

Proyecto Realizado por Cristian Ortega, Carlos
Cortés, Víctor Marín y Samuel Gómez

GESTIÓN DE EQUIPO

Iles Luis Vives



ÍNDICE

1.- Fusión de proyectos, justificaciones de diseño

2.- Requisitos

2.1.- Requisitos funcionales

2.2.- Requisitos no funcionales

2.3.- Requisitos de información

3.- Diagrama de casos de uso

4.- Casos de uso

4.1.- Alta de miembro

4.2.- Modificación de miembro

4.3.- Baja de miembro

4.4.- Realizar convocatoria

5.- Diagrama de clases del modelo de negocio

6.- Modelo entidad-relación y tablas resultantes

7.- Diagramas de secuencia

7.1.- Buscar miembro

7.2.- Alta de miembro

7.3.- Modificación de miembro

7.4.- Baja de miembro

7.5.- Realizar convocatoria

8.- Grafo de navegación y diseño de vistas

9.- Informes de cobertura y tests

10.- Estimación de costes

11.- Planificación y ejecución de tareas (Trello)

12.- Conclusiones



1.- Fusión de proyectos, justificaciones de diseño

A continuación se detalla qué partes se han seleccionado de cada proyecto de los dos que se han fusionado.

Proyecto 1

Almacenamiento: Consideramos que está más optimizado. Además, sigue el patrón fachada: un storage central que deriva la importación o exportación de datos a uno u otro en función de la extensión del archivo que se le pase por parámetros.

Modelos, Entidades, DTOs, State, User, Configuration Properties: todos los modelos han sido seleccionados en bloque para guardar la coherencia con el storage.

Mapeador: Dado que actúa como un traductor entre los modelos.

Vistas, Controladores y ViewModels: Las funcionalidades requeridas estaban más avanzadas.

Proyecto 2

Servicio: Los algoritmos del servicio (incluyendo llamadas a la caché, al repositorio, a los storages y al validador) eran más eficientes, sólidos y seguros. Otro punto a valorar es que tenía ROP correctamente implementada en todas las funciones.

Repositorio: Por coherencia con la elección del servicio y motivos similares, se eligió también el repositorio que llevaba inyectado.

DAO: Para maximizar la compatibilidad y la integración con el repositorio y el servicio seleccionados.

Validador: Las comprobaciones de los datos realizadas nos parecían más acertadas.



2.- Requisitos

2.1.- Requisitos funcionales

RF1: Realizar operaciones CRUD (Create, Read, Update y Delete) para gestionar los miembros del equipo.

RF2: Asignar un id único a cada miembro del equipo guardado.

RF3: Representar a los miembros del equipo como Jugadores o Entrenadores.

RF4: De todos los miembros, almacenar nombre, apellidos, fecha de nacimiento, fecha de incorporación, salario, país de origen y una imagen de perfil.

RF5: De los Jugadores, además, almacenar posición en el campo (portero, defensa, centrocampista o delantero), número de camiseta, altura, peso, número de goles anotados, partidos jugados Y minutos jugados.

RF6: De los Entrenadores, además, almacenar su área de especialización (entrenador de porteros, entrenador principal y entrenador asistente).

RF7: Validar los datos de los miembros antes de guardarlos para evitar errores o inconsistencias.

RF8: Permitir la importación y exportación de miembros desde y a ficheros CSV, JSON, XML, BIN y ZIP a través de la interfaz de usuario siempre y cuando se esté logueado con una cuenta con privilegios de administrador. Además, se deben poder exportar las alineaciones a HTML y PDF.

RF9: Implementar una caché con filosofía LRU con un tamaño y caducidad de datos parametrizables a través de un fichero de configuración.

RF10: Implementar una interfaz de usuario con iconos y un logotipo personalizados y únicos para la aplicación, asegurando su calidad y distinción del resto del mercado.

RF11: Implementar una splash screen para mostrar mientras se abre la aplicación.

RF12: Implementar un inicio de sesión con una base de datos de usuarios en la que se guarda el usuario y su contraseña usando la función de hash criptográfica de BCrypt.

RF13: Crear dos vistas de la aplicación: usuario (sólo puede hacer visualizaciones de datos, filtrados y búsquedas) y administrador (tiene permisos para crear, editar o eliminar integrantes del equipo).

RF14: Conjunto de estadísticas en el footer de la aplicación que recogen datos como la media de goles, minutos jugados, y coste total de toda la plantilla (escogiendo los datos de entidades verdaderamente relevantes para no falsear la estadística final).

RF15: Los usuarios deben poder registrarse.



RF16: Debe existir un usuario administrador creado directamente en la base de datos mediante un script de inicio.

RF17: Realizar operaciones CRUD (Create, Read, Update y Delete) para gestionar las convocatorias.

RF18: Se deben poder realizar copias de seguridad completas del sistema.

2.2.- Requisitos no funcionales

RNF 1: Los datos de los miembros del equipo deben validarse para evitar errores e inconsistencias.

RNF 2: La caché debe mejorar la velocidad de acceso a los datos.

RNF 3: El menú debe ser intuitivo y facilitar el manejo de la aplicación a los usuarios.

RNF 4: La aplicación debe implementar un sistema de logs para facilitar el seguimiento de la traza del programa y la corrección de errores.

RNF 5: Todo el código debe estar documentado y el proyecto debe incluir la documentación generada por dokka.

RNF 6: Todo el código (excepto los controladores) debe ser testeado.

RNF 7: La aplicación debe poder ejecutarse desde la terminal gracias al archivo .jar

RNF 8: Durante el desarrollo del proyecto, el equipo debe trabajar usando GitFlow/Git/GitHub.

RNF 9: Se debe proporcionar un instalador de la aplicación.

2.3 Requisitos de información

RI1: Todos los **Integrantes** del equipo tendrán los siguientes campos comunes.

Id: De tipo Long para posibilitar una mayor escalabilidad del proyecto, al permitir un rango más amplio de valores que el tipo Int.

Nombre: De tipo String. El nombre no puede estar vacío ni tener una extensión menor o igual a 2 caracteres.

Apellidos: De tipo String. Los apellidos no pueden estar vacíos ni tener una extensión menor o igual a 2 caracteres.



Fecha de nacimiento: de tipo LocalDate para que las fechas respeten el estándar ISO-8601. La fecha de nacimiento no puede ser posterior ni a la fecha actual ni a la fecha de incorporación.

Fecha de incorporación: de tipo LocalDate para que las fechas respeten el estándar ISO-8601. La fecha de incorporación no puede ser posterior a la fecha actual ni anterior a la fecha de nacimiento.

Salario: de tipo Double para poder representar los céntimos como cifras decimales. El salario no puede ser menor o igual a 0.0..

País de origen: de tipo String. El país de origen no puede tener una extensión menor o igual a 2 caracteres.

Imagen: de tipo String. No puede estar vacía. Si no la hay, se selecciona una por defecto incluida en resources.

RI2: Los miembros del equipo que sean **Jugadores** tendrán los siguientes campos específicos.

Posición: de tipo Posición (una enum class que contiene los valores portero, defensa, centrocampista y delantero).

Dorsal: De tipo Int, ya que solo se admitirán dorsales hasta el 99.

Altura: De tipo Double, para poder representar los cms como cifras decimales. La altura no puede ser menor o igual a 1 m ni mayor que 3 m.

