáticas + Ingeniería
Informática**
Introducción a la Ingeniería del Software



Padel Now

Componentes del grupo

<https://github.com/Matematicas-Ingenieria-Informatica-UMA/PadelNow>

Alberto Sánchez Muñoz albetrosanchez@uma.es

María José García Tobaruela mjgtobaruela@uma.es

Manuel Castillo Sancho 0610693837@uma.es

Mario Quiñones Madrona mario.qm@uma.es

Matías Ramírez Durán 0610705247@uma.es

Arturo Aguilera González arturoaguilera@uma.es

Alejandro Rey Leyva alereyleyva@uma.es

Ignacio Ávila Reyes naxetee.a.r@uma.es

Javier Guerrero Rico 061981555x@uma.es

1 - Introducción

El software a desarrollar tiene como objetivo principal facilitar al usuario conocer resultados, horarios, clasificaciones, estadísticas y jugadores sobre las diferentes competiciones que existen hoy día en el pádel profesional.

Hemos decidido realizar este proyecto porque actualmente no hay aplicaciones punteras enfocadas exclusivamente a este deporte, que, además, está experimentando una evolución y crecimiento.

2 - Roles

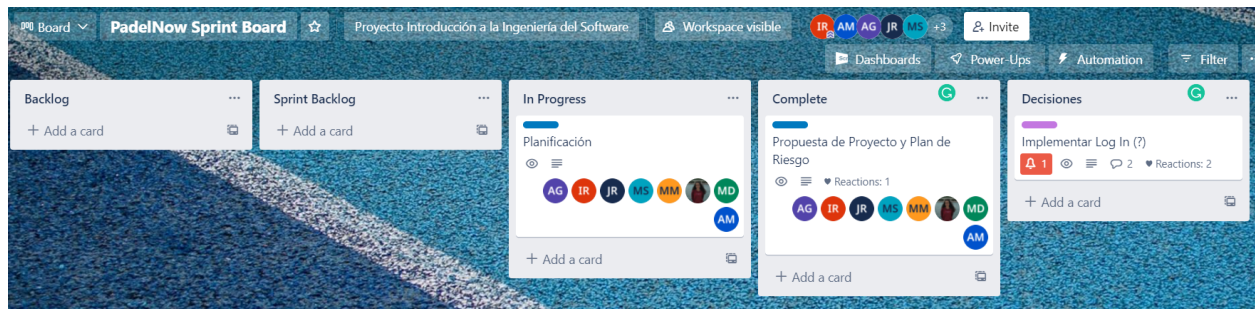
Alberto Sánchez Muñoz:	Diseñador, Desarrollador (Frontend)
María José García Tobaruela:	Tester, Desarrollador (Backend)
Manuel Castillo Sancho:	Diseñador, Desarrollador (Backend)
Mario Quiñones Madrona:	Desarrollador (Frontend), Tester
Matías Ramírez Durán:	Tester, Diseñador
Arturo Aguilera González:	Scrum Master, Desarrollador (Backend)
Alejandro Rey Leyva:	Desarrollador (Frontend y Backend)
Ignacio Ávila Reyes:	Desarrollador (Frontend), Tester
Javier Guerrero Rico:	Tester, Diseñador

3 - Gestión de Riesgo

Tipo de Riesgo	Riesgo	Descripción	Probabilidad	Efectos del riesgo	Estrategia para mitigarlo
Base de Datos	Amplia base de datos inicial	Gran cantidad de datos que incluir inicialmente.	Alta	Tolerable	Búsqueda de una API pública de empresas especializadas que nos muestran los datos
Base de Datos	Difícil automatización de resultados ya finalizados	Uno de los mayores problemas es actualizar la base de datos automáticamente tras cada partido	Alta	Serio	Contratos
Competencia	Creación de una aplicación oficial	Es probable que con el auge del pádel, empresas o periódicos digitales desarrollen un software similar	Media	Serio	Adelantarse a la competencia
Herramientas	Escasa disponibilidad de medios	Como estudiantes sin posibilidad de invertir dinero en el proyecto tenemos medios muy restringidos para lograr los objetivos	Alta	Tolerable	Buscar herramientas gratuitas
Personal	Formación del personal insuficiente	Dificultades relacionadas al desconocimiento del uso de una herramienta por parte del personal	Medio	Tolerable	Aprender sobre la marcha, trabajar en superar esas dificultades
Personal	Falta de experiencia	Es el primer trabajo que hacemos como un equipo en este entorno	Alta	Tolerable	Ayudarnos entre los miembros del grupo

4 - Planificación

El modelo de proceso software que hemos elegido ha sido la metodología Scrum, ya que es una gran metodología para los trabajos en equipo y sus características son adecuadas para el desarrollo software.



Propuesta de Proyecto y Plan de Riesgo

in list [Complete](#)

Members

AM

AG

IR

JR

MS

MM

MD

+

Labels

Documentación

+

1

±

[Description](#)
[Edit](#)

Documento con la propuesta del proyecto. Contiene una descripción breve de la aplicación web que vamos a desarrollar, los roles, un apartado para la gestión de riesgo y para las herramientas software usadas.

[Activity](#)
[Show details](#)

IR

Write a comment...

IR

Ignacio Ávila Reyes a minute ago

Buen trabajo!! 🙌 @card

[Edit](#)
[Delete](#)

Power-Ups

GitHub

Adjunte ramas, confirmaciones, incidencias y solicitudes de extracción a las tarjetas para ver el...

[Configuración](#)

Votar

Otorgue poder a las personas y permita que los usuarios voten sobre las tarjetas.

[Configuración](#)

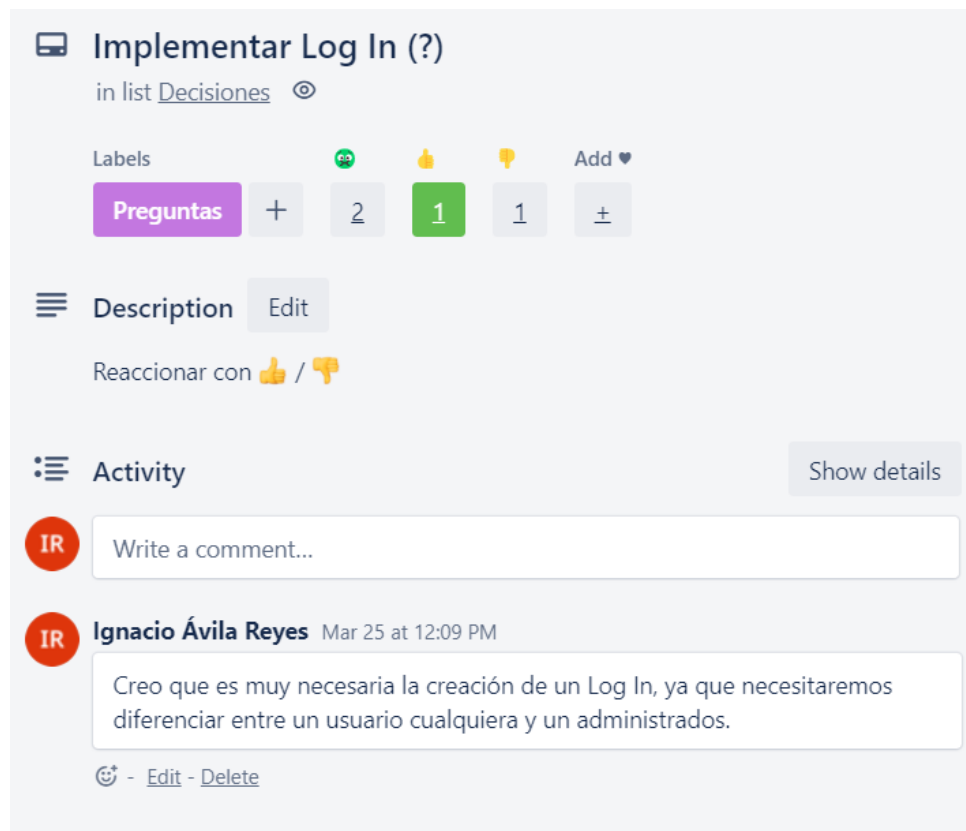
Card Reactions

Express yourself in Trello with Card Reactions - use emoji to vote, celebrate, and more!

[Configuración](#)



Vamos a usar el power-up de GitHub, cuya característica principal es que permite visualizar todo lo que ocurre en GitHub desde Trello, es decir, nos aporta una visión más amplia del estado en el que se encuentra el proyecto. También hemos añadido power-ups para crear votaciones, que nos ayudará a tomar decisiones de diseño, y un power-up que añade la posibilidad de responder a un mensaje con una reacción corta, es decir un emoji.



