CLUB BALONCESTO LA ARBOLEDA

Javier Cordero Toscao

Desarrollo de Aplicaciones Web

## Resumen

El objetivo del proyecto es crear una aplicación que permita la gestión de la cantera de un club de baloncesto, teniendo dos aplicaciones claramente diferenciadas dentro del proyecto, una página web pública a todo el mundo y una aplicación de gestión del club.

La página web pública tiene como objetivo informar sobre el club, mostrando los equipos, entrenadores y jugadores que pertenecen al club. Además, se muestran los partidos ya jugados, con las estadísticas, y los partidos por jugar.

La aplicación administrativa permite gestionar las diferentes entidades que forman un club: los equipos, los entrenadores y los jugadores.

La aplicación permite, también, registrar los partidos que juegan los equipos del club y registrar las estadísticas de los jugadores que han participado en ellos.

Además, guarda un registro histórico de todos los equipos por los que han ido pasando los diferentes jugadores y entrenadores que pertenecen al club.

La aplicación tiene tres tipos de usuarios diferentes: administrador, entrenador y jugador.

El administrador es el encargado de gestionar las altas, bajas y modificaciones de los equipos, jugadores y entrenadores. Además, podrá gestionar los partidos de todos los equipos del club.

El entrenador es el encargado de gestionar sus equipos, con todo lo que conlleva: convocatorias a partidos, evolución de jugadores, calendario de entrenamientos, …

El jugador puede enviar y recibir mensajes de los otros dos tipos de usuarios para estar informado de las novedades.

Índice

[Resumen 2](#_Toc74773529)

[1. Introducción 4](#_Toc74773530)

[1.1. Motivación del proyecto 4](#_Toc74773531)

[1.2. Objetivos 4](#_Toc74773532)

[2. Estado del arte 4](#_Toc74773533)

[3. Análisis 4](#_Toc74773534)

[3.1. Usuarios de la aplicación 5](#_Toc74773535)

[3.2. Requisitos funcionales 5](#_Toc74773536)

[3.3. Casos de uso 6](#_Toc74773537)

[4. Diseño 18](#_Toc74773538)

[4.1. Base de datos 18](#_Toc74773539)

[5. Tecnologías utilizadas 21](#_Toc74773540)

[5.1. Lenguajes 21](#_Toc74773541)

[5.2. Base de datos 22](#_Toc74773542)

[5.3. Frameworks y librerías 22](#_Toc74773543)

[5.4. Control de versiones 22](#_Toc74773544)

[5.5. Software 23](#_Toc74773545)

[6. Desarrollo 24](#_Toc74773546)

[6.1. Estructura 24](#_Toc74773547)

# Introducción

## Motivación del proyecto

La idea del proyecto nace de mi trabajo como entrenador en la cantera del Club Baloncesto Lepe Alius. Durante mi estancia en el club he observado que la gestión se lleva mediante unos métodos bastante anticuados: hojas de Excel, documentos de Word y anotaciones a mano. Gracias a la aplicación desarrollada podremos aunar toda la documentación, evitando así la pérdida de información que puede suponer el tener esta documentación en diferentes archivos. Además, modernizar y hacer más sencillas las tareas que conllevan la gestión de un club.

## Objetivos

El objetivo del proyecto es crear una aplicación que permita la gestión de la cantera de un club de baloncesto y una página web pública que informe sobre el club. Los principales objetivos del proyecto son los siguientes:

* Gestionar los datos de las entidades que componen el club: equipos, entrenadores y jugadores.
* Asignar jugadores y entrenadores a los equipos.
* Guardar un histórico de los equipos por los que han pasado los jugadores y entrenadores.
* Gestionar los horarios de entrenamientos de los equipos.
* Gestionar el calendario de partidos de los equipos.
* Guardar las estadísticas de los jugadores en los partidos disputados por los equipos.
* Registrar la evolución de los jugadores.
* Dotar a los componentes del club de una herramienta de comunicación interna.
* Mostrar la información del club en una página web estética y accesible.

# Estado del arte

# Análisis

El análisis del proyecto nos sirve para examinar cada uno de los puntos que componen el proyecto teniendo en cuenta las necesidades del mismo.

## Usuarios de la aplicación

La necesidad de diferenciar roles dentro de una aplicación se debe, sobre todo, a la seguridad de la aplicación. Necesitamos delimitar los niveles de acceso, de manera que cada usuario pueda manipular la parte del programa que debe. A continuación, detallamos los diferentes roles que existen en la aplicación.

|  |  |
| --- | --- |
| **Administrador** | Tiene permiso total para gestionar las altas, bajas y modificaciones de las diferentes entidades del proyecto. Encargado de crear el resto de usuarios de la aplicación |
| **Entrenador** | Tiene permiso para gestionar sus equipos. |
| **Jugador** | Tiene permiso para enviar y recibir mensajes. |

## Requisitos funcionales

### 3.2.1. Gestión de entrenadores

* Un administrador podrá dar de alta y modificar los entrenadores.
* Un administrador podrá asignar entrenadores a diferentes equipos.

### 3.2.2. Gestión de jugadores

* Un administrador podrá dar de alta y modificar los jugadores.
* Un entrenador podrá actualizar los atributos de los jugadores pertenecientes a sus equipos.

### 3.2.4. Gestión de equipos

* Un administrador podrá dar de alta, baja y modificar los equipos.
* Un administrador podrá gestionar los horarios de todos los equipos del club.
* Un entrenador podrá gestionar los horarios de sus equipos.
* Un administrador podrá asignar y quitar jugadores de un equipo.
* Un administrador podrá mover jugadores de un equipo a otro.
* Un entrenador podrá asignar y quitar jugadores a sus equipos.

### 3.2.4. Gestión de partidos

* Un administrador podrá dar de alta y baja partidos de todos los equipos del club.
* Un entrenador podrá dar de alta y baja partidos de sus equipos.
* Un entrenador podrá hacer la convocatoria de jugadores para los partidos de sus equipos.

## Casos de uso

El modelo de casos de uso especifica la funcionalidad que el sistema ha de ofrecer desde la perspectiva de los usuarios y lo que el sistema ha de realizar para satisfacer las peticiones de estos usuarios. Se han identificado los siguientes Casos de Uso según los tipos de usuario que pueden acceder a la aplicación.

### 3.1.1. Actor Visitante

No es un usuario de los definidos anteriormente, sino que corresponde a cualquier persona que visite la página web.

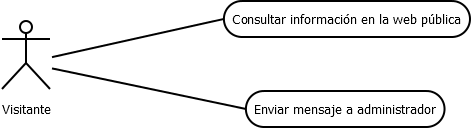


Figura 3.1 – Casos de uso Visitante

Descripción de los casos de uso

* **Consultar información en la web pública:** El usuario navegar por la página web pública, pudiendo ver datos como el calendario de partidos, resultados de partidos, listado de equipos y listado de jugadores.
  + **Precondiciones:** No aplica.
  + **Postcondiciones:** No aplica.
* **Enviar mensaje a administrador:** El usuario puede enviar un mensaje al administrador desde el formulario de contacto, aportando su nombre y email.
  + **Precondiciones:** No aplica.
  + **Postcondiciones:** Se guarda un registro en la tabla Usuario-Mensajes.