Peso: De tipo Double, para poder representar los g como cifras decimales. El peso no puede ser menor o igual a 45 kg ni mayor a 150kg.

Goles: De tipo Int. Los goles no pueden ser negativos.

Partidos jugados: De tipo Int. Los partidos jugados no pueden ser negativos.

Minutos jugados: De tipo Int. Los minutos jugados no pueden ser negativos

RI3: Los miembros del equipo que sean **Entrenadores** tendrán el siguiente campo específico.

Especialidad: de tipo Especialidad (una enum class que contiene los valores entrenador de porteros, entrenador principal o entrenador asistente).

RI4: Respecto a las **convocatorias** tendrán los siguientes campos y condiciones:

Id: de tipo Long para posibilitar una mayor escalabilidad del proyecto, al permitir un rango más amplio de valores que el tipo Int.

Fecha: de tipo LocalDate para que las fechas respeten el estándar ISO-8601.

Descripción: de tipo String.



Pueden tener un máximo de 18 jugadores.

No puede haber más de dos porteros en una convocatoria.

No se pueden incluir miembros del cuerpo técnico como jugadores en una convocatoria.

Se debe incluir al entrenador principal en la convocatoria.

Se debe definir el 11 titular dentro de cada convocatoria.

RI5: Respecto al **equipo**, debe definirse su:

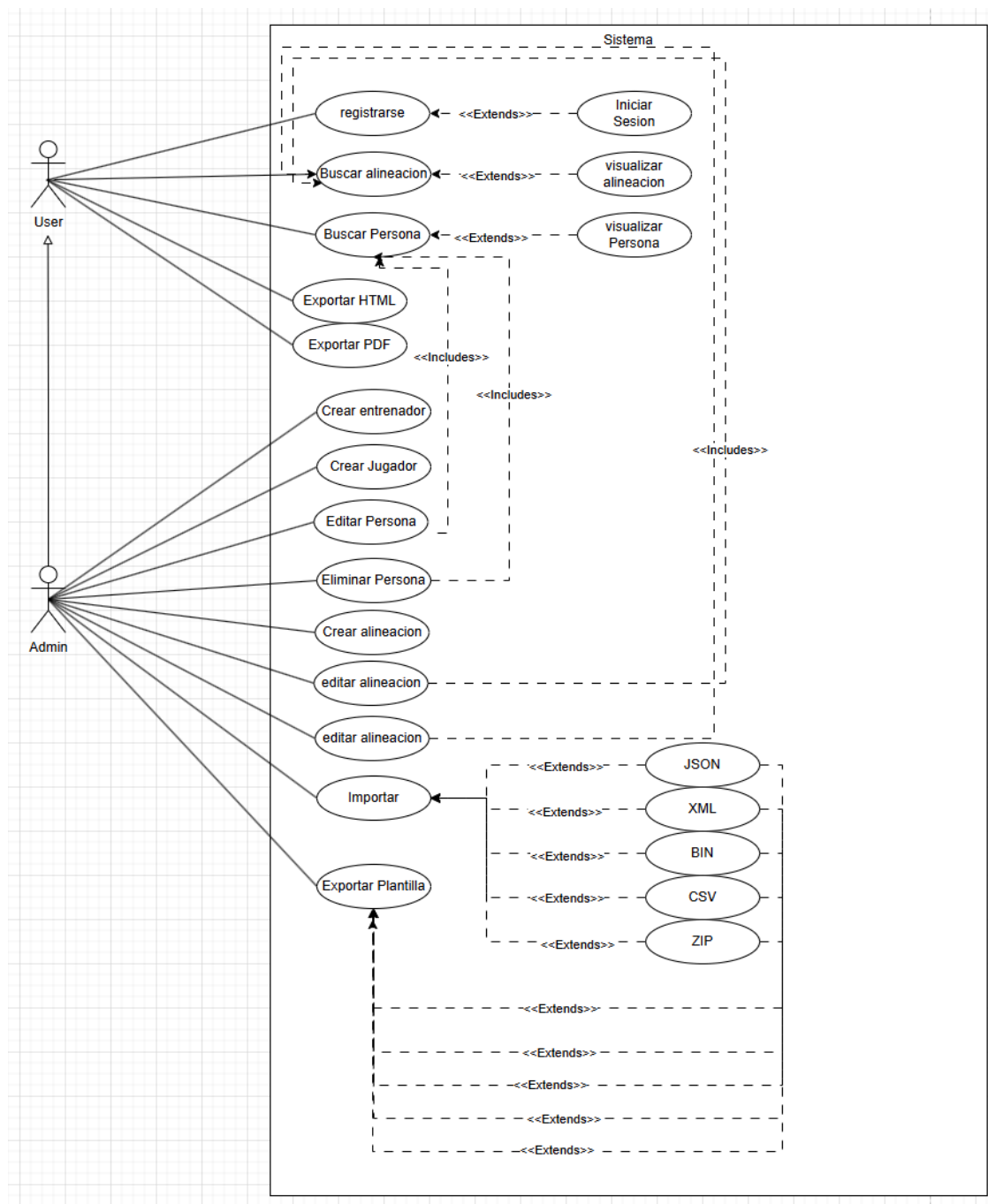
Nombre: de tipo String.

Fecha de fundación: de tipo LocalDate para que las fechas respeten el estándar ISO-8601.

Escudo: imagen en formato PNG.



3.- Diagrama de casos de uso



4.- Casos de uso

4.1.- Alta de miembro

Nombre del caso de Uso	Crear Jugador
Id	CU-1
Descripción	El administrador crea un Jugador y lo inserta en la base de datos
Actores Implicados	Admin
Precondiciones	Se requiere que el admin esté registrado en el sistema
Pasos / curso normal	<ol style="list-style-type: none">1. El admin solicita Crear Jugador2. El sistema le solicita los datos3. El admin introduce los datos4. Se validan los datos5. SI los datos son correctos<ol style="list-style-type: none">a. El sistema guarda el producto6. SI NO son correctos<ol style="list-style-type: none">a. El sistema notifica que los datos son incorrectosb. el sistema cancela la operación
Postcondiciones	El jugador queda insertado en el sistema y queda preparado para su consulta
Alternativa	Ninguna

Nombre del caso de Uso	Crear Entrenador
Id	CU-2
Descripción	El administrador crea un Entrenador y lo inserta en la base de datos
Actores Implicados	Admin
Precondiciones	Se requiere que el admin esté registrado en el sistema
Pasos / curso normal	<ol style="list-style-type: none">1. El admin solicita Crear Entrenador2. El sistema le solicita los datos3. El admin introduce los datos4. Se validan los datos5. SI los datos son correctos<ol style="list-style-type: none">a. El sistema guarda el producto6. SI NO son correctos<ol style="list-style-type: none">a. El sistema notifica que los datos son



	<p>incorrectos</p> <p>b. el sistema cancela la operación</p>
Postcondiciones	El Entrenador queda insertado en el sistema y queda preparado para su consulta
Alternativa	El sistema no es capaz de conectar con la BBDD

