MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Multiplataforma

Título del proyecto: **CHAMPIONS SIMULATOR**

**Autor:** Martín Mancini Pablo José Tolosa Piquín

**Tutor:** Enrique Finistrosa

Fecha de entrega: 30/05/2023

**Año:** 2023

**Documentos del proyecto:**

[**https://drive.google.com/drive/folders/17ZpqnGqDPqM6z9ZtDYoRyn\_4c7V68A3e**](https://drive.google.com/drive/folders/17ZpqnGqDPqM6z9ZtDYoRyn_4c7V68A3e)

2022/2023



Índice de contenidos

[1. Introducción 3](#_Toc136269904)

[1.1. Motivación 3](#_Toc136269905)

[1.2. Abstract 3](#_Toc136269906)

[1.3. Objetivos propuestos 3](#_Toc136269907)

[2. Metodología usada 4](#_Toc136269908)

[3. Tecnologías y herramientas utilizadas en el proyecto 4](#_Toc136269909)

[3.1. Lenguajes de programación utilizados 4](#_Toc136269910)

[3.2. Herramientas utilizadas. 4](#_Toc136269911)

[3.3 Sistema Operativo Utilizado 5](#_Toc136269912)

[4. Estimación de recursos y planificación 6](#_Toc136269913)

[5. Análisis del proyecto 7](#_Toc136269914)

[5.1. Requisitos funcionales 7](#_Toc136269915)

[5.2. Requisitos no funcionales 7](#_Toc136269916)

[5.3. Diagramas de casos de uso 8](#_Toc136269917)

[5.4. Casos de uso 10](#_Toc136269918)

[6. Diseño del proyecto 16](#_Toc136269919)

[6.1. Implementación 17](#_Toc136269920)

[6.2. Diagrama de clases 28](#_Toc136269921)

[1. Despliegue y pruebas 34](#_Toc136269922)

[2. Conclusiones 38](#_Toc136269923)

[3. Vías futuras 39](#_Toc136269924)

[4. Glosario 40](#_Toc136269925)

[5. Bibliografía/Webgrafía 41](#_Toc136269926)

[5.1. Información 41](#_Toc136269927)

[5.2. Código 42](#_Toc136269928)

[6. Anexos 43](#_Toc136269929)

[12.1 Imágenes diseño aplicación 43](#_Toc136269930)

# Introducción

## Motivación

Para la realización del siguiente proyecto, se tuvo la idea de realizar un videojuego de simulación de futbol mediante el uso de las herramientas aprendidas a lo largo del curso.

La idea principal es realizar un videojuego de futbol en el cual tu eres el entrenador del equipo y la tarea que tienes es antes de empezar el partido poner la alineación adecuada para tener más oportunidades de ganar el partido.

Se tuvo que pensar muy bien cómo realizar esta tarea debido a que muchas cosas que queríamos implementar se tuvieron que estudiar por nuestra cuenta ya que, con los conocimientos curriculares, no nos era suficiente.

## Abstract

For the realization of the following project, we had the idea of making a soccer simulation video game by using the tools learned throughout the course.

The main idea is to make a soccer video game in which you are the coach of the team and the task you have is before starting the game to put the right line-up to have more chances to win the game. We had to think very well how to do this task because many things we wanted to implement had to be studied on our own since, with the curricular knowledge, it was not enough for us.

## Objetivos propuestos

El objetivo es que nuestra aplicación sea capaz de simular un partido de futbol entre dos equipos que, mediante un sistema de estrellas, tengan una diferencia de probabilidad a la hora de ganar. A su vez, generar una especie de liga donde el resto de equipos compitan entre sí de forma automática paralelamente a los partidos que realizamos con nuestro equipo. Queremos que este sistema de estrellas también influya en eventos que surjan a lo largo del partido como, por ejemplo, si tiene un menor número de estrellas, que sea más propenso a cometer una falta o tener una menor probabilidad de gol, pero que no imposibilite a un equipo con menos estrellas poder ganar un partido contra un equipo con mejores estadísticas.

# Metodología usada

Hemos utilizado la metodología Scrum debido a que como teníamos horarios diferentes utilizamos el programa de dailys y planning (una daily cada día y un planning cada semana) para ver que tareas íbamos a realizar esa semana.