### 3.1.2. Actor Jugador

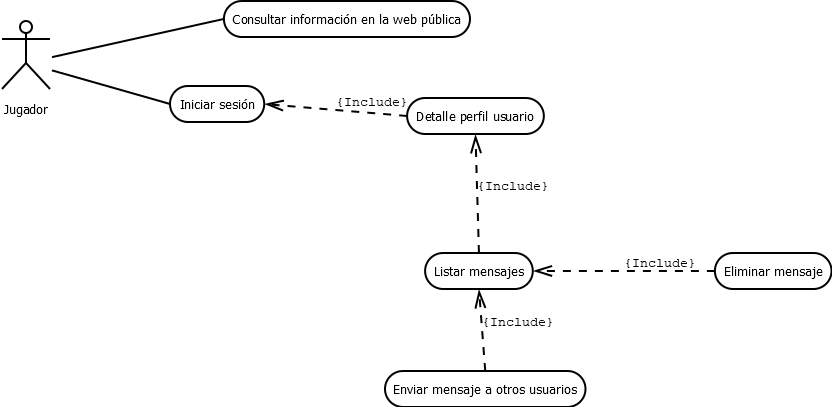


Figura 3.2 – Casos de uso Jugador

Descripción de los casos de uso

* **Consultar información en la web pública:** El usuario navegar por la página web pública, pudiendo ver datos como el calendario de partidos, resultados de partidos, listado de equipos y listado de jugadores.
  + **Precondiciones:** No aplica.
  + **Postcondiciones:** No aplica.
* **Iniciar sesión:** El usuario puede iniciar sesión en la aplicación con su nombre de usuario y contraseña.
  + **Precondiciones:** No aplica.
  + **Postcondiciones:** El usuario queda logueado en la aplicación.
* **Detalle perfil usuario:** El usuario puede acceder a su página de detalle de perfil de usuario, a modo de consulta.
  + **Precondiciones:** 
    - El usuario debe estar logueado.
  + **Postcondiciones:** No aplica.
* **Listar mensajes:** El usuario puede listar sus mensajes recibidos.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir mensajes recibidos por el usuario.
  + **Postcondiciones:** No aplica.
* **Enviar mensaje a otros usuarios:** El usuario puede enviar mensaje a uno o varios usuarios de la aplicación.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir usuarios en la aplicación.
  + **Postcondiciones:** Se guarda un registro en la tabla Usuario-Mensajes.
* **Eliminar mensaje:** El usuario puede eliminar cualquiera de sus mensajes recibidos.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir mensajes recibidos por el usuario.
  + **Postcondiciones:** Se elimina el registro de la tabla Usuario-Mensajes.