4.2.- Modificación de miembro

Nombre del caso de Uso	Editar Persona
Id	CU-3
Descripción	El administrador edita la información de una persona y lo guarda en la base de datos
Actores Implicados	Admin
Precondiciones	Se requiere que el admin esté registrado. Se requiere que haya una persona seleccionada.
Pasos / curso normal	<ol style="list-style-type: none"> 1. El admin solicita Editar Persona 2. El sistema le solicita los cambios 3. El admin introduce los cambios 4. Se validan los datos 5. SI los datos son correctos <ol style="list-style-type: none"> a. El sistema guarda el producto 6. SI NO son correctos <ol style="list-style-type: none"> a. El sistema notifica que los datos son incorrectos b. El sistema cancela la operación
Postcondiciones	La Persona queda actualizada en el sistema y queda preparada para su consulta.
Alternativa	El sistema no es capaz de conectar con la BBDD. La persona actualizada no es encontrada en la BBDD



4.3.- Baja de miembro

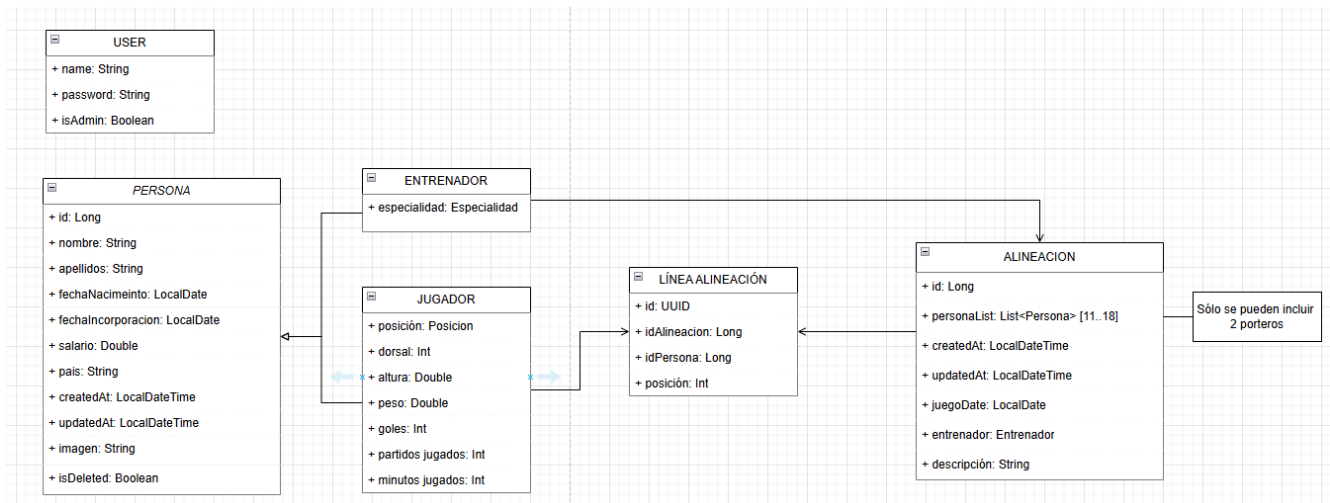
Nombre del caso de Uso	Eliminar Persona
Id	CU-4
Descripción	El administrador deshabilita a una persona de la base de datos
Actores Implicados	Admin
Precondiciones	Se requiere que el admin esté registrado. Se requiere que haya una persona seleccionada.
Pasos / curso normal	1. El admin solicita eliminar una persona
Postcondiciones	La persona queda eliminada del sistema
Alternativa	El sistema no es capaz de conectar con la BBDD. La persona actualizada no es encontrada en la BBDD

4.4.- Realizar convocatoria

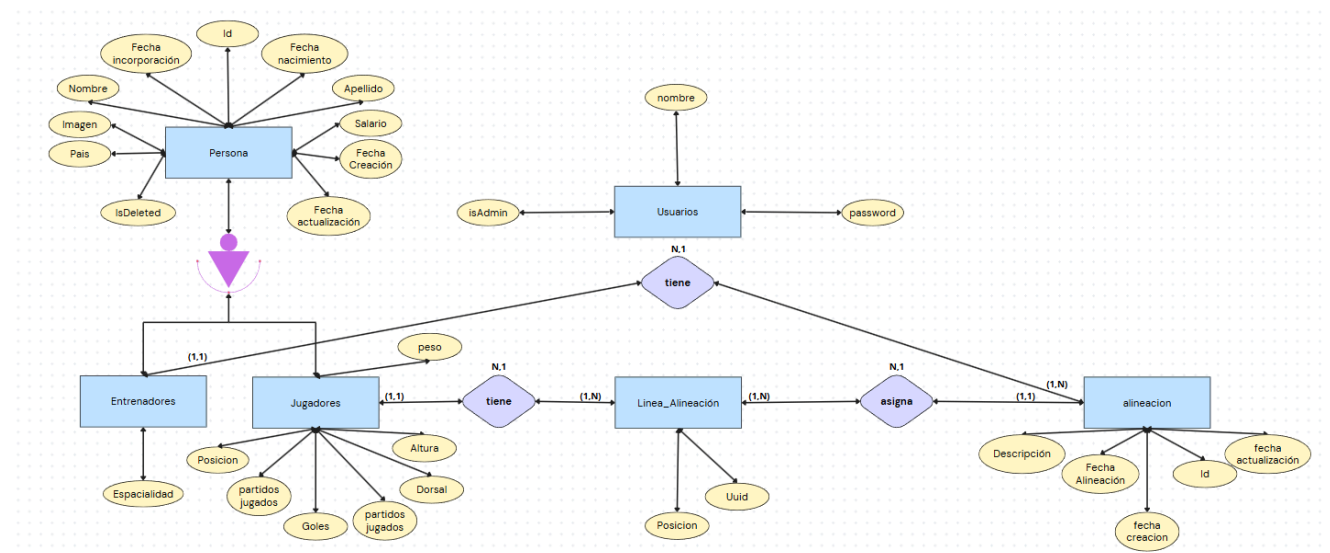
Nombre del caso de Uso	Crear Alineación
Id	CU-5
Descripción	El administrador crea e inserta una alineación en la BBDD
Actores Implicados	Admin
Precondiciones	Se requiere que el admin este registrado
Pasos / curso normal	<ol style="list-style-type: none">1. El admin solicita Crear Jugador2. El sistema le solicita los datos3. El admin introduce los datos4. Se validan los datos5. SI los datos son correctos<ol style="list-style-type: none">a. El sistema guarda el producto6. SI NO son correctos<ol style="list-style-type: none">a. El sistema notifica que los datos son incorrectosb. el sistema cancela la operación
Postcondiciones	La alineación queda insertada en el sistema y queda preparada para su consulta
Alternativa	El sistema no es capaz de conectar con la BBDD