5 - Requisitos

5.1 - Requisitos Funcionales

RF1 - Log In

Como administrador
quiero una posibilidad de diferenciarme de cualquier usuario
para poder realizar acciones exclusivas.

PRUEBAS ACEPTACIÓN

- Si se introducen credenciales correctas, se redirecciona al dashboard de administrador.
- Si se introducen credenciales incorrectas, se muestra un mensaje de error.

RF2 - Menú de navegación

Como usuario
quiero un menú/barra de navegación de fácil uso y comprensión
para acceder rápidamente a contenidos.

PRUEBAS ACEPTACIÓN

- Deben mostrarse todos los botones de los diferentes contenidos de la aplicación.
- Cuando se hace click en un contenido, se debe redirigir a la ruta asociada y mostrar el contenido.

RF3 - Buscador

Como usuario
quiero una herramienta que filtre contenidos por texto
para poder acceder a la información deseada rápidamente.

PRUEBAS ACEPTACIÓN

- Debe ser un input que permite escribir al usuario.
- Cuando el usuario escribe debe aparecer en tiempo real todos los contenidos de la app relacionados con el input introducido.
- Si no hay resultados debe informar al usuario.
- Si se hace click en uno de los resultados, se debe redirigir a la vista del contenido elegido.

RF4 - Consultar Partidos

Como usuario
quiero tener acceso a una lista
para ver todos los partidos jugados o por jugar.

PRUEBAS ACEPTACIÓN

- Inicialmente deben aparecer todos los partidos del día actual, por orden cronológico.
- Se puede seleccionar otro día y consultar dichos partidos.
- Se puede filtrar también por torneo. (opcional)
- Si se hace click en uno de los partidos, se debe redirigir a la vista del partido elegido.

RF5 - Consultar Torneos

Como usuario
quiero tener acceso a una lista
para ver todos los torneos disponibles.

PRUEBAS ACEPTACIÓN

- Debe mostrar una lista con todos los torneos.
- Si se hace click en una de los torneos, se debe redirigir a la vista del torneo elegido.

RF6 - Consultar Circuitos

Como usuario
quiero tener acceso a una lista
para ver todas los circuitos disponibles.

PRUEBAS ACEPTACIÓN

- Debe mostrar una lista con todos los circuitos.
- Si se hace click en una de los circuitos, se debe redirigir a la vista del circuito elegido.

RF 7 - Consultar Parejas

Como usuario
quiero tener acceso a una lista
para ver todas las parejas disponibles.

PRUEBAS ACEPTACIÓN

- Debe mostrar una lista con todas las parejas actuales.
- Si se hace click en una de las parejas, se debe redirigir a la vista de la pareja elegida.
- Se puede ordenar por ranking, torneos ganados, estadísticas varias, etc... (opcional)

RF8 - Consultar jugadores

Como usuario
quiero tener acceso a una lista
para ver todos los jugadores disponibles.

PRUEBAS ACEPTACIÓN

- Deben mostrar una lista con todos los jugadores actuales.
- Si se hace click en uno de los jugadores, se despliega el perfil del jugador elegido.
- Se puede filtrar por sus estadísticas, aspectos personales, etc... (opcional)

<p style="text-align: center;">RF 9 - Ver Partido</p> <p>Como usuario quiero poder acceder a un partido concreto para consultar información más detallada sobre este.</p> <p style="text-align: center;">PRUEBAS ACEPTACIÓN</p> <ul style="list-style-type: none"> • Debe aparecer la fecha del partido, las parejas implicadas, el resultado si ha finalizado, fase de la competición, y toda la información establecida. (CHECK) 	<p style="text-align: center;">RF 10 - Ver Torneo</p> <p>Como usuario quiero poder acceder a un torneo concreto para consultar información más detallada sobre este.</p> <p style="text-align: center;">PRUEBAS ACEPTACIÓN</p> <ul style="list-style-type: none"> • Debe mostrarse toda la información de un torneo. • Si se hace click en el cuadro de partidos, calendario, venta de tickets, etc... se ha de mostrar más detalladamente. (opcional)
<p style="text-align: center;">RF11 - Ver Circuito</p> <p>Como usuario quiero poder acceder a un circuito concreto para consultar información más detallada sobre este.</p> <p style="text-align: center;">PRUEBAS ACEPTACIÓN</p> <ul style="list-style-type: none"> • Debe mostrarse toda la información de un circuito. • Si se hace click en un torneo, se debe redirigir a la vista del torneo elegido. (opcional) 	<p style="text-align: center;">RF12 - Ver Pareja</p> <p>Como usuario quiero poder acceder a una pareja concreta para consultar información más detallada sobre esta.</p> <p style="text-align: center;">PRUEBAS ACEPTACIÓN</p> <ul style="list-style-type: none"> • Debe mostrarse toda la información de una pareja • Si se hace click en un jugador, se debe redirigir a la vista del perfil del jugador elegido. (opcional)

RF13 - Ver Jugador

Como usuario
quiero poder acceder a un jugador concreto
para consultar información más detallada sobre este.

PRUEBAS ACEPTACIÓN

- Debe mostrarse toda la información de un jugador
- Debe poder accederse a los resultados de los últimos partidos/torneos. (opcional)

RF14 - CRUD Partido

Como administrador
quiero crear, leer, modificar y eliminar partidos
para mantener la lista de partidos actualizada.

PRUEBAS ACEPTACIÓN

- Se registra un partido nuevo, se pide un listado de partidos y aparece el partido nuevo.
- Se modifican los datos de un partido, se pide un listado de partidos y aparece el partido con los datos modificados.
- Se elimina un partido, se pide un listado de partidos y no aparece el partido.
- No se permiten registros de partidos nuevos ni modificaciones que dejen sin valor algún dato fundamental.

RF15 - CRUD Torneo

Como administrador
quiero crear, leer, modificar y eliminar torneos
para mantener la lista de torneos actualizada.

PRUEBAS ACEPTACIÓN

- Se registra un torneo nuevo, se pide un listado de torneos y aparece el torneo nuevo.
- Se modifican los datos de un torneo, se pide un listado de torneos y aparece el torneo con los datos modificados.
- Se elimina un torneo, se pide un listado de torneos y no aparece el torneo.
- No se permiten registros de torneos nuevos ni modificaciones que dejen sin valor algún dato fundamental.

RF16 - CRUD Circuito

Como administrador
quiero crear, leer, modificar y eliminar circuitos
para mantener la lista de circuitos actualizada.

PRUEBAS ACEPTACIÓN

- Se registra un circuito nuevo, se pide un listado de circuitos y aparecen los circuitos nuevos.
- Se modifican los datos de un circuito, se pide un listado de circuitos y aparece un circuito con los datos modificados.
- Se elimina un circuito, se pide un listado de circuitos y no aparece el circuito.
- No se permiten registros de circuitos nuevos ni modificaciones que dejen sin valor algún dato fundamental.

RF 17 - CRUD Pareja

Como administrador
quiero crear, leer, modificar y eliminar parejas
para mantener la lista de parejas actualizada.

PRUEBAS ACEPTACIÓN

- Se registra una pareja nueva, se pide un listado de parejas y aparece la pareja nueva.
- Se modifican los datos de una pareja, se pide un listado de parejas y aparece la pareja con los datos modificados.
- Se elimina una pareja, se pide un listado de parejas y no aparece la pareja.
- No se permiten registros de parejas nuevas ni modificaciones que dejen sin valor algún dato fundamental.

RF 18 - CRUD Jugador

Como administrador
quiero crear, leer, modificar y eliminar jugadores
para mantener la lista de jugadores actualizada.

PRUEBAS ACEPTACIÓN

- Se registra un jugador nuevo, se pide un listado de jugadores y aparece el jugador nuevo.
- Se modifican los datos de un jugador, se pide un listado de jugadores y aparece el jugador con los datos modificados.
- Se elimina un jugador, se pide un listado de jugadores y no aparece el jugador.
- No se permiten registros de jugadores nuevos ni modificaciones que dejen sin valor algún dato fundamental.

RF 19 - Enviar Incidencia

Como usuario
quiero poder contactar con el servicio de atención de la aplicación
para reportar cualquier fallo o incidencia.

PRUEBAS ACEPTACIÓN

- Deben aparecer una serie de campos que se han de rellenar.
- Si se rellenan de manera correcta, y se le da a enviar. Este mensaje debe llegar al destinatario en cuestión
- Si no están correctamente rellenos alguno de los campos, un error ha de saltar

RF 20 - Botón Partidos

Como usuario
quiero un medio en el menú de navegación
para acceder a la lista de partidos.

PRUEBAS ACEPTACIÓN

- Debe mostrarse un botón en el menú de navegación con un icono o texto identificador "Partidos".
- Si se hace click, redirige a la vista de partidos.

RF 21 - Botón Circuito

Como usuario
quiero un medio en el menú de navegación
para acceder a la lista de circuitos.