Para esta metodología de trabajo utilizamos la herramienta de Trello. Esta herramienta nos permite tener un tablero donde ver las diferentes tareas que tenemos que llevar a cabo y ponerlas en las diferentes fases del proceso de desarrollo del proyecto para poder diferenciar que tareas estamos realizando cada uno y cuáles de estas ya estaban terminadas. Todo esto unido al uso de GitHub para poder tener un control de versiones e ir desarrollando la aplicación de forma paralela.

# Tecnologías y herramientas utilizadas en el proyecto

## Lenguajes de programación utilizados

* **Java:** hemos utilizado java para elaborar toda la lógica de la aplicación y la conexión a la base de datos
* **JavaFX:** de la mano de java, hemos empleado JavaFX para elaborar la interfaz gráfica de toda la aplicación.
* **CSS:** el lenguaje “Hojas de estilo en cascada” lo hemos empleado para realizar el diseño de diversos elementos de la aplicación, así como para elaborar pequeñas animaciones para los botones.
* **SQL:** hemos realizado la base de datos de nuestro programa con este lenguaje ya que es el lenguaje de programación en la parte de bases de datos que mejor nos sabíamos desarrollar.

## Herramientas utilizadas.

* **IDE NetBeans IDE 8.2:** elegimos este IDE porque es el que hemos utilizado todo el año en esta asignatura y en el que nos sentíamos más cómodos para manejar JavaFX y programar en lenguaje Java.
* **SceneBuilder**: utilizamos esta herramienta para la elaboración de la interfaz y la estética de la aplicación.
* **XAMPP**: se utiliza esta herramienta para tener la base de datos operativa de forma local que pueda utilizar nuestra aplicación.
* **Git, GitHub y GitHub Desktop:** utilizamos esta herramienta para tener un repositorio en la nube que nos permite tener una dinámica de trabajo más eficiente en la que podemos realizar cambios en el código de la aplicación y compartirlo con mayor facilidad. Os dejamos el enlace del proyecto: https://github.com/Matmart13/ChampionSimulator
* **VisualStudio Code:** Este IDE lo hemos utilizado para la elaboración del CSS que utilizan las ventanas de nuestra aplicación.
* **Photoshop CS6:** Para la elaboración del logo de la aplicación, así como de los fondos de pantalla utilizados en las diferentes ventanas.
* **Sony Vegas 16:** Para la edición y postproducción de los gifs utilizados en la aplicación.
* **Venngage:** Para la elaboración de los diagramas de Gantt.
* **Microsoft Word:** Para la elaboración de esta memoria.

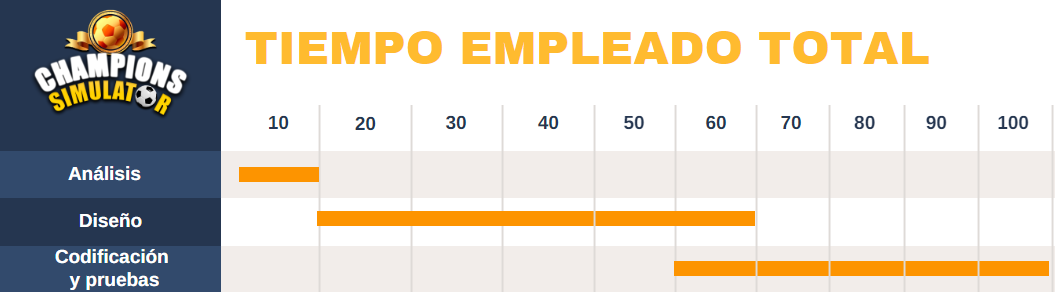
# 3.3 Sistema Operativo Utilizado

Se decidió utilizar el sistema operativo Windows debido a que es el sistema operativo con el que más familiarizados estamos a la hora de trabajar. A su vez, hemos utilizado tanto la versión 10 como la 11 ya que eran las que teníamos instaladas en nuestros ordenadores.

# Estimación de recursos y planificación

El tiempo que se estimó necesitar para realizar la aplicación es de alrededor de 100 horas, este tiempo lo dividiremos en las diferentes fases de desarrollo que nos hemos propuesto para llevar a cabo el proyecto (análisis y planificación, diseño de la aplicación, codificación y pruebas)

De manera inicial estimamos que necesitaremos:

* **Análisis y planificación:** Se dedicará alrededor de unas 10 horas para el análisis y planificación iniciales.
* **Diseño de la aplicación:** Se dedicará alrededor de unas 20 horas para elaborar el diseño de la aplicación.
* **Codificación y pruebas:** Se dedicará alredor de unas 70 horas en elaborar toda la programación de la aplicación, así como de la realización de las pruebas unitarias.