5.- Diagrama de clases del modelo de negocio



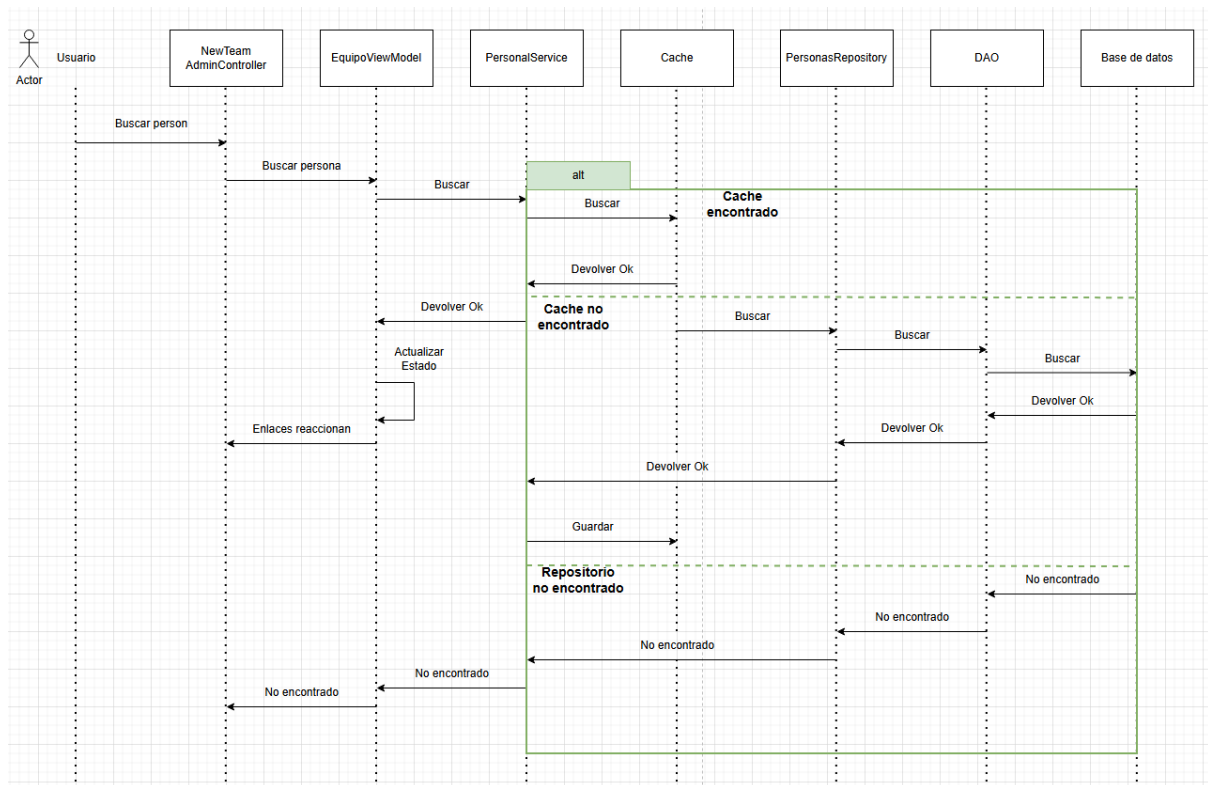
6.- Modelo entidad-relación y tablas resultantes