PRUEBAS ACEPTACIÓN

- Debe mostrarse un botón en el menú de navegación con un icono o el texto identificador “Circuitos”.
- Si se hace click, redirige a la vista de circuitos.

RF22 - Botón Torneo

Como usuario
quiero un medio en el menú de navegación
para acceder a la lista de torneos.

PRUEBAS ACEPTACIÓN

- Debe mostrarse un botón en el menú de navegación con un icono o el texto identificador “Torneos”.
- Si se hace click, redirige a la vista de torneos.

RF 23- Botón Parejas

Como usuario
quiero un medio en el menú de navegación
para acceder a la lista de parejas.

PRUEBAS ACEPTACIÓN

- Debe mostrarse un botón en el menú de navegación con un icono o el texto identificador “Parejas”.
- Si se hace click, redirige a la vista de parejas.

RF24 - Botón Jugadores

Como usuario
quiero un medio en el menú de navegación
para acceder a la lista de jugadores.

PRUEBAS ACEPTACIÓN

- Debe mostrarse un botón en el menú de navegación con un icono o el texto identificador “Jugadores”.
- Si se hace click, redirige a la vista de jugadores.

RF25 - Consultar Información sobre la aplicación

Como usuario
quiero datos de contacto, redes sociales, etc ...
para informarme sobre la aplicación.

SIN PRUEBAS

RF 26 - Botón Información

Como usuario
quiero un medio en el menú de navegación
para acceder a la información de la aplicación.

PRUEBAS ACEPTACIÓN

- Debe mostrarse un botón en el menú de navegación con un icono o texto identificador “Información”.
- Si se hace click, redirige a la vista de jugadores.

RF 27 - Botón Claro/Oscuro

Como usuario
quiero un medio en el menú de navegación
para cambiar el modo de visualización del entorno y facilitar la lectura.

PRUEBAS ACEPTACIÓN

- Debe mostrarse un botón en el menú de navegación con un icono
- Si se hace click, cambia de modo claro a oscuro y viceversa.

RF 28 - Consultar Noticias

Como usuario
quiero tener acceso a una lista
para ver todas las noticias disponibles.

PRUEBAS ACEPTACIÓN

- Deben mostrar una lista con las noticias más relevantes ordenadas cronológicamente.
- Si se hace click en una de las noticias, se debe redirigir a la vista de la noticia concreta.

RF 29- CRUD Noticias

Como administrador
quiero crear, leer, modificar y eliminar noticias
para mantener la lista actualizada.

PRUEBAS ACEPTACIÓN

- Se registra una noticia nueva, se pide un listado de noticias y aparece la noticia nueva.
- Se modifican los datos de una noticia , se pide un listado de noticias y aparece la noticia con los datos modificados.
- Se elimina una noticia, se pide un listado de noticias y no aparece la noticia.
- No se permiten registros de noticias nuevas ni modificaciones que dejen sin valor algún dato fundamental.

RF 30 - Ver Noticia

Como usuario
quiero poder acceder a una noticia concreta
para consultar información más detallada sobre esta.

PRUEBAS ACEPTACIÓN

- Debe mostrarse el contenido de la noticia. Puede incluir texto, elementos multimedia, hipervínculos...

RF 31 - Botón noticias

Como usuario
quiero poder acceder a una noticia concreta
para consultar información más detallada sobre esta.

PRUEBAS ACEPTACIÓN

- Si se hace click a una noticia concreta, se puede visualizar en detalle el contenido de la noticia.

RF 32 - Botón inicio

Como usuario
quiero un medio en el menú de navegación
para acceder a la página de inicio.

PRUEBAS ACEPTACIÓN

- Debe mostrarse un botón en el menú de navegación con el logo de la aplicación.
- Si se hace click, dirige a la vista principal.

RF 33 - Página inicio

Como usuario
quiero una vista principal de toda la app
para obtener un resumen general de todo el contenido.

PRUEBAS ACEPTACIÓN

- Cuando se accede a la web se debe ver una página de inicio en la que aparecen la barra de navegación y un feed con la última noticia . Al final de la página se pueden acceder a otras noticias recomendadas.

RF 34 - Página inicio admin

Como administrador
quiero una vista principal de
para acceder a la página de inicio.

PRUEBAS ACEPTACIÓN

RF35 - CRUD instituciones

Como administrador
quiero crear, leer, modificar y eliminar instituciones.
para mantener la lista de instituciones actualizada.

PRUEBAS ACEPTACIÓN

- Se registra una institución nueva, se pide un listado de instituciones y aparecen las instituciones nuevas.
- Se modifican los datos de una institución, se pide un listado de instituciones y aparece la institución con los datos modificados.
- Se elimina una institución, se pide un listado de instituciones y no aparece el circuito.
- No se permiten registros de circuitos nuevos ni modificaciones que dejen sin valor algún dato fundamental.



Requisitos Obligatorios



Requisitos Opcionales

5.2 - Requisitos No Funcionales

RNF 1 - Responsive Design

Como usuario,
quiero que el diseño de la aplicación se adapte a mi dispositivo
para tener mayor comodidad durante su uso.

Sin pruebas

RNF 2 - Diseño Minimalista

Como usuario
quiero que la información tenga una estructura simple y de fácil visualización
para tener una mejor experiencia.

Sin pruebas

RNF 3 - Plantillas de carga

Como usuario,
quiero ver componentes de espera cuando sea necesario,
para saber que la aplicación está procesando.

PRUEBAS ACEPTACIÓN

- Si se hace click en un proceso que requiera información proporcionada por el backend deberán renderizarse componentes que den feedback de que el proceso se está llevando a cabo

RNF4 - Multiplataforma

Como administrador,
quiero que la aplicación esté disponible en formato web como en formato app (distintos sistemas operativos,
para que la aplicación sea accesible a más gente.

Sin pruebas

RNF5 - Modo claro	RNF6 - Modo oscuro
Como usuario quiero que la aplicación tenga disponible un modo de visualización claro para una mejor visualización según el entorno	Como usuario quiero que la aplicación tenga disponible un modo de visualización oscuro para una mejor visualización según el entorno
PRUEBAS ACEPTACIÓN <ul style="list-style-type: none">• A la luz del día, el usuario puede visualizar los contenidos de una manera legible y con una gama de colores no dañina a la vista.	PRUEBAS ACEPTACIÓN <ul style="list-style-type: none">• En la oscuridad, el usuario puede visualizar los contenidos de una manera legible y con una gama de colores no dañina a la vista.



Requisitos Obligatorios



Requisitos Opcionales

Observar diagrama de requisitos en [ANEXO I](#).

6 - Casos de Uso

Ver [ANEXO II](#)

CU1- Iniciar Sesión

Contexto de uso

Cuando un administrador lo necesite puede autenticarse para diferenciarse de un usuario común.

Precondiciones

- Tener una cuenta de administrador en el sistema.
- Conocer los credenciales

Garantías de éxito

Accedes a la interfaz propia del administrador

Escenario principal

1. El administrador accede a la sub-ruta correspondiente del Log In.
2. Introduce las credenciales.
3. El sistema muestra la interfaz de administrador de la página de inicio.

Escenarios alternativos

- El administrador introduce erróneamente las credenciales y el sistema muestra un mensaje de error para que el usuario vuelva a intentarlo.

CU2- Crear recurso

Contexto de uso

Cuando un administrador lo necesite puede crear un recurso

Precondiciones

- Haber iniciado sesión como administrador.
- Tener las características necesarias para la creación de un objeto

Garantías de éxito

Creas el recurso en la base de datos

Escenario principal

1. El administrador pulsa la opción para crear un nuevo recurso.
2. El sistema muestra el formulario para crear un nuevo recurso.
3. El administrador rellena la información para crear el recurso.
4. El administrador pulsa la opción de crear el recurso.
5. El recurso se crea en el sistema y se muestra un mensaje de éxito al administrador.

Escenarios alternativos

- El recurso no se ha creado correctamente y se muestra un mensaje al administrador.

CU3- Consultar recurso

Contexto de uso

Cuando un administrador lo necesite puede leer la información guardada en la base de datos de forma simplificada y esquemática.

Precondiciones

- Haber iniciado sesión en el sistema como administrador.

Garantías de éxito

Se muestra la información requerida en pantalla.

Escenario principal

1. El administrador pulsa la opción de consultar un recurso.
2. El administrador selecciona la información a la que desea acceder.
3. El sistema muestra la información requerida.

Escenarios alternativos

- El recurso no se puede leer y se muestra un mensaje de error al administrador.

CU4- Editar recurso

Contexto de uso

Cuando un administrador lo necesite puede editar cualquier recurso: Jugador, Partido, Competición, Noticia, etc.

Precondiciones

- Haber iniciado sesión como administrador en el sistema.
- Que exista el recurso que se quiere editar.