### 3.1.3. Actor Entrenador

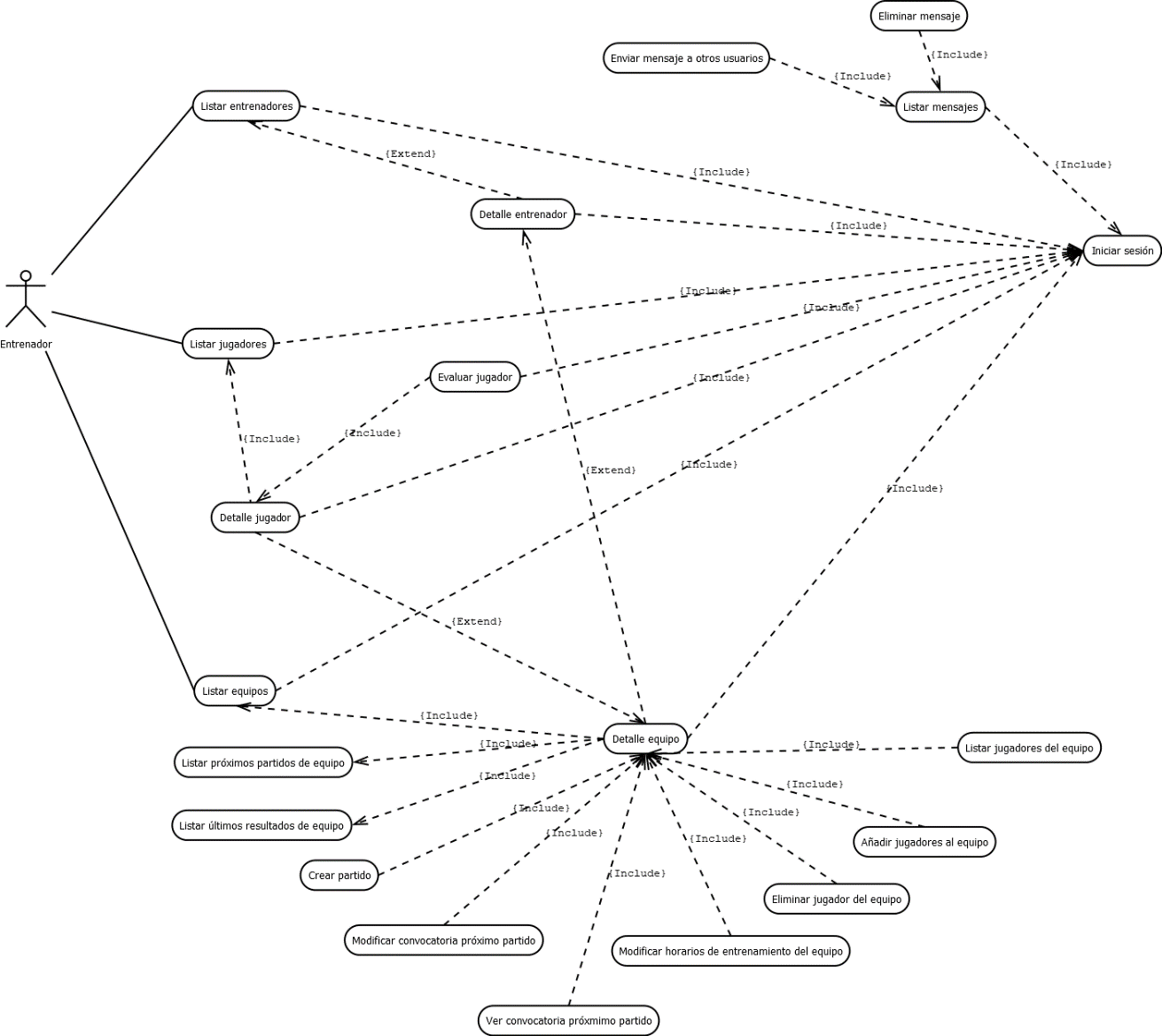


Figura 3.3 – Casos de uso Entrenador

Descripción de los casos de uso

* **Iniciar sesión:** El usuario puede iniciar sesión en la aplicación con su nombre de usuario y contraseña.
  + **Precondiciones:** No aplica.
  + **Postcondiciones:** El usuario queda logueado en la aplicación.
* **Listar mensajes:** El usuario puede listar sus mensajes recibidos.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir mensajes recibidos por el usuario.
  + **Postcondiciones:** No aplica.
* **Enviar mensaje a otros usuarios:** El usuario puede enviar mensaje a uno o varios usuarios de la aplicación.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir usuarios en la aplicación.
  + **Postcondiciones:** Se guarda un registro en la tabla Usuario-Mensajes.
* **Eliminar mensaje:** El usuario puede eliminar cualquiera de sus mensajes recibidos.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir mensajes recibidos por el usuario.
  + **Postcondiciones:** Se elimina el registro de la tabla Usuario-Mensajes.
* **Listar entrenadores:** El usuario puede obtener un listado de los entrenadores, con los siguientes datos:
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir entrenadores.
  + **Postcondiciones:** No aplica.
* **Detalle entrenador:** El usuario puede acceder a la página de detalle de un entrenador. En la página de detalle se muestran los datos personales del entrenador, un historial de los equipos que ha entrenado y un listado de los equipos que ha entrenado.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir entrenadores.
  + **Postcondiciones:** No aplica.
* **Listar jugadores:** El usuario puede obtener un listado de los jugadores.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir jugadores.
  + **Postcondiciones:** No aplica.
* **Detalle jugador:** El usuario puede acceder a la página de detalle de un jugador. En la página de detalle se muestran los datos personales del entrenador, un historial de los equipos que ha entrenado y un listado de los equipos que ha entrenado.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir jugadores.
  + **Postcondiciones:** No aplica.
* **Evaluar jugador:** El usuario puede realizar una evaluación de todos los atributos del jugador.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir jugadores.
    - El jugador debe pertenecer a uno de los equipos que entrena el usuario.
  + **Postcondiciones:** Se crean nuevos registros en la tabla Jugadores-Atributos con fecha del momento de la evaluación.
* **Listar equipos:** El usuario puede obtener un listado de los equipos
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
  + **Postcondiciones:** No aplica.
* **Detalle equipo:** El usuario puede acceder a la página de detalle de un equipo. En la página de detalle se muestra un listado de los jugadores que pertenecen al equipo, los horarios de entrenamiento, la convocatoria para el próximo partido, los últimos resultados y los próximos partidos.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
  + **Postcondiciones:** No aplica.
* **Listar próximos partidos de equipo**: El usuario puede obtener un listado de los próximos partidos que tiene por jugar el equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener partidos por jugar.
  + **Postcondiciones:** No aplica.
* **Listar últimos resultados de equipo**: El usuario puede obtener un listado de los últimos resultados en los partidos del equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener partidos jugados.
  + **Postcondiciones:** No aplica.
* **Detalle partido**: El usuario puede acceder a la página de detalle de un partido en la que se muestra el resultado y las estadísticas de los jugadores que participaron en él.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener partidos jugados.
  + **Postcondiciones:** No aplica.
* **Crear partido**: El usuario puede crear un partido para el equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - Deben existir rivales.
    - El usuario debe ser entrenador del equipo.
  + **Postcondiciones:** Se crea un nuevo registro en la tabla partidos.
* **Ver convocatoria próximo partido**: El usuario puede ver un listado de los jugadores convocados para el próximo partido.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener partidos por jugar.
    - Debe haber registros en la tabla Jugadores-Partidos para el próximo partido.
  + **Postcondiciones:** No aplica.
* **Modificar convocatoria próximo partido**: El usuario puede realizar y modificar la convocatoria de jugadores para el próximo partido del equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener partidos por jugar.
    - El usuario debe ser entrenador del equipo.
  + **Postcondiciones:** Se crea un registro en la tabla Jugadores-Partidos por cada jugador seleccionado para el próximo partido.
* **Modificar horarios de entrenamiento del equipo**: El usuario puede modificar los horarios de entrenamiento del equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El usuario debe ser entrenador del equipo.
  + **Postcondiciones:** Se actualizan los registros en la tabla Equipo-Dias\_de\_entrenamiento.
* **Listar jugadores del equipo**: El usuario puede obtener un listado de los jugadores que pertenecen actualmente al equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener jugadores.
  + **Postcondiciones:** No aplica.
* **Eliminar jugador del equipo**: El usuario puede eliminar un jugador del equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener jugadores.
    - El usuario debe ser entrenador del equipo.
  + **Postcondiciones:** Se actualiza el registro de la tabla Contratos para el contrato actual del jugador, asignándole fecha final al momento de la acción.
* **Añadir jugadores al equipo**: El usuario puede realizar y modificar la convocatoria de jugadores para el próximo partido del equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - Deben existir jugadores del mismo género y la misma categoría del equipo que no pertenezcan a ningún equipo
    - El usuario debe ser entrenador del equipo.
  + **Postcondiciones:** Se crea un registro en la tabla Contratos para los jugadores seleccionados, con fecha de comienzo del momento de la acción.