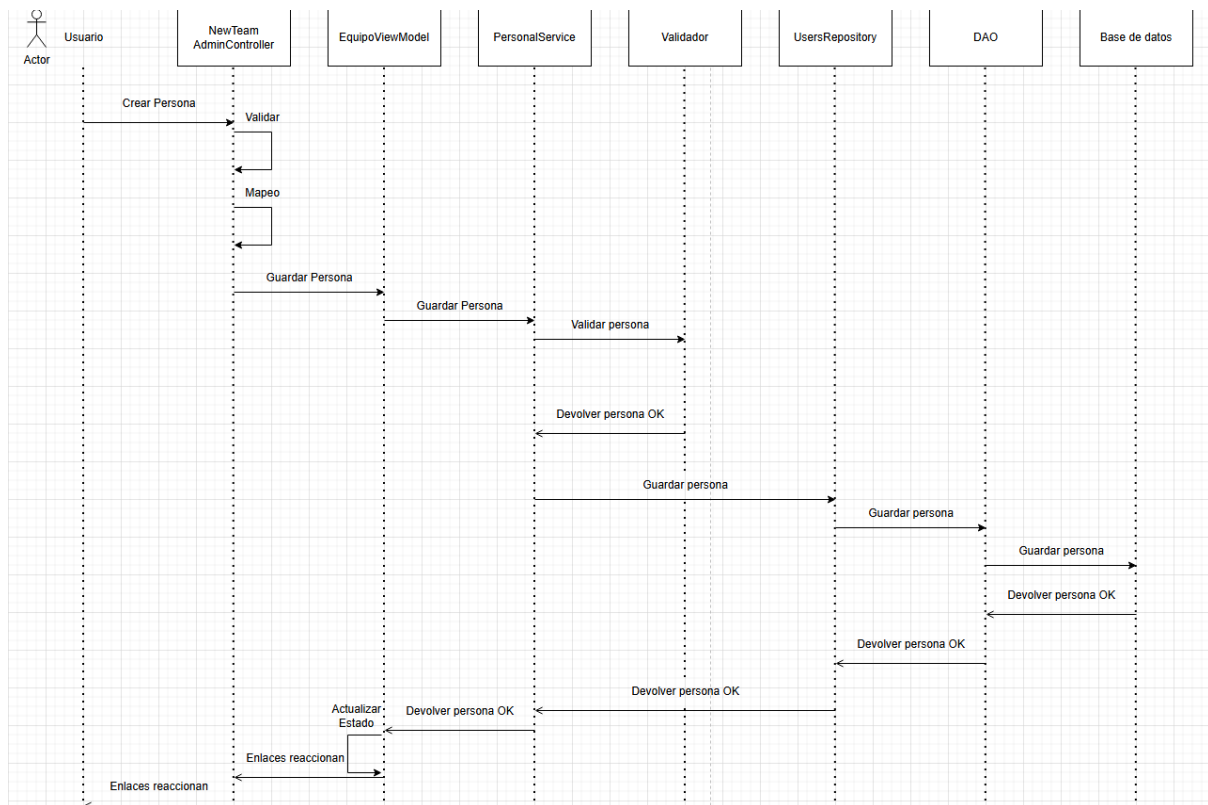
7.- Diagramas de secuencia



7.1.- Buscar miembro

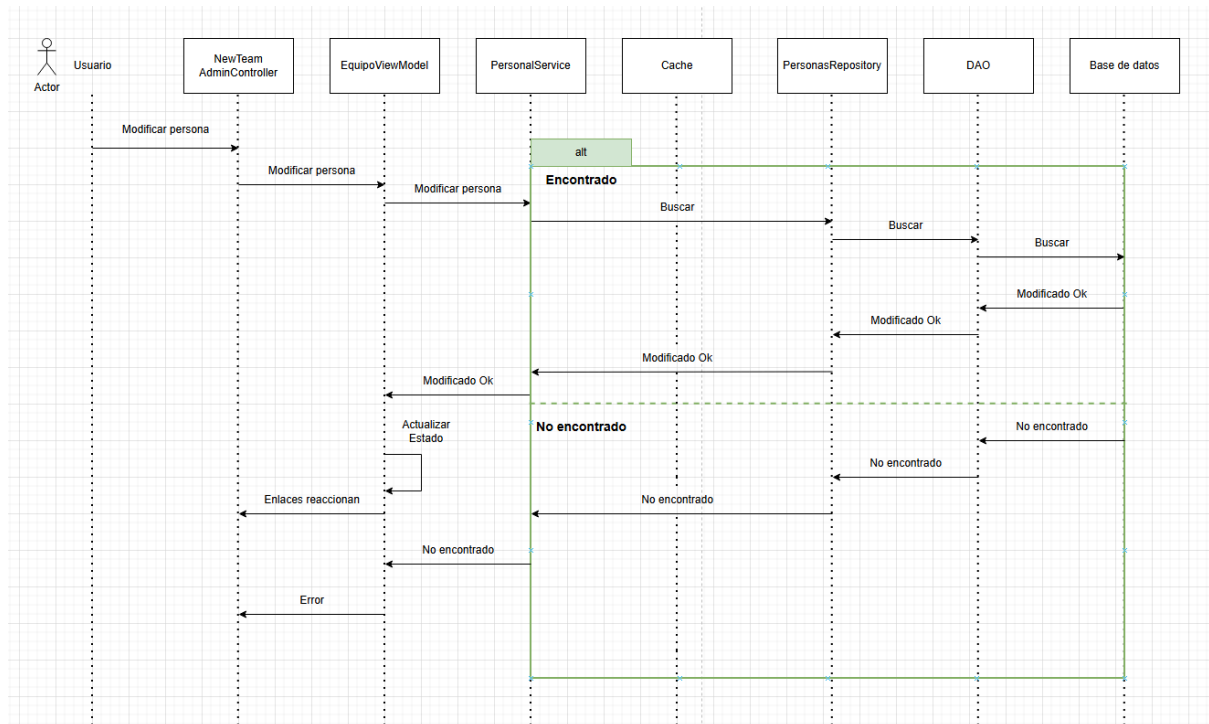


7.2.- Alta de miembro

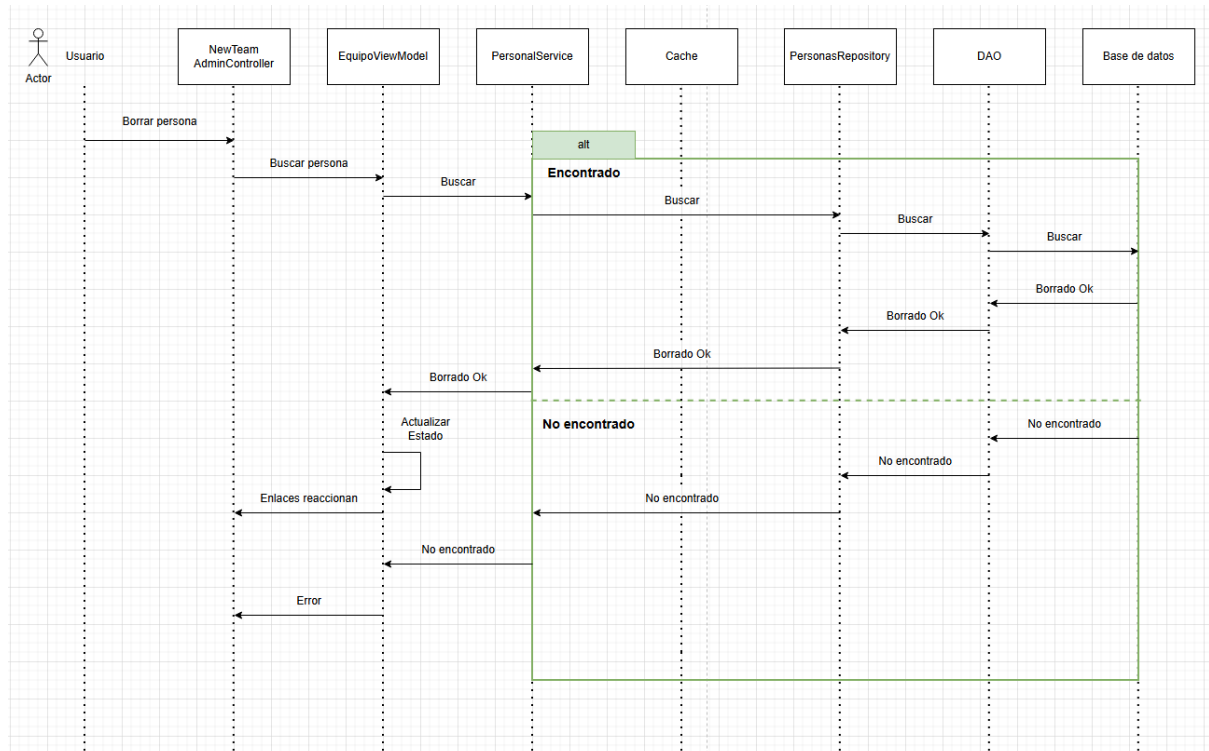


7.3.- Modificación de miembro



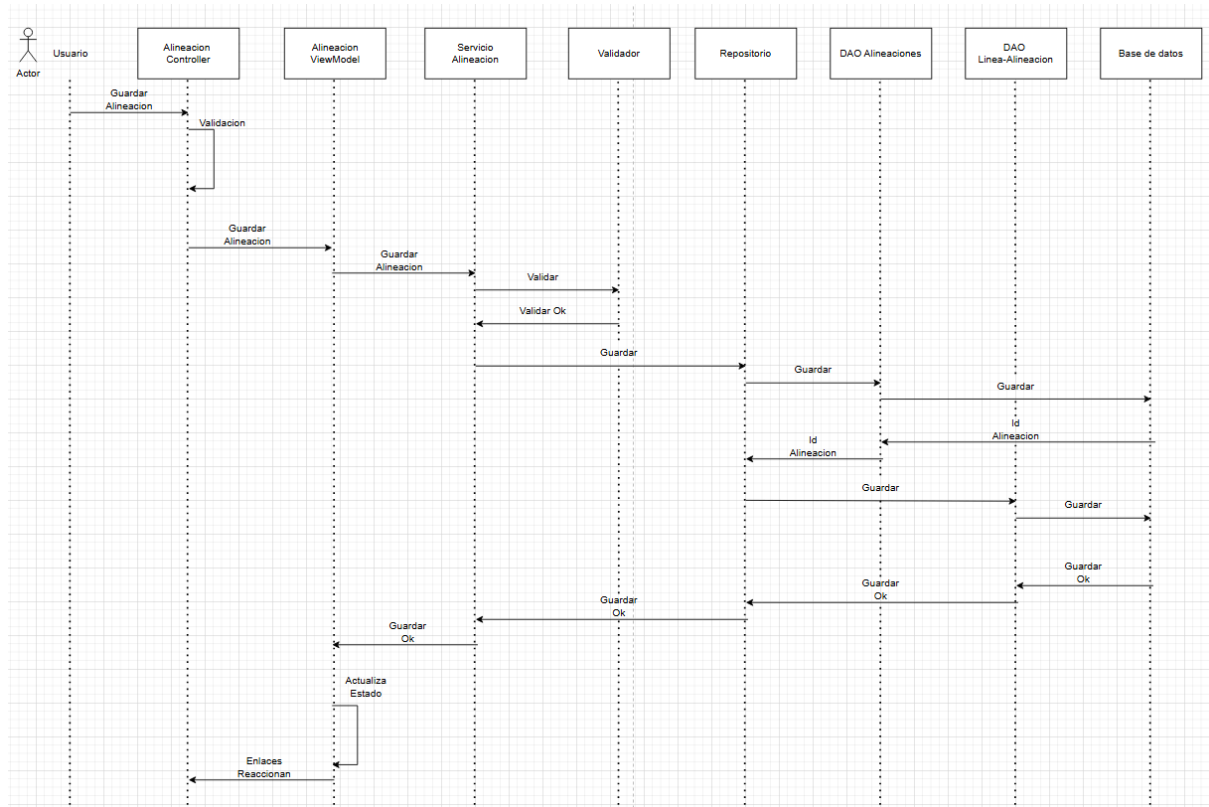


7.4.- Baja de miembro

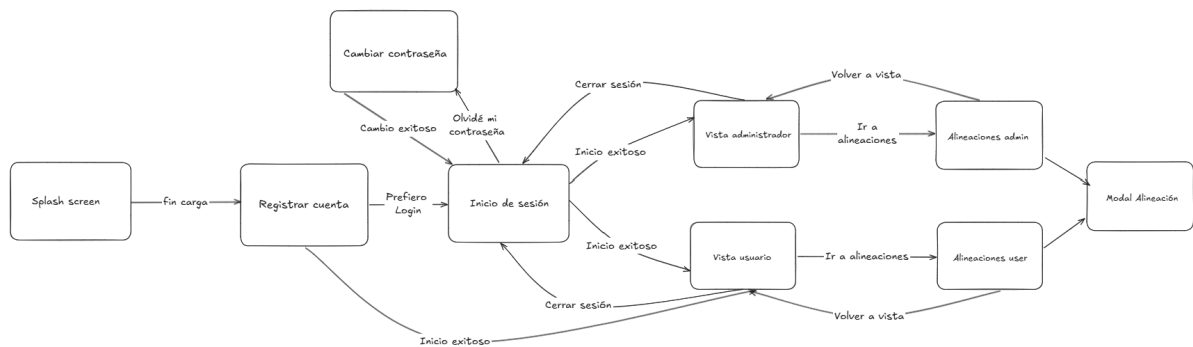


7.5.- Realizar convocatoria





8.- Grafo de navegación y diseño de vistas



Muestra las nueve vistas distintas de la aplicación y su modo de acceso.

Splash screen





Muestra la pantalla de carga de la aplicación, teniendo una barra de carga que muestra el progreso de la carga cuyo porcentaje se muestra en la parte inferior de esta.

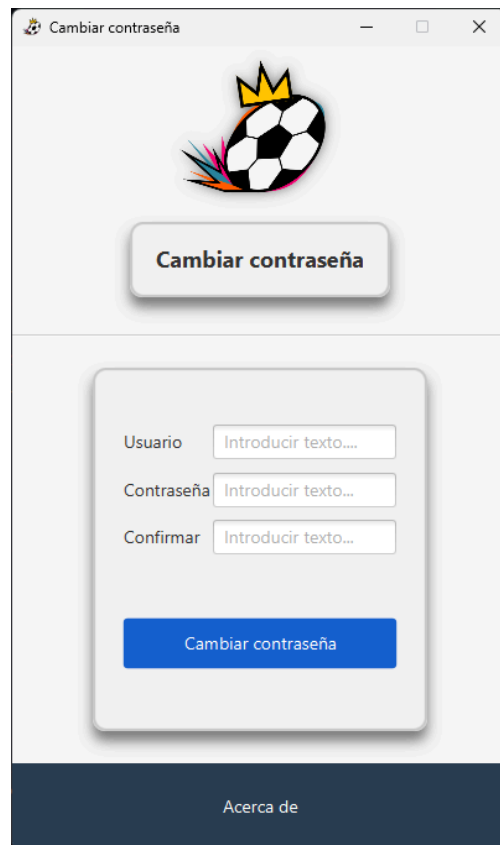
Registro de cuenta

Inicia sesión'. The footer of the window has a dark blue bar with the text 'Acerca de'."/>

Pantalla de registro de usuarios, necesita de confirmación de la contraseña para evitar equivocaciones a la hora de escribirla.

Cambio de contraseña