Garantías de éxito

El cambio se muestra correctamente tras haberlo actualizado a su nuevo valor.

Escenario principal

1. El administrador accede a la página de administrador.
2. El administrador selecciona el recurso a editar y se muestra la interfaz correspondiente.
3. Edita dicho recurso y confirma los cambios.
4. El sistema muestra los cambios realizados y devuelve al administrador a la página de inicio.

Escenarios alternativos

1. El administrador no confirma los cambios antes de salir del editor.
2. Algunos de los campos editados son incorrectos o incompatibles y el sistema muestra un error al administrador.

CU5- Eliminar recurso

Contexto de uso

- Cuando un administrador lo necesite puede eliminar cualquier recurso: Jugador, Partido, Competición, Noticia, etc.

Precondiciones

- Haber iniciado sesión en el sistema.
- Conocer los credenciales

Garantías de éxito

Una vez realizado, el recurso ya no aparece.

Escenario principal

1. El administrador accede a la página de administrador.
2. El administrador selecciona el recurso a eliminar y entra en el menú correspondiente.
3. Elimina dicho recurso y confirma los cambios.
4. El sistema muestra la interfaz de administrador de la página de inicio.

Escenarios alternativos

1. El administrador no confirma los cambios antes de salir del editor.

CU6- Botón claro/oscuro

Contexto de uso

Cuando un usuario lo necesite puede cambiar el color de la aplicación.

Precondiciones

- Tener acceso al menú de navegación de la aplicación.

Garantías de éxito

Cambias el entorno de visualización de la aplicación.

Escenario principal

1. El usuario pulsa el botón.
2. El entorno de visualización de la aplicación cambia de tono claro a oscuro o viceversa.

CU7- Buscador

Contexto de uso

Un usuario puede filtrar contenidos por texto y acceder a ellos

Precondiciones

- Tener acceso al buscador
- Tener las características necesarias para la creación de un objeto

Garantías de éxito

El usuario ve la información deseada y accede a ella

Escenario principal

1. El usuario rellena el campo con la información que desea buscar.
2. El servidor devuelve la información filtrada según la búsqueda del usuario.
3. Se muestra la información.

Escenarios alternativos

1. La búsqueda no tiene resultados

CU8- Enviar incidencia

Contexto de uso

Un usuario puede encontrar algún error en la aplicación y puede reportarlo.

Precondiciones

- Ver qué recurso o dato ha fallado o es erróneo.

Garantías de éxito

El usuario ve un pop-up diciendo que la solicitud de arreglo se ha enviado de manera correcta.

Escenario principal

1. El administrador observa y recopila las características del error para la creación de la incidencia.
2. Invoca la función de Enviar Incidencia.
3. La incidencia ha sido enviada.

Escenarios alternativos

1. El usuario o administrador introduce las características de una incidencia que es correcta, es decir, que no causa ningún error.
2. El sistema le envía un correo al usuario comentándole que no se ha producido ninguna modificación.

CU9- Plantilla de carga

Contexto de uso

Cuando el usuario ha realizado alguna petición y el servidor tarda cierto tiempo en responder.

Precondiciones

- Hacer una petición a alguna funcionalidad.

Garantías de éxito

La plantilla realiza durante un periodo de tiempo (suele ser corto no más de 5 segundos) una animación que indica al usuario que debe esperar.

Escenario principal

1. El usuario quiere consultar cierto recurso.
2. Invoca cualquier función que le haga llegar a ese recurso.
3. Mientras que por detrás se trabaja para acceder a ese recurso, al usuario se le ofrece una animación que le hace ver que está cargando el contenido.
4. Termina de cargar, la plantilla desaparece y puede seguir usando la aplicación.

Escenarios alternativos

1. La plantilla de carga no desaparece debido a algún problema de la petición o de la base de datos.
2. Después de cierto “time-out” la aplicación le notifica que no ha sido posible encontrar cierto recurso.

CU10- Botón de Inicio

Contexto de uso

Cuando el usuario hace click en el botón de inicio accede a la página de inicio.

Precondiciones

- Existencia del botón de inicio y de la página de inicio.

Garantías de éxito

La vista del usuario se redirige a la vista principal

Escenario principal


1. El usuario quiere regresar al inicio.
2. Click en el botón de inicio.
3. Su vista es redirigida a la vista principal.

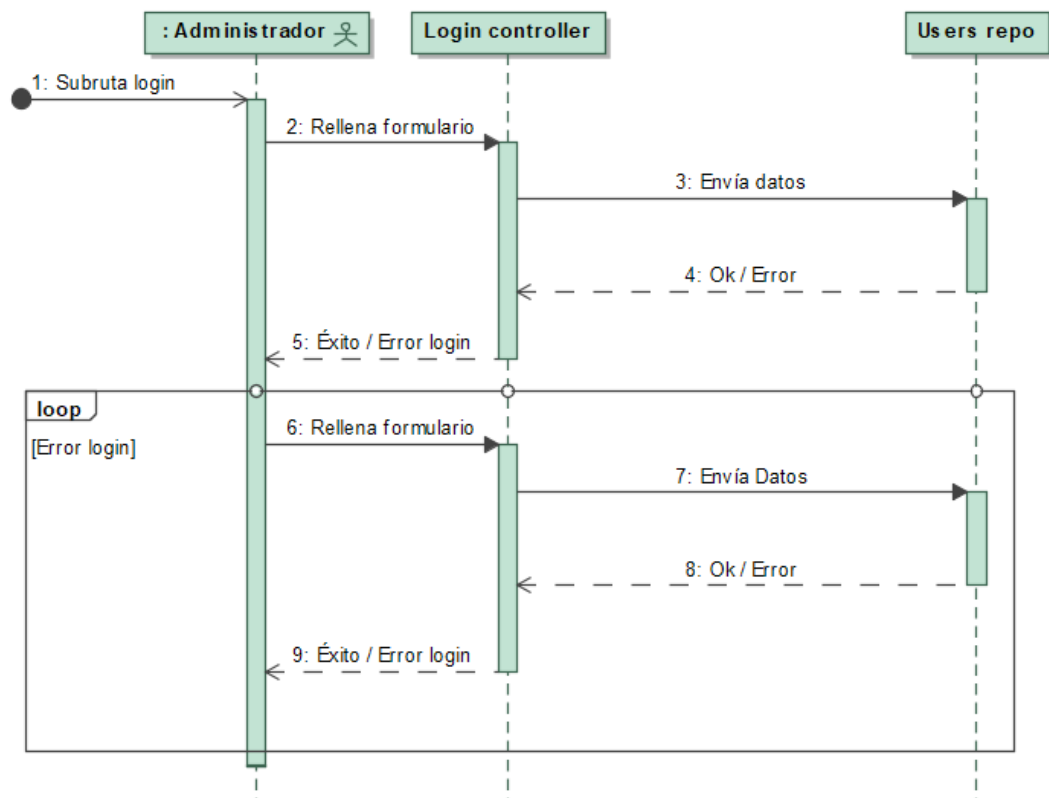
7 - Modelo de Dominio

Ver [ANEXO III](#)

8 - Diagramas de Secuencia

CU1- Iniciar Sesión

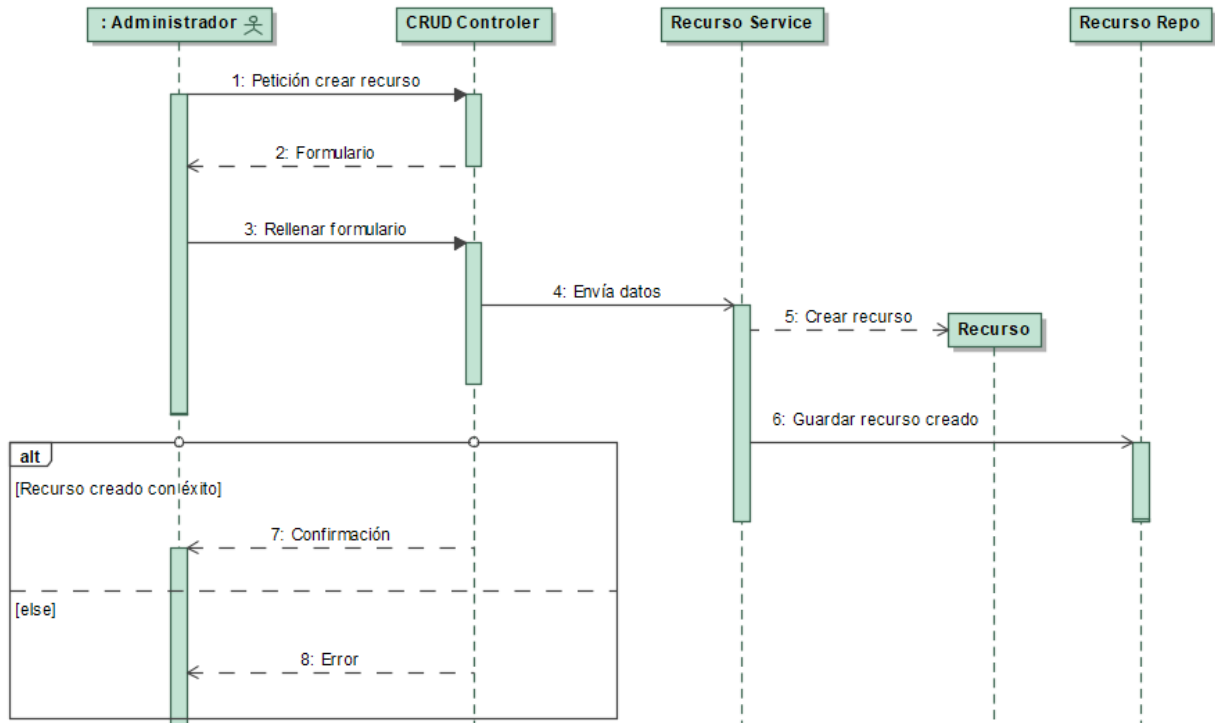
interaction CU1 - Iniciar Sesión [ CU1 - Iniciar Sesión]