### 3.1.4. Actor Administrador

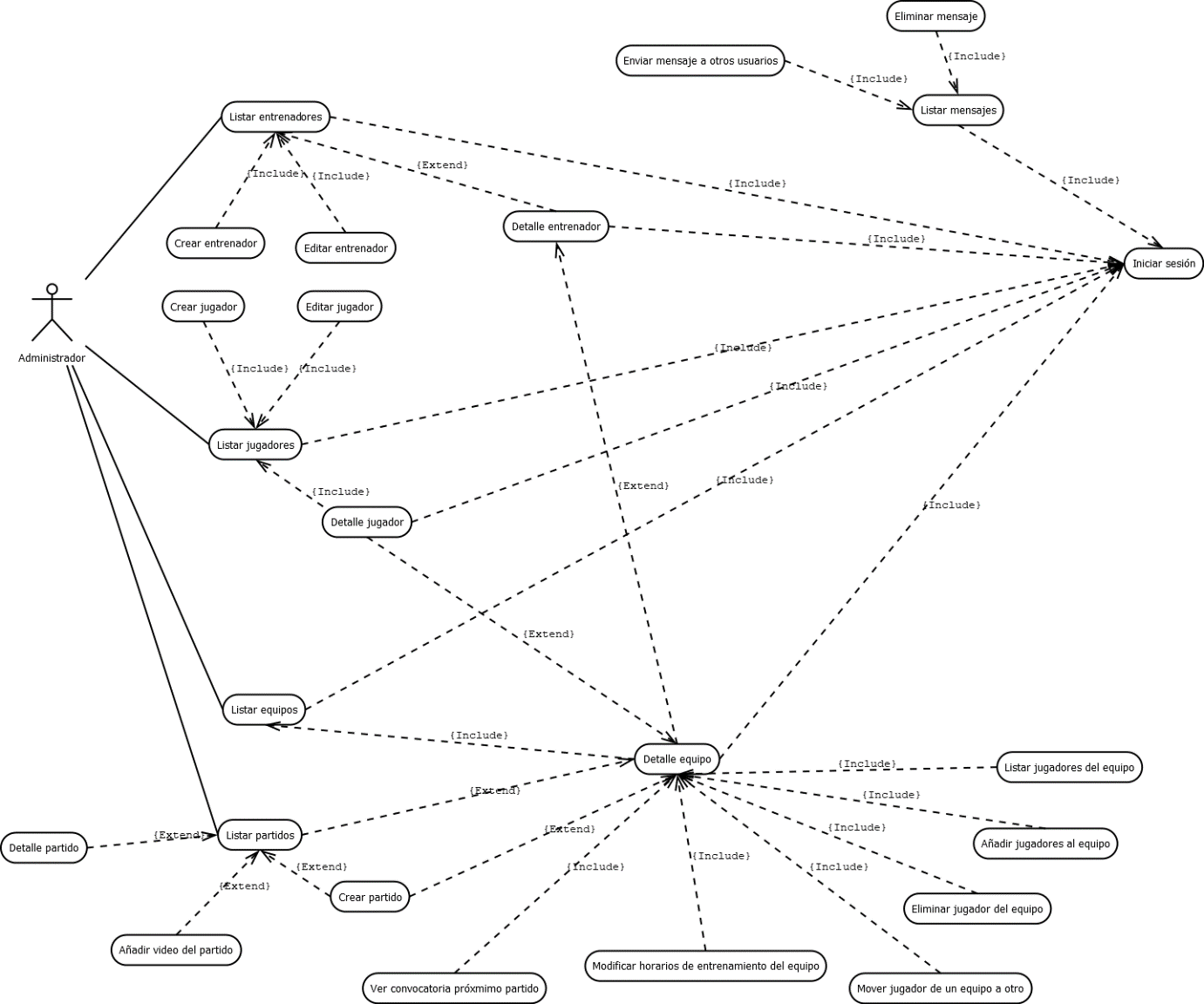


Figura 3.4 – Casos de uso administrador

Descripción de los casos de uso

* **Iniciar sesión:** El usuario puede iniciar sesión en la aplicación con su nombre de usuario y contraseña.
  + **Precondiciones:** No aplica.
  + **Postcondiciones:** El usuario queda logueado en la aplicación.
* **Listar mensajes:** El usuario puede listar sus mensajes recibidos.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir mensajes recibidos por el usuario.
  + **Postcondiciones:** No aplica.
* **Enviar mensaje a otros usuarios:** El usuario puede enviar mensaje a uno o varios usuarios de la aplicación.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir usuarios en la aplicación.
  + **Postcondiciones:** Se guarda un registro en la tabla Usuario-Mensajes.
* **Eliminar mensaje:** El usuario puede eliminar cualquiera de sus mensajes recibidos.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir mensajes recibidos por el usuario.
  + **Postcondiciones:** Se elimina el registro de la tabla Usuario-Mensajes.
* **Listar entrenadores:** El usuario puede obtener un listado de los entrenadores, con los siguientes datos:
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir entrenadores.
  + **Postcondiciones:** No aplica.
* **Crear entrenador:** El usuario puede crear un entrenador.
  + **Precondiciones:**
    - El usuario debe estar logueado.
  + **Postcondiciones:**
    - Se crea un registro en la tabla Entrenadores.
    - Se crea un registro en la tabla Personas.
    - Se crea un registro en la tabla Usuarios.
* **Editar entrenador:** El usuario puede editar los campos de un entrenador.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir entrenadores.
  + **Postcondiciones:**
    - Se modifican los valores del registro en la tabla Entrenadores.
* **Detalle entrenador:** El usuario puede acceder a la página de detalle de un entrenador. En la página de detalle se muestran los datos personales del entrenador, un historial de los equipos que ha entrenado y un listado de los equipos que ha entrenado.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir entrenadores.
  + **Postcondiciones:** No aplica.
* **Listar jugadores:** El usuario puede obtener un listado de los jugadores.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir jugadores.
  + **Postcondiciones:** No aplica.
* **Crear jugador:** El usuario puede crear un jugador.
  + **Precondiciones:**
    - El usuario debe estar logueado.
  + **Postcondiciones:**
    - Se crea un registro en la tabla Entrenadores.
    - Se crea un registro en la tabla Personas.
    - Se crea un registro en la tabla Usuarios.
* **Editar jugador:** El usuario puede editar los campos de un jugador.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir jugadores.
  + **Postcondiciones:**
    - Se modifican los valores del registro en la tabla Jugadores.
* **Detalle jugador:** El usuario puede acceder a la página de detalle de un jugador. En la página de detalle se muestran los datos personales del entrenador, un historial de los equipos que ha entrenado y un listado de los equipos que ha entrenado.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir jugadores.
  + **Postcondiciones:** No aplica.
* **Listar equipos:** El usuario puede obtener un listado de los equipos
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
  + **Postcondiciones:** No aplica.
* **Detalle equipo:** El usuario puede acceder a la página de detalle de un equipo. En la página de detalle se muestra un listado de los jugadores que pertenecen al equipo, los horarios de entrenamiento, la convocatoria para el próximo partido, los últimos resultados y los próximos partidos.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
  + **Postcondiciones:** No aplica.
* **Listar partidos**: El usuario puede obtener un listado de los próximos partidos del club.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener partidos por jugar.
  + **Postcondiciones:** No aplica.
* **Listar últimos resultados de equipo**: El usuario puede obtener un listado de los últimos resultados del club.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener partidos jugados.
  + **Postcondiciones:** No aplica.
* **Detalle partido**: El usuario puede acceder a la página de detalle de un partido en la que se muestra el resultado y las estadísticas de los jugadores que participaron en él.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir partidos en estado jugado.
    - El partido ha de estar en estado jugado.
  + **Postcondiciones:** No aplica.
* **Añadir video del partido**: El usuario puede incluir un enlace de YouTube con el video del partido.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir partidos en estado jugado.
  + **Postcondiciones:** No aplica.
* **Crear partido**: El usuario puede crear un partido.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - Deben existir rivales.
  + **Postcondiciones:** Se crea un nuevo registro en la tabla partidos con estado no jugado.
* **Ver convocatoria próximo partido**: El usuario puede ver un listado de los jugadores convocados para el próximo partido.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener partidos por jugar.
    - Debe haber registros en la tabla Jugadores-Partidos para el próximo partido.
  + **Postcondiciones:** No aplica.
* **Modificar convocatoria próximo partido**: El usuario puede realizar y modificar la convocatoria de jugadores para el próximo partido del equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener partidos por jugar.
    - El usuario debe ser entrenador del equipo.
  + **Postcondiciones:** Se crea un registro en la tabla Jugadores-Partidos por cada jugador seleccionado para el próximo partido.
* **Modificar horarios de entrenamiento del equipo**: El usuario puede modificar los horarios de entrenamiento del equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El usuario debe ser entrenador del equipo.
  + **Postcondiciones:** Se actualizan los registros en la tabla Equipo-Dias\_de\_entrenamiento.
* **Listar jugadores del equipo**: El usuario puede obtener un listado de los jugadores que pertenecen actualmente al equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener jugadores.
  + **Postcondiciones:** No aplica.
* **Eliminar jugador del equipo**: El usuario puede eliminar un jugador del equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - El equipo debe tener jugadores.
    - El usuario debe ser entrenador del equipo.
  + **Postcondiciones:** Se actualiza el registro de la tabla Contratos para el contrato actual del jugador, asignándole fecha final al momento de la acción.
* **Añadir jugadores al equipo**: El usuario puede realizar y modificar la convocatoria de jugadores para el próximo partido del equipo.
  + **Precondiciones:**
    - El usuario debe estar logueado.
    - Deben existir equipos.
    - Deben existir jugadores del mismo género y la misma categoría del equipo que no pertenezcan a ningún equipo
    - El usuario debe ser entrenador del equipo.
  + **Postcondiciones:** Se crea un registro en la tabla Contratos para los jugadores seleccionados, con fecha de comienzo del momento de la acción.