Vista de cambio de contraseña de la aplicación. Una vez se realice el cambio de contraseña exitosamente, redirigirá al usuario a la pantalla de inicio de sesión.

Acerca de



Vista "Acerca de" del proyecto. Con acceso a los perfiles de github de los desarrolladores y del repositorio de la aplicación.

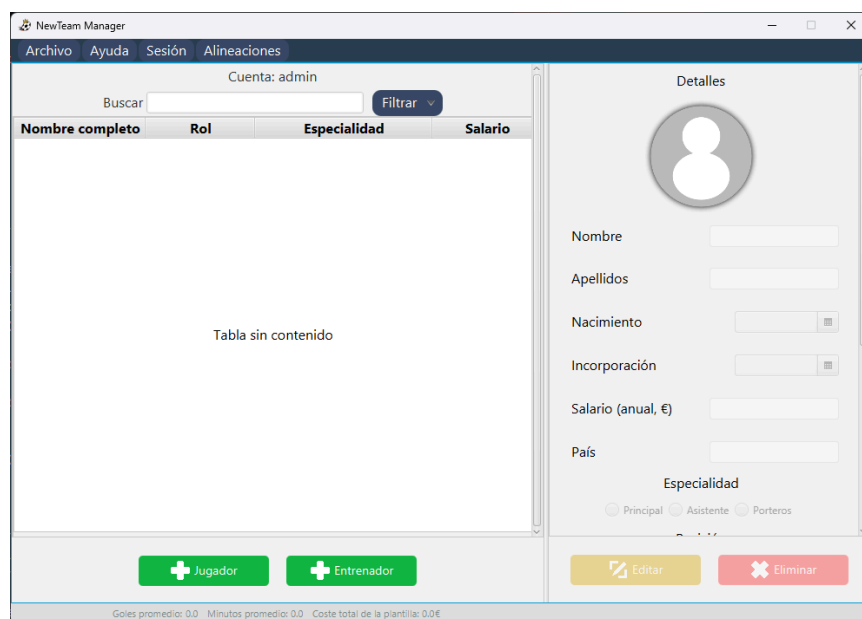
Vista de usuario





Vista de usuario de la pantalla principal de la aplicación. Desde esta se puede visualizar todo el personal del equipo que está en la base de datos de la aplicación. No es posible realizar todas las operaciones CRUD desde aquí, pero es posible ir a visualizar las alineaciones, cerrar sesión y salir de la aplicación.

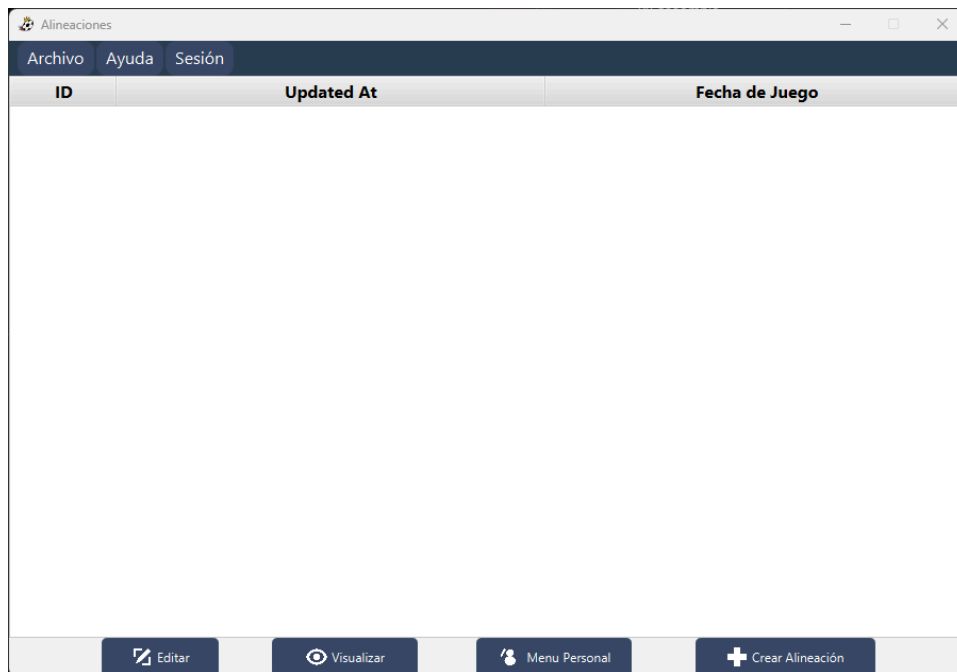
Vista de administrador



Desde esta vista no solo es posible visualizar, sino que también es posible realizar todas las operaciones CRUD sobre los jugadores y entrenadores del equipo. Además se puede realizar todo lo nombrado en la pantalla de Usuario.