**Diagrama de Gantt tiempo empleado total**

**Diagrama de Gantt tiempo estimado**

# Análisis del proyecto

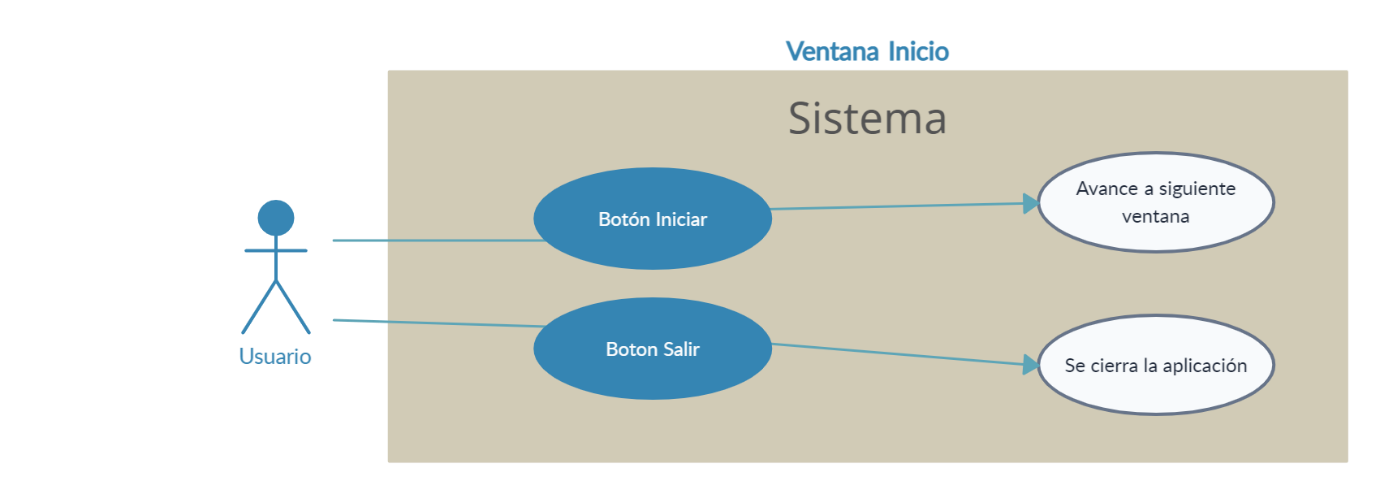
## Requisitos funcionales

* El usuario debe poder utilizar los botones de la ventana “Selección” para acceder a la partida o para salir de la aplicación.
* En la pantalla “Selección” de equipo el usuario debe ser capaz de seleccionar entre todos los equipos que hay disponibles.
* En la pantalla de “Temporada” debe mostrarse el logo y la alineación seleccionados en la pantalla anterior.
* En la pantalla de “Temporada” debe mostrarse el estado de la temporada actual.
* El usuario debe poder utilizar los botones de la ventana de temporada para comenzar el partido o para salir de la aplicación
* Durante estas 3 primeras ventanas sonara una canción de fondo.
* En la ventana “Partido” saldrá el logo del equipo que el usuario ha seleccionado y el de su rival.
* En la ventana “Partido” aparecerá un contador funcional que marcará el tiempo de cada parte.
* En la ventana “Partido” aparecerá un marcador que mostrará los goles de cada equipo.
* En la ventana “Partido”, en el JTextField aparecerá todo lo que está sucediendo en el partido.
* Durante el partido sonara una canción de fondo diferente a la del menu.
* Al finalizar el partido aparecerá una ventana con el ganador.
* La ventana “Ganador” tendrá un botón que te redireccionará a la ventana temporada para poder seguir jugando.
* Las pruebas unitarias deben mostrar que todos los métodos de la aplicación son correctos.