# Diseño

## Base de datos

Entendemos como base de datos al conjunto de datos que pertenecen a un mismo concepto y sistemáticamente almacenado para posteriormente usarlo. En caso de una aplicación, se usan las bases de datos para organizar y operar con los datos guardados en el servidor, en nuestro caso hemos escogido una base de datos MySQL. A la hora de realizar una aplicación es muy importante realizar un buen diseño de la base de datos, para obtener una aplicación de calidad y, fácilmente escalable y mantenible en el tiempo.

### 4.1.1. Diagrama entidad-relación

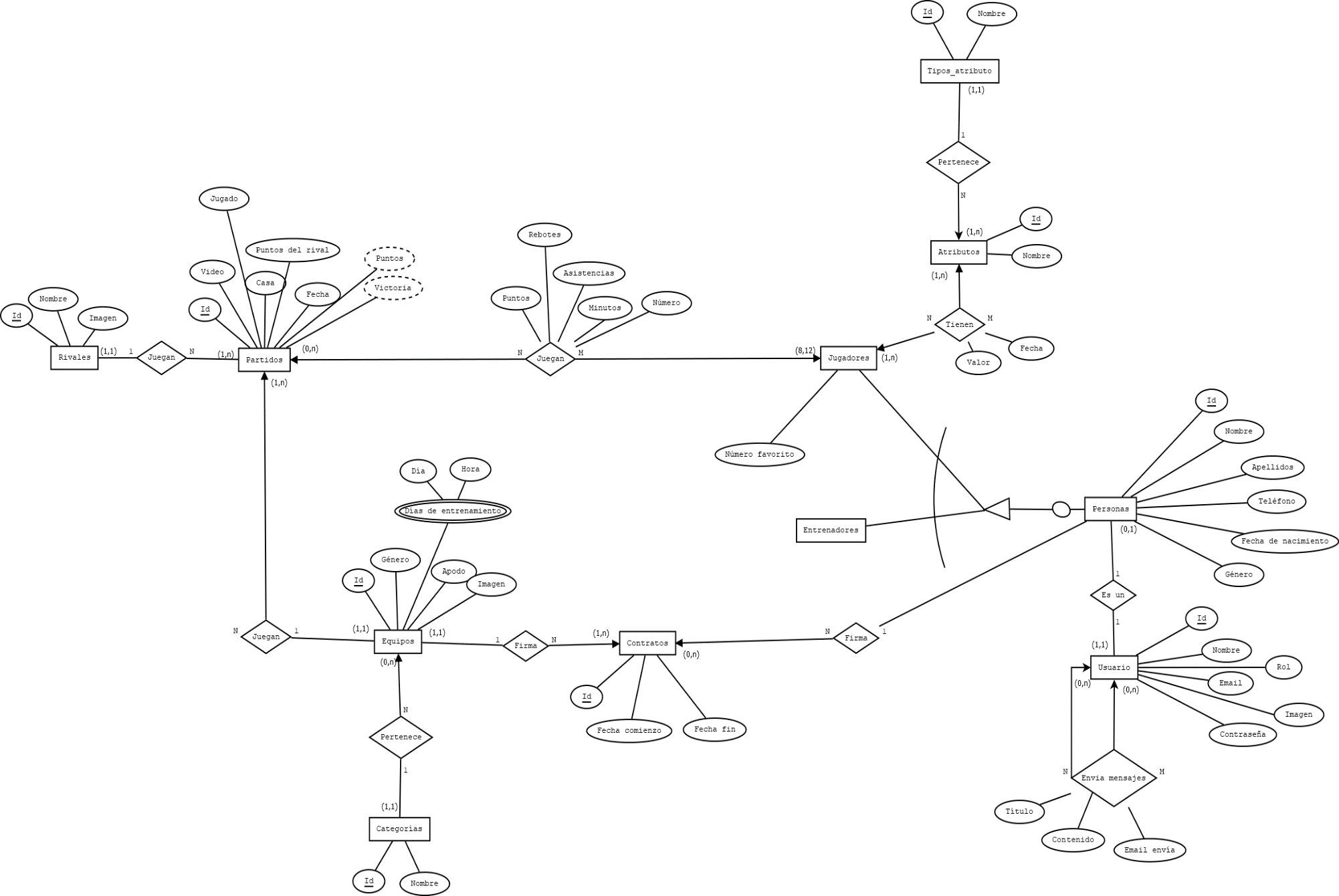


Figura 3.1 – Diagrama entidad relación

### 4.1.2. Diagrama relacional

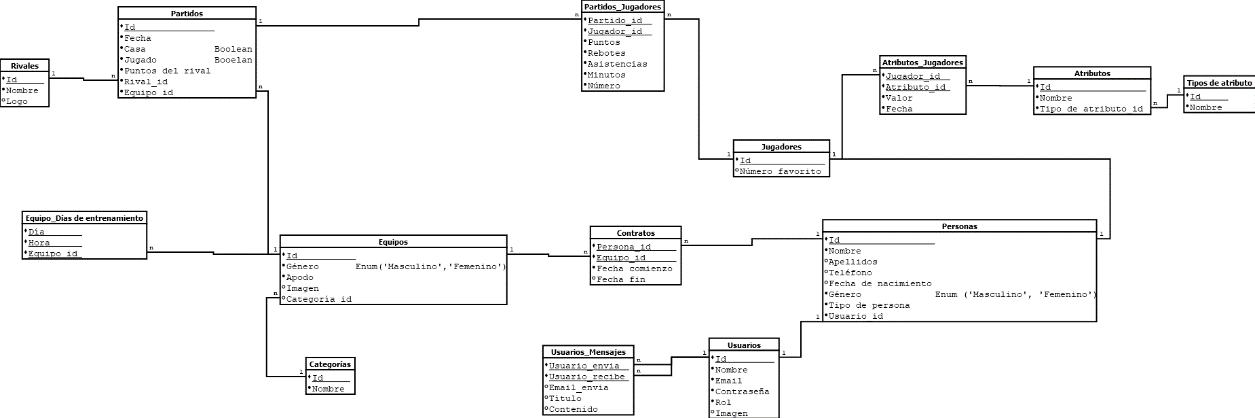


Figura 3.2 – Diagrama relacional

## Diagrama de clases

Un diagrama de clases de diseño recoge tanto los conceptos del dominio del sistema propios del diagrama de clases de análisis como aquellos conceptos que forman parte de la implementación de la propia aplicación del Sistema. A continuación, se muestra el diagrama de clases del proyecto