Vista de alineaciones





Vista principal de administrador de alineaciones, permite realizar todas las operaciones CRUD sobre las alineaciones. La vista de usuario es la misma pero detecta los permisos de la sesión y desactiva o activa los botones de creación, edición y manipulación general de estas.

Vista de creación de alineación



Ventana modal sobre la que puedes construir tus alineaciones con los jugadores y entrenadores almacenados en la base de datos.

9.- Informes de cobertura y tests



Se han realizado un total de **253 tests**, tratando de cubrir la mayor cantidad de clases, funciones y ramas posibles:






Se ha utilizado la librería jacoco para obtener el reporte de la cobertura de las clases testeadas, obteniendo un resultado final del **98%** de cobertura total en la aplicación.

NewTeamUltimateEdition


Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes	
org.example.newteamultimateedition.personal.storage	<div><div></div></div>	95 %	<div><div></div></div>	78 %	23	88	9	274	0	30	0	14	
org.example.newteamultimateedition.personal.extensions	<div><div></div></div>	95 %		n/a	1	5	3	64	1	5	0	1	
org.example.newteamultimateedition.alineacion.storage	<div><div></div></div>	99 %	<div><div></div></div>	100 %	0	11	0	74	0	7	0	3	
org.example.newteamultimateedition.personal.services	<div><div></div></div>	99 %	<div><div></div></div>	95 %	1	20	0	73	0	8	0	1	
org.example.newteamultimateedition.personal.mapper	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0	17	0	241	0	12	0	1	
org.example.newteamultimateedition.alineacion.service	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0	23	0	89	0	11	0	1	
org.example.newteamultimateedition.alineacion.repository	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0	25	0	58	0	7	0	1	
org.example.newteamultimateedition.personal.repository	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0	15	0	29	0	7	0	1	
org.example.newteamultimateedition.users.service	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0	12	0	38	0	7	0	1	
org.example.newteamultimateedition.personal.validator	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0	19	0	37	0	3	0	1	
org.example.newteamultimateedition.alineacion.mapper	<div><div></div></div>	100 %		n/a	0	6	0	31	0	6	0	2	
org.example.newteamultimateedition.users.repository	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0	10	0	14	0	6	0	1	
org.example.newteamultimateedition.alineacion.dao	<div><div></div></div>	100 %		n/a	0	2	0	6	0	2	0	2	
org.example.newteamultimateedition.alineacion.validador	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0	4	0	6	0	2	0	1	
org.example.newteamultimateedition.users.mapper	<div><div></div></div>	100 %		n/a	0	3	0	9	0	3	0	1	
org.example.newteamultimateedition.personal.dao	<div><div></div></div>	100 %		n/a	0	1	0	3	0	1	0	1	
org.example.newteamultimateedition.users.dao	<div><div></div></div>	100 %		n/a	0	1	0	3	0	1	0	1	
org.example.newteamultimateedition.common.locale	<div><div></div></div>	100 %		n/a	0	2	0	5	0	2	0	1	
Total		97 of 6.735	98 %	25 of 286	91 %	25	264	12	1.054	1	120	0	35


10.- Estimación de costes



Requisitos funcionales 		
RF 	Nombre 	Horas estimadas 
RF1	Operaciones CRUD (pantilla)	4
RF2	Id único	0,5
RF3	Modelos (jugador y entrenador)	2
RF4	Campos comunes	1
RF5	Campos específicos jugador	2
RF6	Campos específicos entrenador	2
RF7	Validador de datos	4
RF8	Importación y exportación CSV, JSON, XML, BIN, HTML, PDF	16
RF9	Cache LRU	5
RF10	Interfaz de usuario	25
RF11	Splash screen	4
RF12	Inicio de sesión Bcrypt	2
RF13	Vistas admin y usuario	10
RF14	Estadísticas (goles, minutos, salarios)	3
RF15	Registro de usuarios	5
RF16	Admin creado mediante script	5
RF17	Operaciones CRUD (convocatorias)	8
RF18	Copia de seguridad completa del sistema	4
		102,5



Requisitos no funcionales ▾ 		
RNF ▾	Nombre ▾	Horas estimadas ▾
RNF1	Validación de datos	2
RNF2	Rapidez de acceso mediante caché	4
RNF3	Facilidad de uso del menú	3
RNF4	Logs para debug	2
RNF5	Documentación del código	7
RNF6	Tests	30
RNF7	.jar ejecutable	2
RNF8	Uso de GitFlow/Git/Pull Request	15
RNF9	Instalador de la aplicación	5
		70

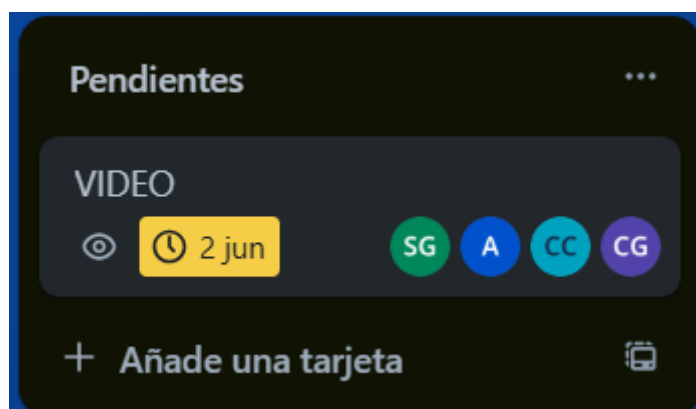
Requisitos de información ▾ 		
RI ▾	Nombre ▾	Horas estimadas ▾
RI1	Campos comunes	5
RI2	Campos de jugador	6
RI3	Campos de entrenador	1
RI4	Campos y condiciones de las convocatorias	10
RI5	Información del equipo	1
		23

HORAS DE TRABAJO TOTALES	
	195,5
PRECIO POR HORA DE TRABAJO	
	135,00 €
PRECIO POR HORA EXTRA DE TRABAJO	
	160,00 €
COSTE TOTAL DE LAS HORAS DE TRABAJO	
	26.392,50 €