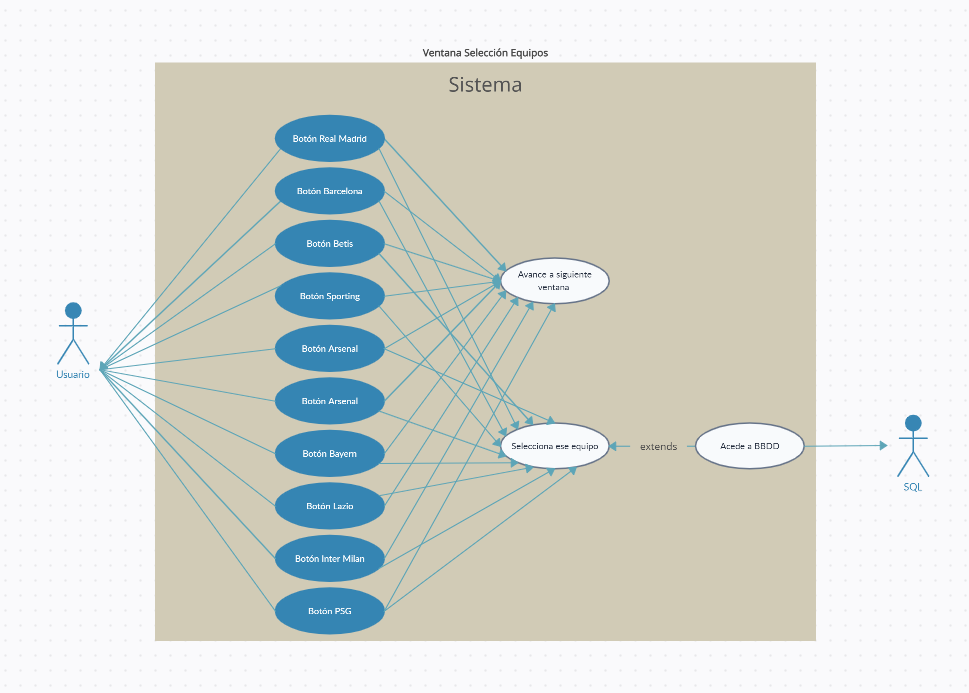
## Requisitos no funcionales

* Toda la aplicación estará escrita en lenguaje Java a excepción del apartado grafico que será estilizado con CSS.
* Los botones de toda la aplicación deben contar con animaciones al pasar el cursor por encima y al ser pulsados.
* Todos los botones de los equipos deben contar con animaciones al pasar el botón por encima.
* El código deber ser lo más limpio y conciso posible.
* Se debe utilizar el código fuente (git) para registrar todos los cambios y progresos del desarrollo de la aplicación una vez que estos han sido probados.