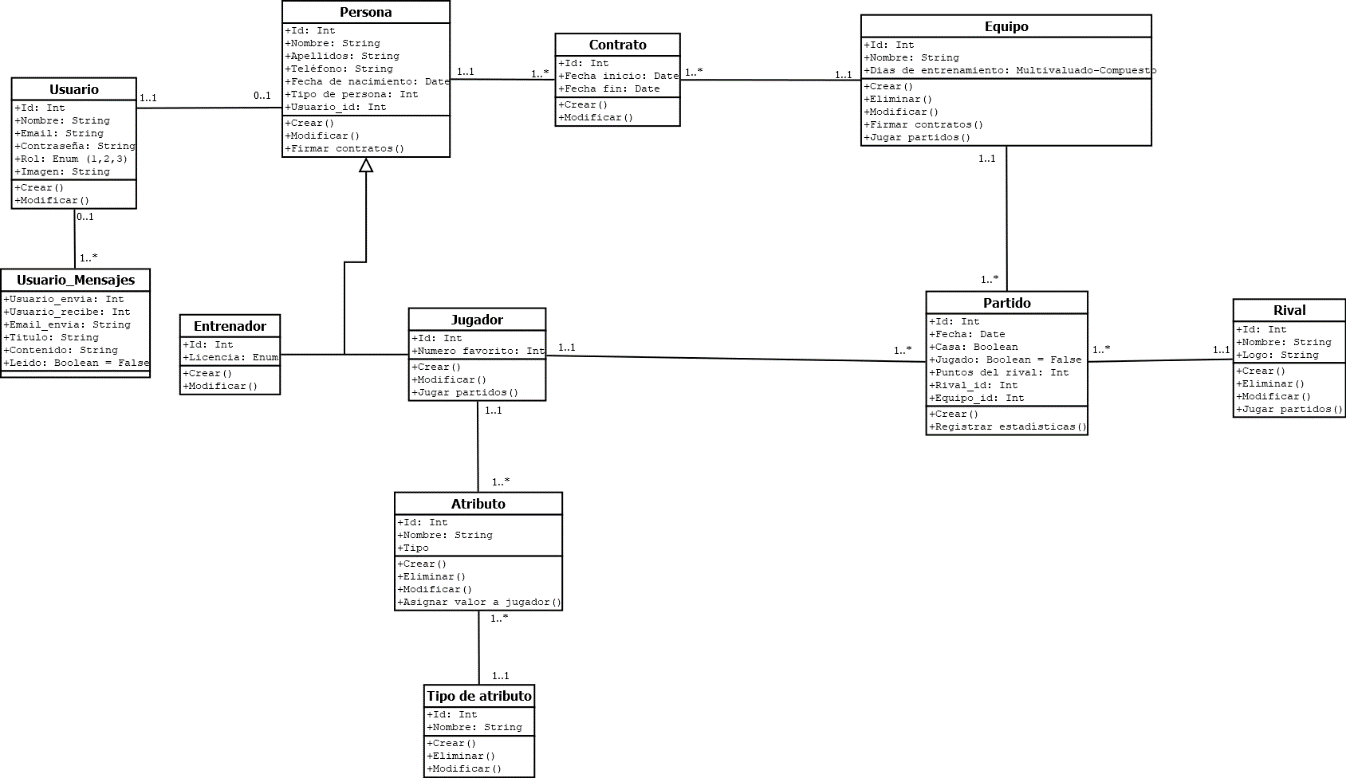


Figura 4.3 – Diagrama de clases

# Tecnologías utilizadas

A continuación, se detallan las diferentes tecnologías que se han usado para la realización del proyecto.

## Lenguajes

Para desarrollar una aplicación web debemos comunicarnos con el ordenador mediante una serie de instrucciones escritas en una sintaxis que la máquina entienda. A continuación, se detallan los lenguajes utilizados en el proyecto.

HTML5

Siglas correspondientes a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto. Es un lenguaje de marcado basado en etiquetas que el navegador interpreta para dar estructura a la página web.

CSS3

Siglas correspondientes a Cascading Style Sheets, es decir, Hojas de Estilo en Cascada. Es un lenguaje de diseño gráfico para dotar de estilo a un documento escrito en un lenguaje de marcado.

PHP

Es un lenguaje de programación diseñado especialmente para el desarrollo de aplicaciones web. El código PHP se ejecuta en el servidor y es este el que envía la página HML al cliente.

JavaScript

Es un lenguaje de programación diseñado especialmente para el desarrollo de aplicaciones web. Se ejecuta en el lado cliente, aunque, gracias a frameworks como NodeJs, puede ejecutarse también en el servidor.

## Base de datos

Para el desarrollo de una aplicación web necesitamos, además, una base de datos que guarde la información de manera organizada para luego poder acceder a ella y utilizarla de manera fácil. A continuación, se detalla el sistema de base de datos utilizado en el proyecto.

MySQL

Es un sistema de gestión de base de datos relacionales de código abierto, siendo considerado como el sistema de gestión de base de datos más popular para el desarrollo web.

## Frameworks y librerías

A la hora de desarrollar una aplicación web podemos apoyarnos en frameworks (entornos de trabajo) y librerías que nos permiten escribir código de una manera más rápida, fácil y mejor estructurada.

Laravel

Es un framework del lenguaje PHP, de código abierto, que sigue el patrón MVC (Modelo, Vista, Controlador). Permite escribir código de una manera sencilla, rápida y con una buena organización de carpetas.

JQuery

Es una biblioteca del lenguaje JavaScript que permite simplificar la manera de interactuar con los elementos de un documento HTML. Además, simplifica el desarrollo de animaciones, el manejo de eventos y las llamadas al servidor mediante AJAX.

Composer

Es un gestor de dependencias para PHP, similar a lo que [npm](https://www.npmjs.com/) es para JavaScript. Es una aplicación PHP que ayuda a administrar las librerías desarrolladas por terceros que vas a incorporar a tu proyecto.

## Control de versiones

Cuando desarrollamos una aplicación web, se considera buena práctica el contar con un sistema de control de versiones que registre los cambios realizados en los archivos de la aplicación a lo largo del tiempo, de modo que nos permite recuperar versiones específicas más adelante. A continuación, se detallan las herramientas utilizadas para el control de versiones.

GIT

Es un software de control de versiones, diseñado por Linus Trovalds y de software libre. Es el más famoso y utilizado en el desarrollo de aplicaciones.

GitHub

Es una plataforma de desarrollo colaborativo, que nos permite alojar el proyecto utilizando el sistema de control de versiones Git. Además, funciona como una red social pensada para desarrolladores.

## Software

Para desarrollar una aplicación necesitamos utilizar una serie de programas que nos permitan ejecutar diferentes tares en nuestro ordenador. A continuación, se detallan los programas utilizados en la realización del proyecto.

Visual Studio Code

Es un editor de código desarrollado por Microsoft y de código abierto. Es el editor más utilizado en la actualidad, permite trabajar con múltiples lenguajes, es muy fácil de personalizar y consta de miles de plugins gratuitos fácilmente instalables.

XAMPP

Es un paquete que contiene el servidor web Apache, un gestor de base de datos MySQL y los intérpretes para los lenguajes de programación PHP y Perl. En el proyecto se ha usado para alojar la aplicación en un servidor local desde el que trabajar.

phpMyAdmin

Es una herramienta que permite manejar la administración de bases de datos MySQL a través de un navegador.

Dia

Es un programa diseñado para el dibujo de diagramas y esquemas. Permite realizar diferentes tipos de esquemas con una interfaz muy fácil de utilizar.