El administrador accede a la sub-ruta para iniciar sesión, rellena el formulario que se le muestra y el controlador envía los datos que se comparan con los registrados en la base de datos y responde iniciando sesión o mostrando error. En caso de error se vuelve a mostrar el formulario hasta que la respuesta del servidor sea afirmativa.

CU2- Crear Recurso

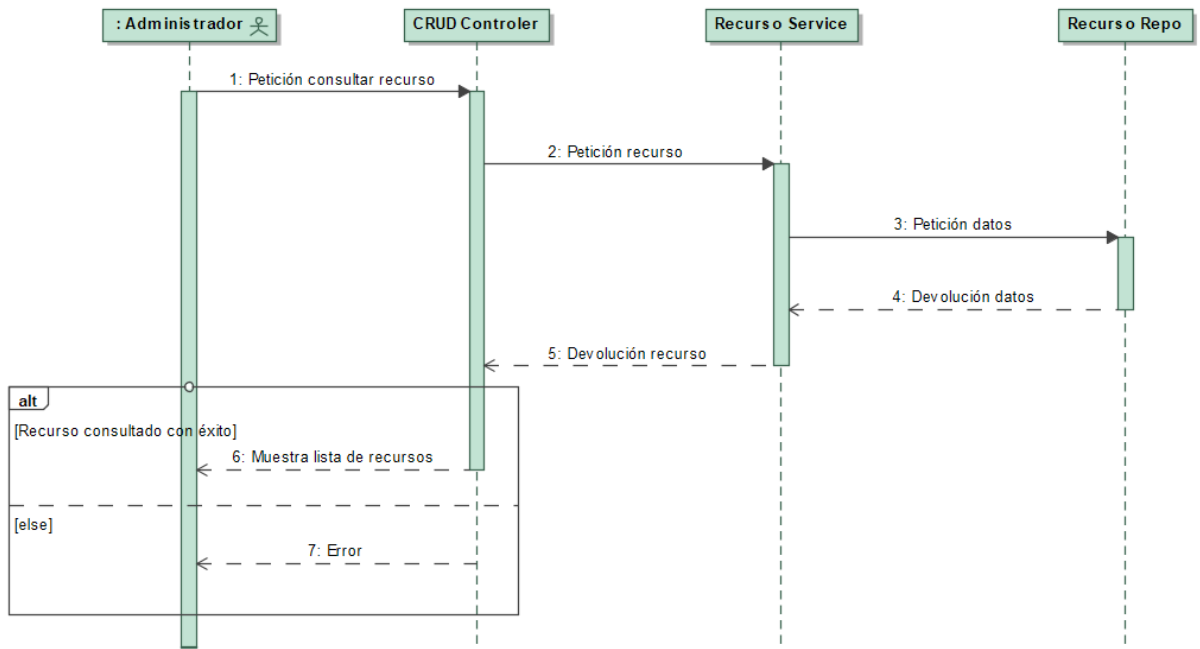
interaction CU2 - Crear Recurso [CU2 - Crear Recurso]



El administrador pide al CRUD controller que quiere crear un nuevo recurso. Este le devuelve un formulario a rellenar, y cuando tiene los datos, envía los datos al Recurso Service y este crea el nuevo recurso y lo guarda en el repositorio. El CRUD Controller no espera confirmación y nada más enviar los datos al Recurso Service, manda confirmación / error al administrador en función de si ha enviado los datos con éxito o no.

CU3- Consultar Recurso

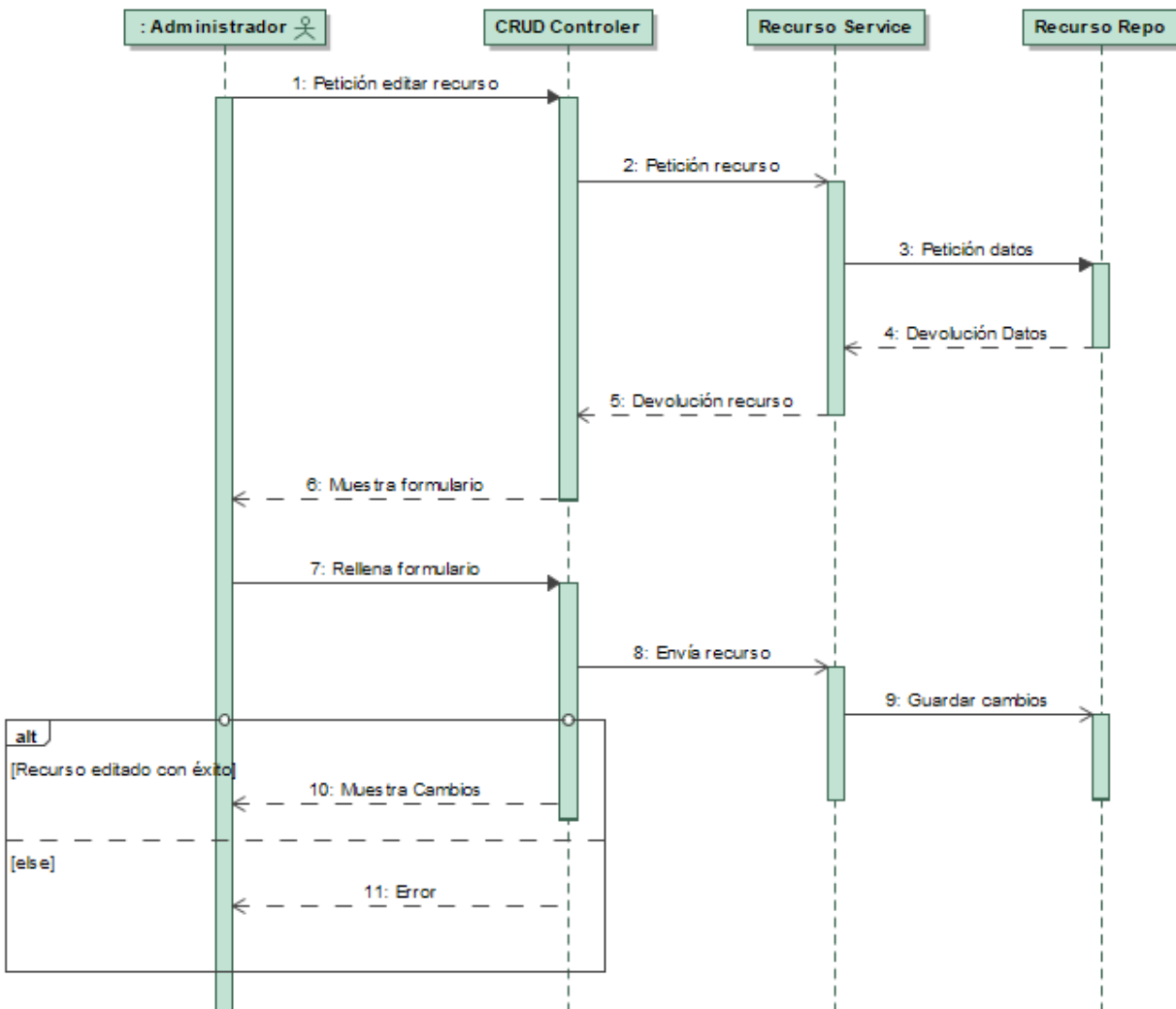
interaction CU3 - Consultar Recurso [CU3 - Consultar Recurso]



El administrador pide al CRUD controller que quiere consultar la lista de recursos. Este le traslada la petición al Recurso Service y este a su vez, al repositorio, que devuelve los datos de vuelta hasta el administrador, o un error en caso de no haber sido posible la consulta.