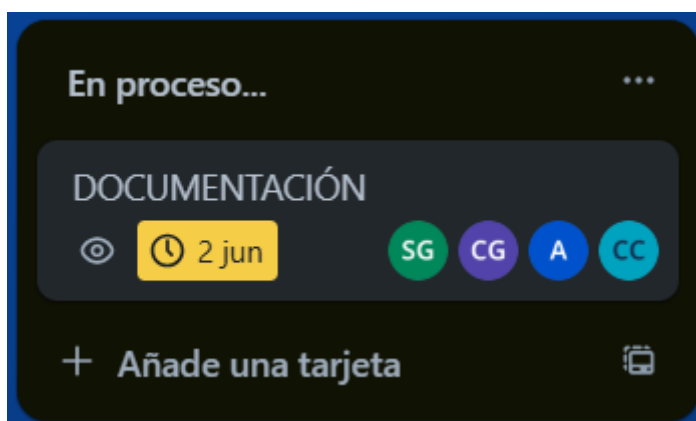


ESTIMACIÓN DE COSTES				
Nº	Concepto	Desglose		Coste
		Precio por hora	Horas	
1	Horas de trabajo estimadas por requisitos	135,00 €	195,5	26.392,50 €
		Precio por hora extra	15% de las horas totales	
2	Horas extra para cubrir imprevistos	160,00 €	29,33	4.692,00 €
		Costes antes de margen	17% sobre los costes	
3	Margen de beneficios	31.084,50 €	0,17	5.284,37 €
				36.368,87 €

11.-Trello

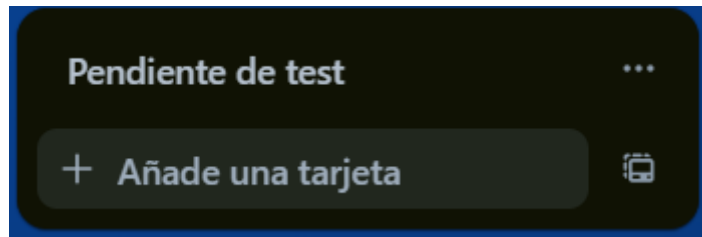


Aquí teníamos las cosas pendientes por hacer, es decir, lo que todavía no habíamos empezado. Cuando empezábamos a realizar una tarea, la pasábamos a las siguientes listas.

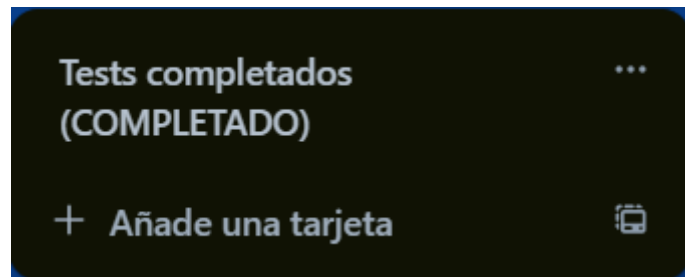


Esta lista es a la que pasamos las tareas que habíamos empezado, pero que todavía no están acabadas, aquí le ponemos una fecha a la que queremos que esté acabado y las personas las cuales están destinadas a hacer la tarea.

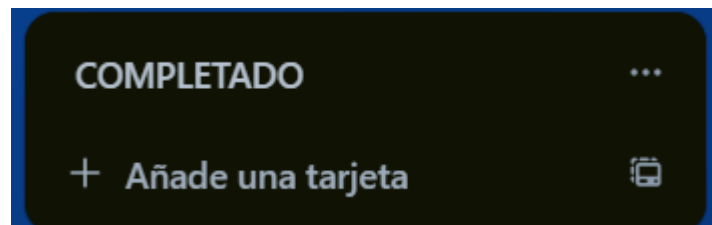




También tenemos esta lista, la cual cuando empezábamos a hacer los test, movíamos las tarjetas del test correspondiente que se estaba realizando a esta misma. Aquí también ponemos la fecha a la que queríamos que estuviese acabado de testear, y la persona la cual lo realizaba.

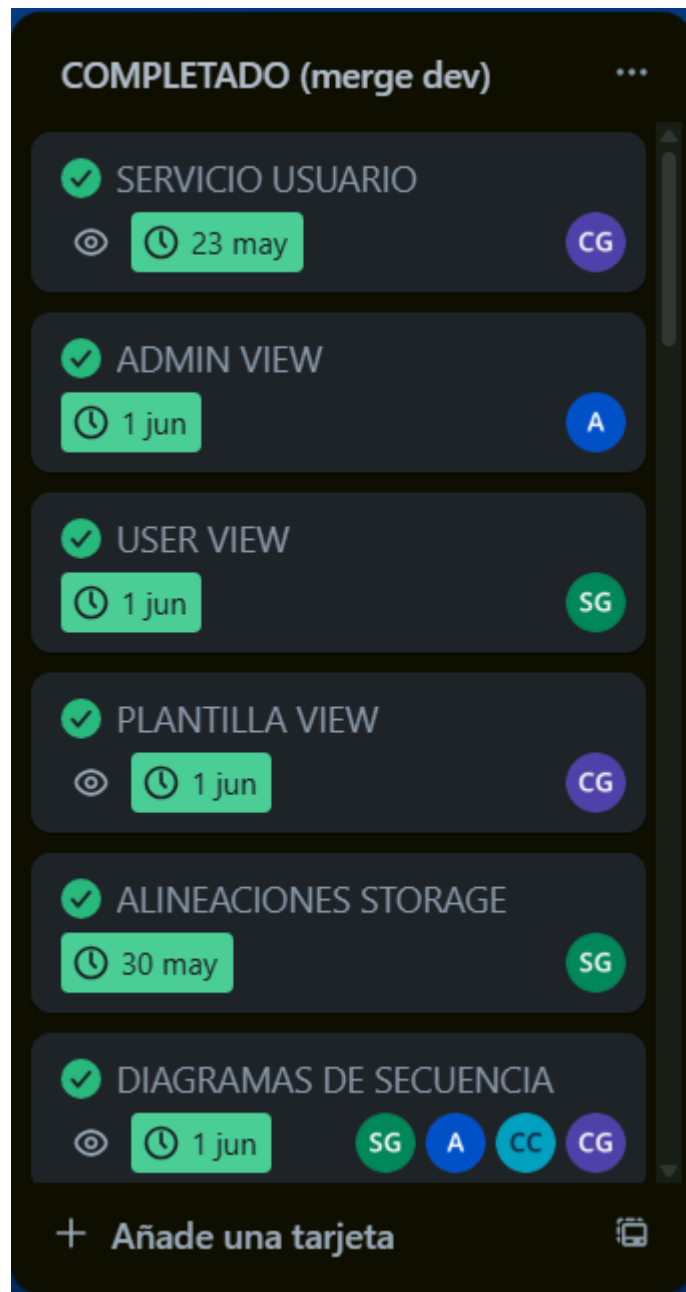


Cuando acabamos dichos test, se pasaban a la lista de test completados que podemos observar, aquí ya no se ponía la fecha si no que se marcaba el botón el cual se ponía verde para indicar que estaba realizado correctamente. También estaba la persona la cual lo había realizado.



En este caso no teníamos los test, si no que teníamos las tareas como las views, los service, los storage y todo lo demás realizado correctamente. Lo indicamos marcándolo en verde y con la persona correspondiente que ha realizado dicha tarea.





Por último, cuando el programa ya estaba todo bien realizado, se hace un merge a la rama dev, y aquí tenemos todo el programa realizado correctamente y con éxito.





Metele un clic a la foto si la quieres ver con más calidad

enlaces:
[GitHub](#)
[Youtube](#)