Sourcetree

Es un programa que nos permite trabajar con Git de una manera muy cómoda y sencilla. Permite clonar el repositorio de GitHub y trabajar con el desde una interfaz bastante clara.

# Desarrollo

Para desarrollar la aplicación se ha usado el paradigma de programación MVC (Modelo, vista, controlador), apoyado en el framework de PHP Laravel. El uso de Laravel ha permitido que el desarrollo de la aplicación haya sido ágil, facilitando la creación de los modelos, vistas y controladores necesarios; además de aportar diferentes herramientas como Seeders (semillas) y Factories (factorías). Se ha optado por el uso de Laravel por lo fácil que es crear un proyecto desde cero, únicamente se necesitan una serie de líneas de comandos.

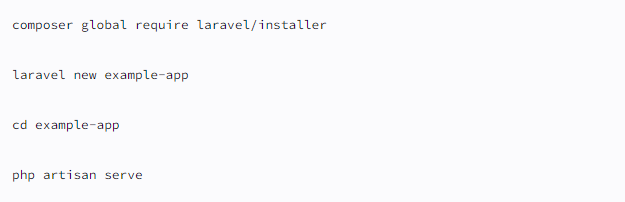


Figura 6.1 – Comandos para crear un proyecto con Laravel y lanzar el servidor

A continuación, se detalla cómo ha sido el desarrollo del proyecto.

## Estructura

Una vez creado el proyecto Laravel obtenemos la siguiente estructura de carpetas:

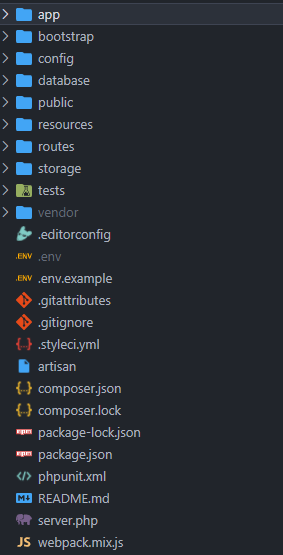


Figura 6.2 – Estructura de carpetas Laravel

A continuación, se detallan los ficheros y directorios más importantes de la aplicación.

### 6.1.1. Directorio App

En este directorio se pueden encontrar diferentes directorios entre las que destacan los siguientes.

Directorio Models

Incluye todos los modelos de la aplicación. (Incluir foto de la carpeta)

Los modelos, además de mapear la base de datos permiten relacionarse entre ellos y obtener los datos de las tablas sin necesidad de escribir código SQL. En las imágenes siguientes se muestran cómo se define una relación en el modelo y cómo se utiliza en el controlador. He querido destacar la relación entre jugadores y personas al ser una relación polimórfica.



Figura 6.3 – Definición de la relación con personas en el modelo jugadores

Con el siguiente código obtenemos dos instancias de dos clases diferentes: la variable $player contiene una instancia de la clase Jugador y la variable $person contiene una instancia de la clase Persona.

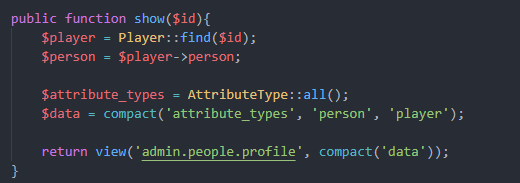


Figura 6.4 –Método show (mostrar) del controlador de jugadores

Además, los modelos nos aportan la posibilidad de crear accessors. Un accessor es un método que permite generar atributos extras en el modelo. Son muy útiles para los atributos derivados y para formatear fechas y horas. El siguiente accessor devuelve la ruta relativa de la imagen preparada para todos los usuarios y para que funcione correctamente en el atributo *‘src’* de una etiqueta HMTL <img>.

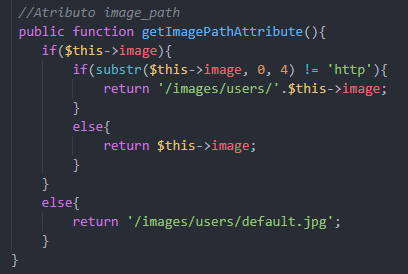


Figura 6.5 – Accesor getImagePathAttribute() del modelo Usuarios

Directorio Traits

Contiene los traits (rasgos) de la aplicación. Un trait es un mecanismo de reutilización de código en lenguajes de herencia simple, como PHP. Es similar a una clase, pero con el objetivo de agrupar funcionalidades específicas de manera coherente. El trait a continuación se usa para formatear una fecha pasando de “21/06/2021” a “Lunes 21 de junio de 2021”.



Figura 6.3 – Trait FormatearDate

Directorio Htpp/Middleware

Contiene los middlewares (lógica de intercambio de información entre aplicaciones) de la aplicación. Un middleware un archivo que filtra las peticiones HTTP en un sistema, es un archivo adicional que va en el medio de la petición y del resultado final en la vista. En el caso de la aplicación hemos creado el AdminMiddleware que gestiona el acceso a las rutas de la aplicación de los diferentes roles de usuario. (Incluir foto del archivo)

Directorio Htpp/Controllers

Contienes los controladores de la aplicación. En el proyecto se han agrupado los controladores en tres carpetas diferentes:

* Directorio raíz: Contiene los controladores necesarios para la aplicación administrativa.
* Publics: Contiene los controladores necesarios para la página web pública de la aplicación.
* Auth: Contiene los controladores del paquete de autenticación de Laravel. En el proyecto solo se ha usado el LoginController para gestionar el inicio de sesión de los usuarios, aunque no se han eliminado el resto de controladores por si en un futuro se desea implementar alguna otra funcionalidad del paquete.

(Incluir foto de la carpeta controllers)

### 6.1.2. Directorio database

En este directorio se pueden encontrar diferentes directorios entre las que destacan los siguientes.

Directorio migrations

Contiene las migraciones de la aplicación. Las migraciones son las encargadas de crear las tablas y campos de la base de datos. En la imagen se muestra la migración encargada de crear la tabla personas, he querido destacarla porque es la tabla padre de una jerarquía y para implementarla hay que usar una relación polimórfica.

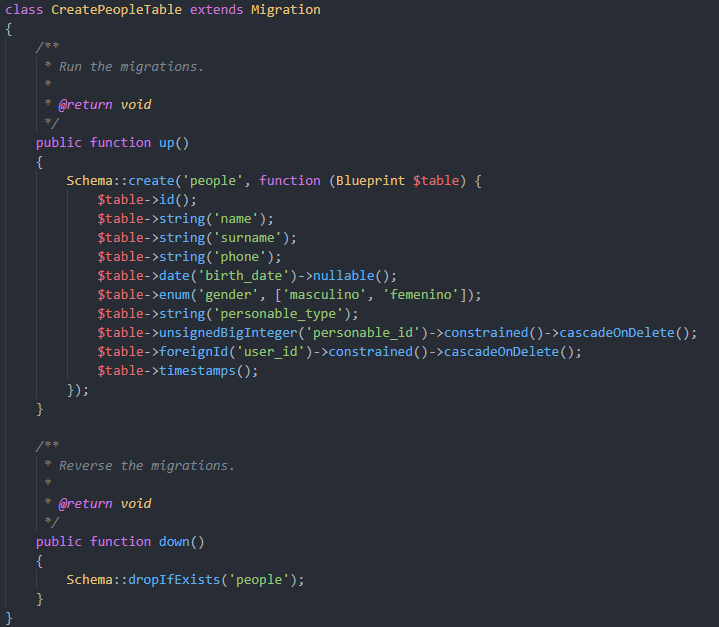


Figura 6.4 – Migración tabla personas

Directorio seeders

Contiene los seeders (semillas) de la aplicación. Los seeders son componentes que nos permiten rellenar las tablas de datos. Al comienzo del desarrollo se utilizaron para llenar las datas con datos ficticios y poder hacer pruebas. Al final se han seguido usando algunos de ellos, principalmente el UserSeeder, encargado de crear usuarios y de enlazarlos con las tablas personas, jugadores y entrenadores, rellenando todas las tablas con datos ficticios, apoyándose en la librería faker.