CU4- Editar Recurso

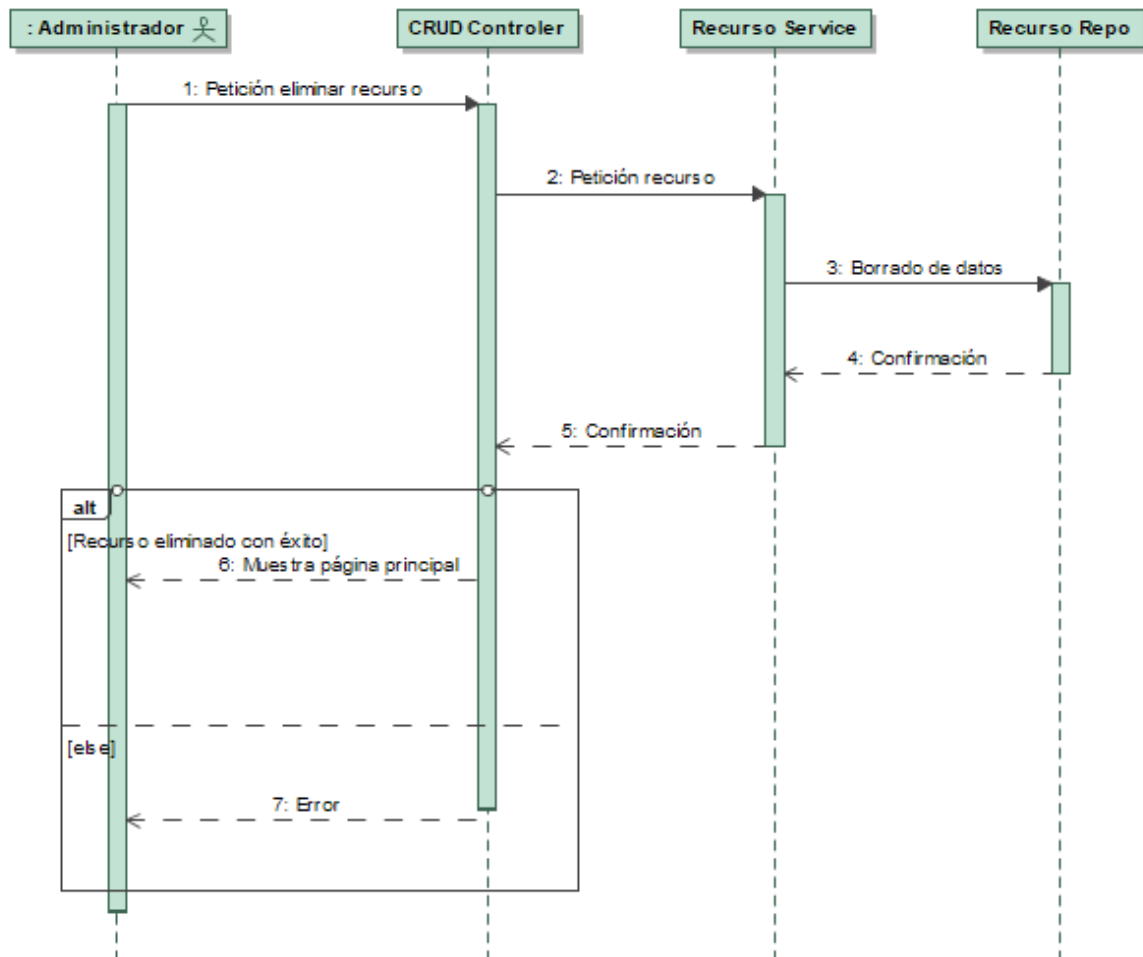
interaction CU4 - Editar Recurso [CU4 - Editar Recurso]



El administrador pide al CRUD Controller que quiere editar un recurso. Siguiendo el diagrama de la consulta de recursos, se le muestra al administrador una lista de recursos. Este puede seleccionar y modificar los recursos que desee. Tras ello, enviará los cambios al CRUD Controller, que los enviará al Recurso Service y a su vez este guardará los cambios en el repositorio. El CRUD Controller mostrará un mensaje de confirmación o de error en función de si la modificación se ha realizado con éxito.

CU5- Eliminar Recurso

interaction CU5 - Eliminar Recurso [[CU5 - Eliminar Recurso]]

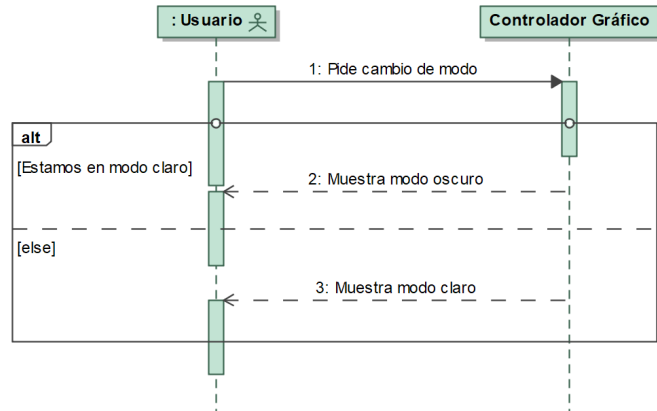


El administrador pide al CRUD Controller que quiere eliminar un recurso. Siguiendo el mismo proceso de la edición de datos, se mostrará al administrador una lista de recursos y este podrá seleccionar el recurso que desee eliminar. Una vez seleccionado, el CRUD Controller trasladará la petición al Recurso Service y este llevará a cabo el borrado de datos del repositorio. Que devolverá un mensaje de confirmación o de error al administrador en función de si el borrado se ha realizado con éxito.

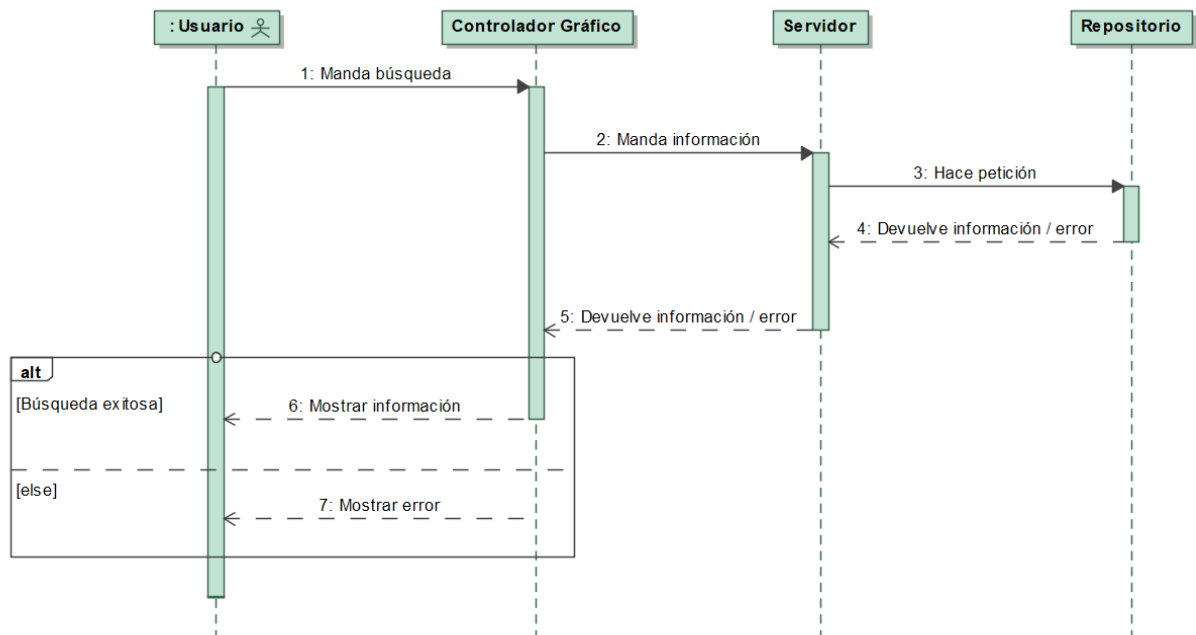
CU6- Botón Claro / Oscuro

interaction CU6 - Botón Claro/Oscuro [CU6 - Botón Claro/Oscuro]

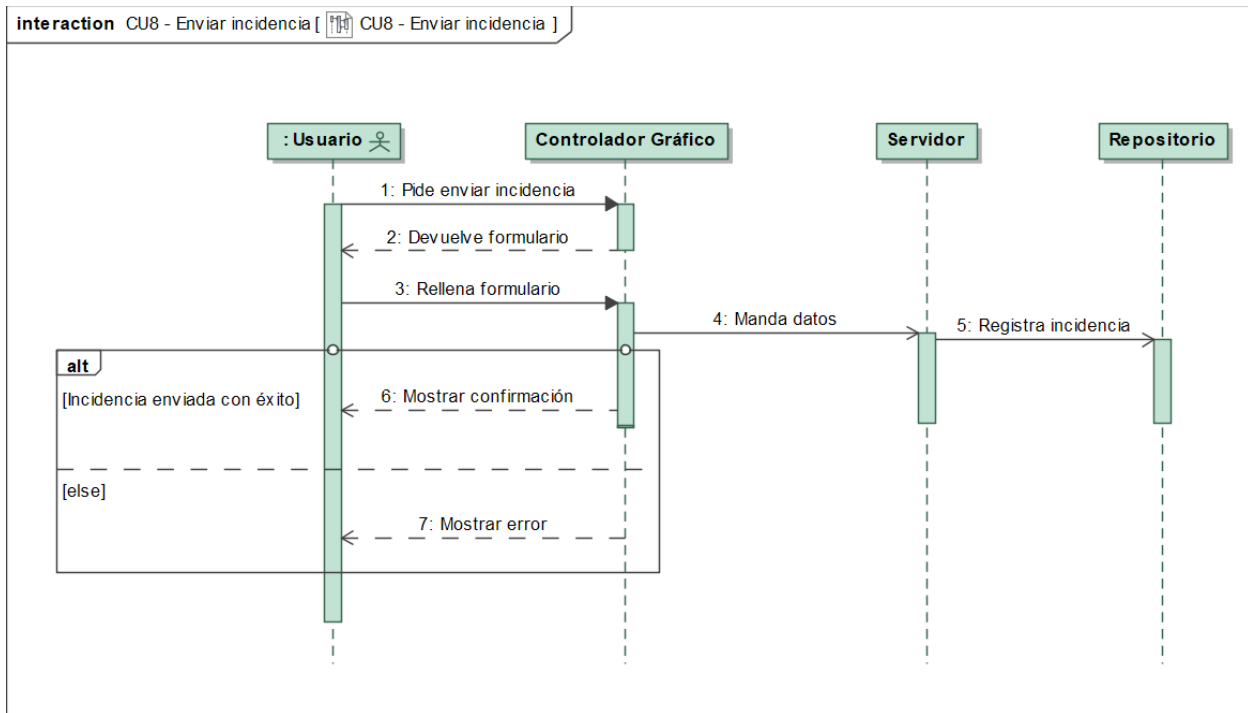
El usuario pide un cambio de modo de color, si estamos en modo claro el controlador gráfico cambiará a modo oscuro y viceversa.

**CU7- Buscador**

interaction CU7 - Buscador [CU7 - Buscador]




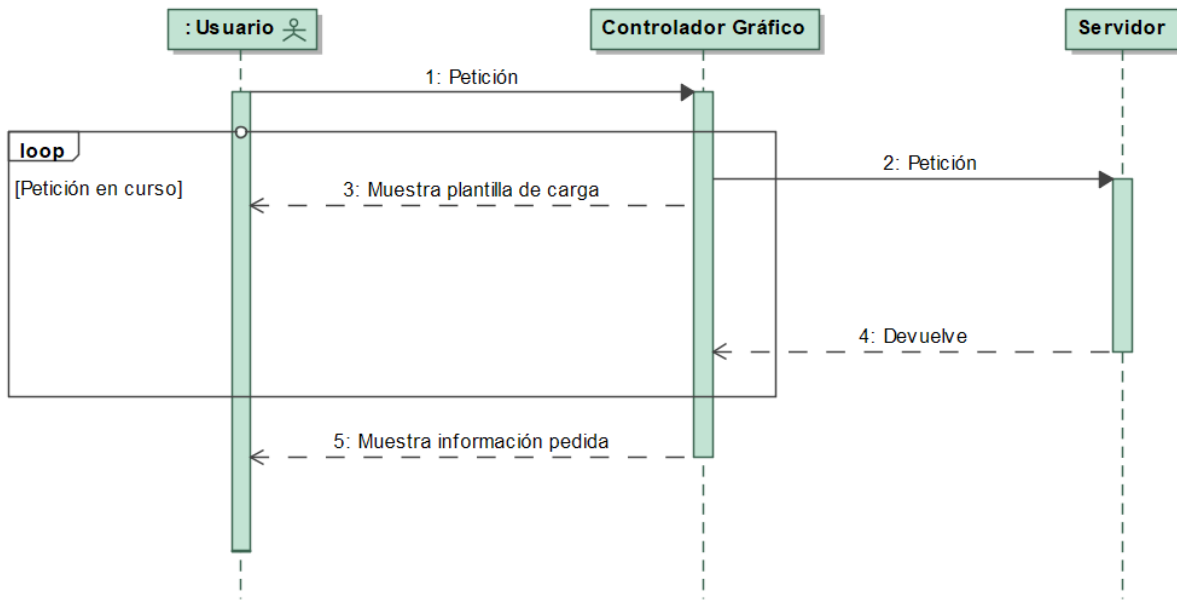
El usuario rellena el formulario del buscador y el controlador gráfico manda los datos al servidor, y este hace una petición al repositorio. El repositorio devuelve la información requerida o un error, el servidor y el controlador transmiten de vuelta la información o el error hasta el usuario.

CU8- Enviar Incidencia

El usuario indica al controlador que quiere enviar una incidencia, este le devuelve un formulario y cuando el usuario lo rellena, el controlador manda los datos al servidor y este al repositorio. Si la incidencia se envió con éxito, el controlador muestra un mensaje de confirmación, si no, mostrará uno de error.


CU9- Plantilla de Carga

interaction CU9 - Plantilla de carga [ CU9 - Plantilla de carga]

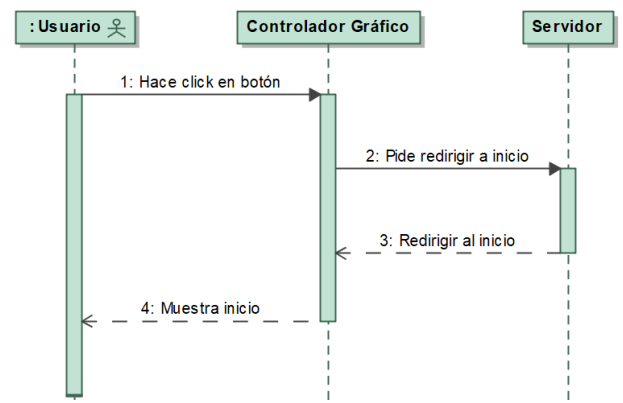


El usuario hace una petición cualquiera al controlador gráfico, y mientras la petición esté en curso, el controlador mostrará una plantilla de carga al usuario hasta que la petición se haya completado. Entonces el controlador mostrará la información requerida.

CU10- Botón de Inicio

interaction CU10 - Botón de inicio [ CU10 - Botón de inicio]

El usuario hace click en el botón de inicio, el controlador gráfico pregunta al servidor por la página de inicio, este redirecciona la aplicación y el controlador muestra por pantalla al usuario la página de inicio.



9 - Pruebas JUnit

Adjuntamos capturas de pantalla de las pruebas JUnit de los test de los casos de uso de jugadores, las clases se llaman *JugadorCreatorTest*, *JugadorFinderTest* y *JugadorRemoverTest*.

En todas ellas se utilizan correctamente los mocks y se tienen en cuenta todos los tests posibles.

Clase *JugadorCreator*

En la clase *JugadorCreator* implementamos las funciones necesarias para crear un jugador y guardarlo en la base de datos. En la siguiente clase de testeo comprobamos que el jugador se crea y guarda correctamente en la base de datos.

```
public class JugadorCreatorTest {
    JugadorRepository repository = mock(JugadorRepository.class);
    JugadorCreator creator = new JugadorCreator(repository);

    @Test
    @DisplayName("El test debería invocar de forma correcta el metodo save del repositorio")
    void guardaUnJugadorValido() {
        CreateJugadorRequest request = CreateJugadorRequestMother.random();
        Jugador jugador = JugadorMother.createFromRequest(request);
        creator.create(request);
        verify(repository, atLeastOnce()).save(jugador);
    }
}
```