### 6.1.3. Directorio public

Contiene todos los assets (activos digitales) necesarios para usar la aplicación. Un asset es una representación de cualquier item utilizado en el proyecto. El directorio incluye los ficheros JavaScript, CSS y las imágenes. Está dividido en dos directorios:

* Directorio raíz: Contiene los assets necesarios para la aplicación de administrativa.
* Public-template: Contiene los assets necesarios para la página web pública de la aplicación.

### 6.1.4. Directorio resources

Contiene varios directorios entre los que destaca el directorio views.

Directorio views

El directorio views (vistas) contiene todas las vistas de la aplicación. Las vistas están escritas en sintaxis Blade, el sistema de plantillas predeterminado de Laravel. Blade nos permite integrar código PHP en el HTML de una manera sencilla y clara. La imagen siguiente nos muestra una tabla recorriendo la variable $data[‘players’], incluyendo varios condicionales.

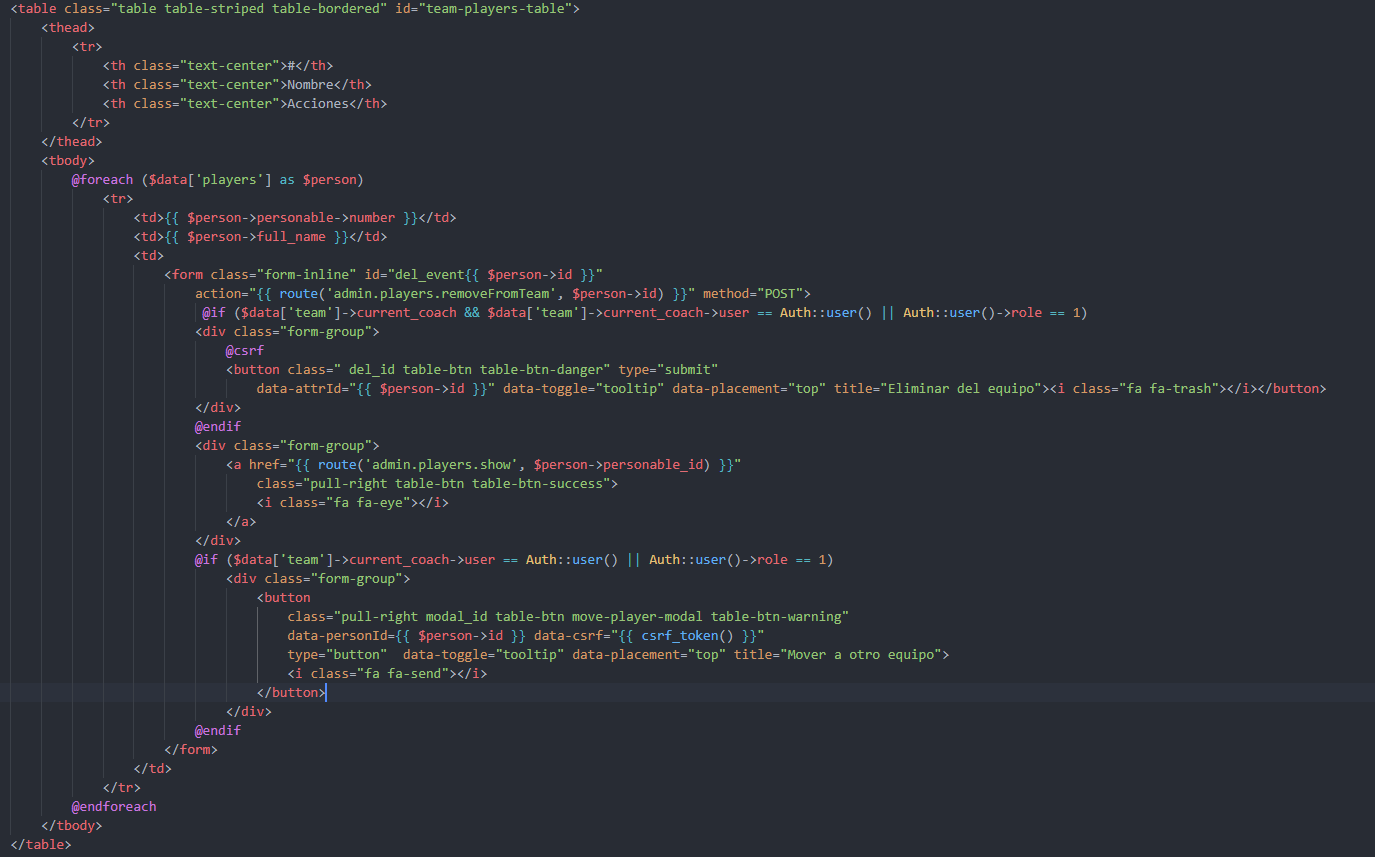


Figura 6.6 – Parte de la vista admin.teams.players.blade.php

Está dividido en dos directorios:

* Admin: Contiene las vistas de la aplicación administrativa.
* Directorio raíz: Contiene las vistas de la página web pública de la aplicación.

### 6.1.5. Directorio routes

Incluye diferentes ficheros entre los que destaca el fichero web.php. Este fichero es el encargado de gestionar las rutas de la aplicación de una manera intuitiva y fácil de utilizar.

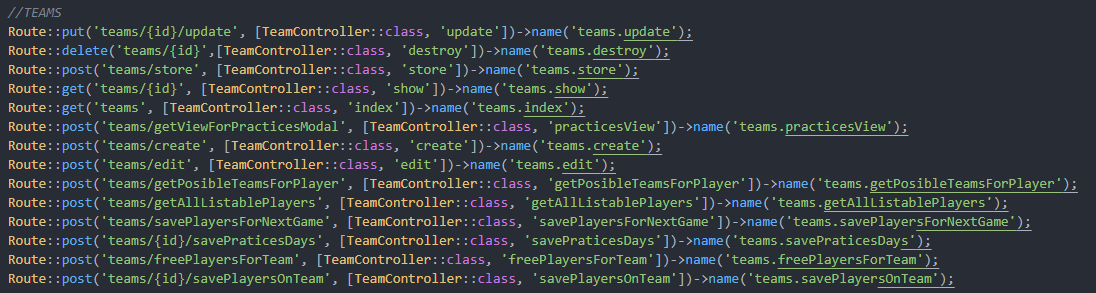


Figura 6.5 – Rutas del controlador de equipos