Clase *JugadorFinder*

En la clase *JugadorFinder* implementamos las funciones necesarias para encontrar un jugador a partir de su id en la base de datos o todos los jugadores en la misma. En la siguiente clase de testeo comprobamos que efectivamente la función nos devuelve un jugador a partir de su id, si este existe, null en caso de que no exista, o todos los jugadores anteriormente guardados en la base de datos.

```
class JugadorFinderTest {

    JugadorRepository repository = mock(JugadorRepository.class);
    JugadorFinder finder = new JugadorFinder(repository);

    @Test
    void shouldReturnAValidJugadorIfExists() {
        Long id = 99L;
        Jugador jugador = JugadorMother.random();

        when(repository.findById(id)).thenReturn(Optional.of(jugador));

        Jugador findJugador = finder.find(id);

        assertNotNull(findJugador);
    }

    @Test
    void shouldReturnNullIfJugadorDoesntExist() {
        Long id = 99L;
        when(repository.findById(id)).thenReturn(Optional.empty());
        Jugador findJugador = finder.find(id);
        assertNull(findJugador);
    }

    @Test
    void shouldReturnAllJugadores() {
        List<Jugador> jugadores = Arrays.asList(
            JugadorMother.create(nombre: "Arturo", apellidos: "Aguilera", Sexo.MASCULINO, Pais.ES),
            JugadorMother.create(nombre: "Maria Jose", apellidos: "Garcia Tobaruela", Sexo.FEMENINO, Pais.ES),
            JugadorMother.random()
        );

        when(repository.findAll()).thenReturn(jugadores);

        List<Jugador> listaJugadores = finder.findAll();

        assertEquals(listaJugadores.size(), jugadores.size());
        assertNotNull(listaJugadores);
    }
}
```

Clase *JugadorRemover*

En la clase *JugadorRemover* implementamos las funciones necesarias para que, dada la id de un jugador, podamos eliminarlo de la base de datos. En caso de que no exista ningún jugador con esa id, debe elevar una excepción de tipo *IllegalStateException*. En esta clase de Test, comprobamos que efectivamente, un jugador existente es eliminado correctamente si le damos una id válida, y recibe la excepción en caso contrario.

```
class JugadorRemoverTest {
    private final JugadorRepository jugadorRepository = mock(JugadorRepository.class);
    private final JugadorRemover jugadorRemover = new JugadorRemover(jugadorRepository);
    Long id = 99L;

    @Test
    @DisplayName("Debe eliminar un jugador si este existe")
    void shouldDeleteJugadorById() {
        when(jugadorRepository.existsById(id)).thenReturn(true);
        jugadorRemover.remove(id);
        verify(jugadorRepository, atLeastOnce()).deleteById(id);
    }

    @Test
    @DisplayName("Debe elevar un excepcion si el jugador no existe")
    void shouldThrowAnExceptionIfJugadorDoesNotExists() {
        when(jugadorRepository.existsById(id)).thenReturn(false);
        assertThrows(IllegalStateException.class, () -> {
            jugadorRemover.remove(id);
        });
    }
}
```

10 - Herramientas de software usadas en el proyecto

En cuanto a comunicación y coordinación, usamos WhatsApp y Discord, y para la planificación de proyecto, Trello.

Para la creación de documentos empleamos Google Docs y para el trabajo colaborativo GitHub.

Para la creación del logo de nuestra aplicación software usamos Adobe Illustrator 2018 y Adobe Photoshop 2020.

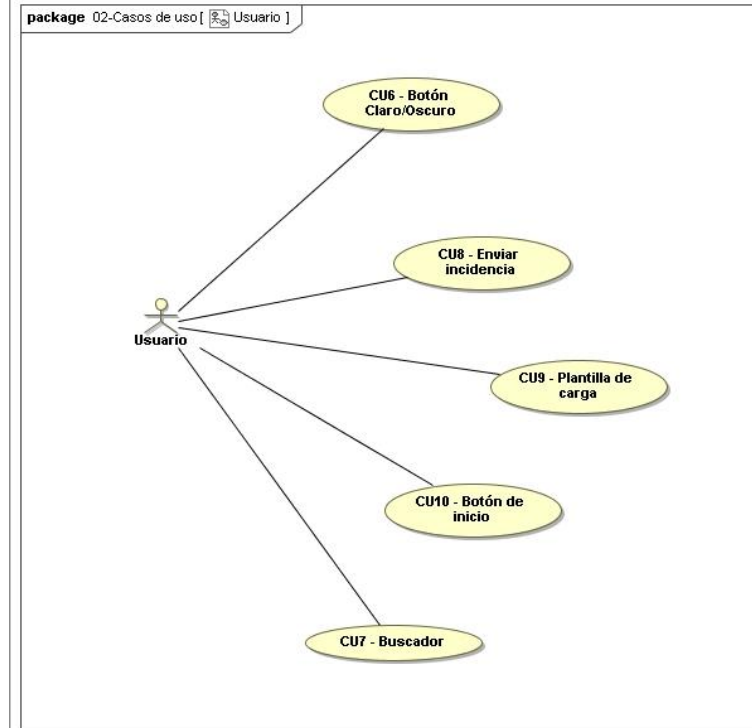
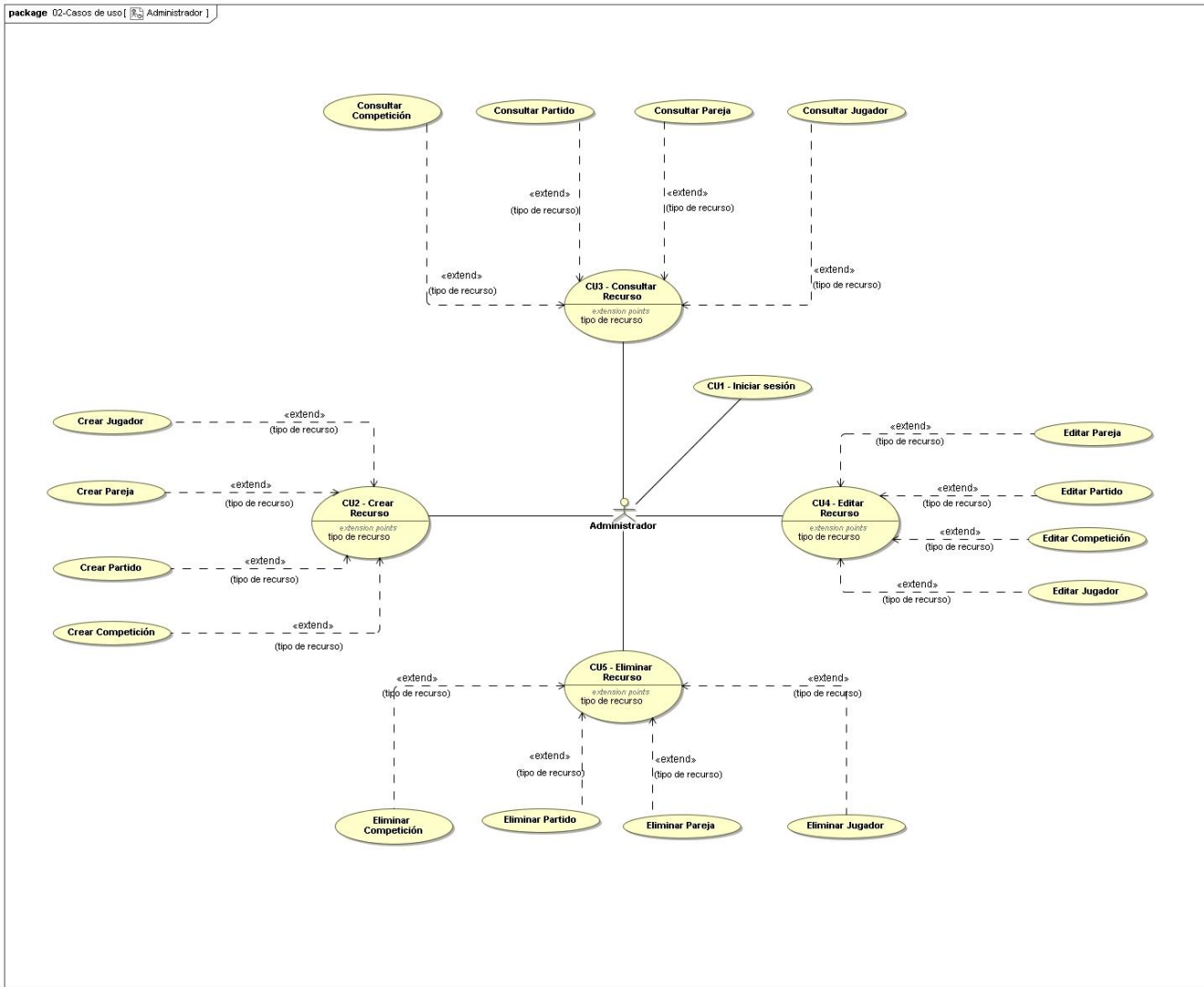
En lo relacionado a diagramas de requisitos, diagramas de casos de uso, etc... Usamos MagicDraw.

Para realizar el diseño de la interfaz gráfica estamos trabajando en Figma, y para la programación empleamos Visual Studio Code, tanto en backend como frontend.





ANEXO II - Diagramas de casos de uso - Magic Draw



ANEXO III - Diagramas de clases - Magic Draw