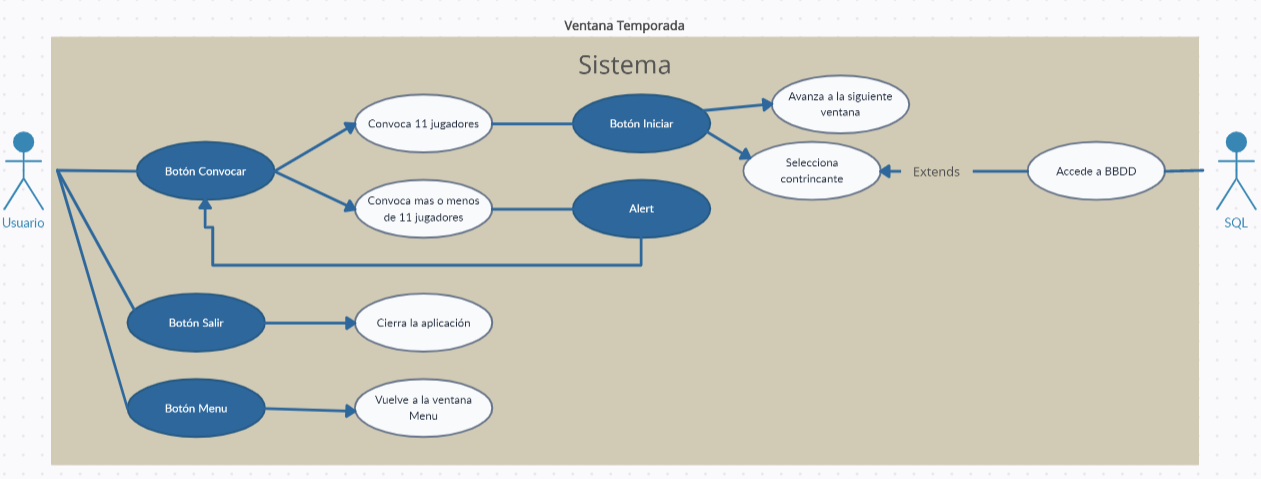
## Diagramas de casos de uso

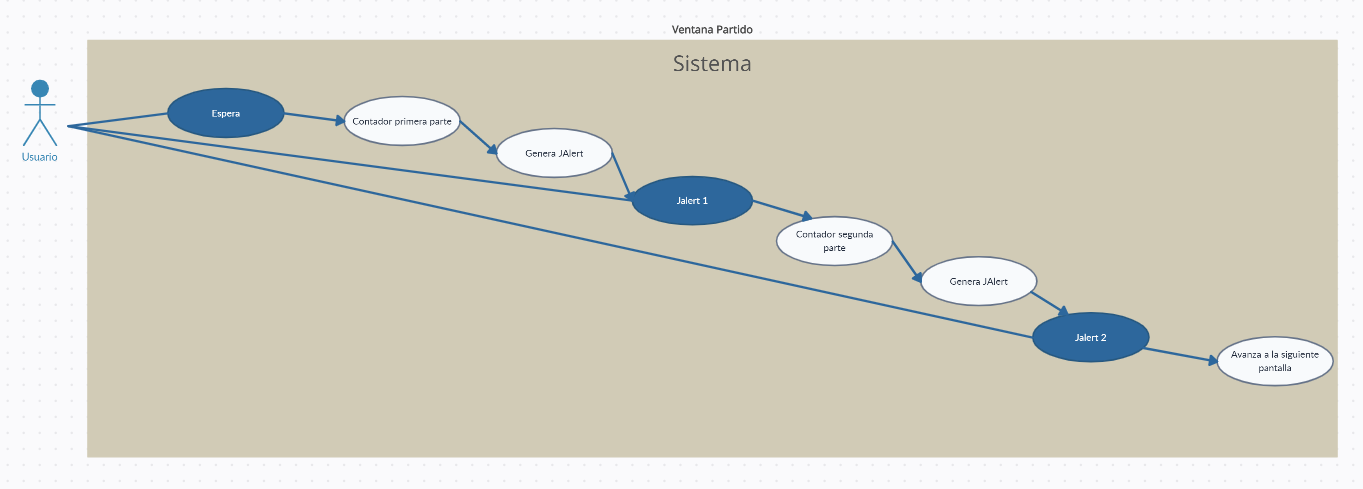
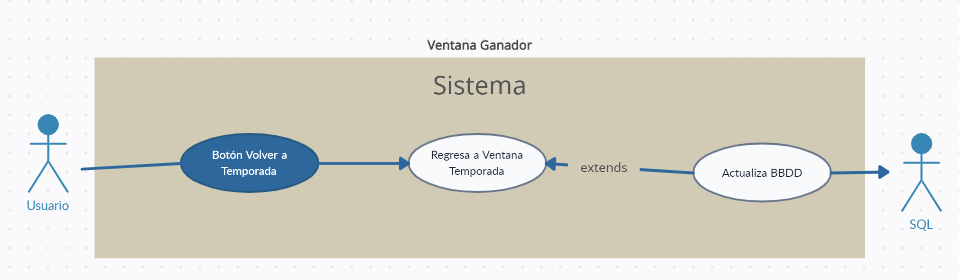


**Diagrama casos de uso Ventana Inicial**



**Diagrama casos de uso Ventana Selección**





**Diagrama de casos de uso Ventana Ganador**

**Diagrama de casos de uso Ventana Partido**

**Diagrama casos de uso Ventana Temporada**

## Casos de uso

|  |  |  |
| --- | --- | --- |
| CASO 1 | El usuario pulsa el botón de Inicio | |
| Dependencias | Ninguna | |
| Precondición | Ninguna | |
| Descripción | El sistema debería de enviar al usuario a la ventana de “Selección de equipo | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario pulsa el botón |
| 2 | El sistema envía al usuario a la ventana “Selección de equipo” correctamente |
| Postcondición | El usuario ejecuta alguno de los casos de uso dependientes de este. | |
| Excepciones | La ventana Selección lanza la excepción por un error de invocación | |

|  |  |  |
| --- | --- | --- |
| CASO 2 | El usuario pulsa el botón de salir | |
| Dependencias | Ninguna | |
| Precondición | Ninguna | |
| Descripción | El sistema debería de cerrar la ejecución de la aplicación | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario pulsa el botón |
| 2 | El sistema cierra la ejecución de la aplicación |
| Postcondición | La aplicación finaliza todo procesos | |
| Excepciones | Ninguna | |

|  |  |  |
| --- | --- | --- |
| CASO 3 | El usuario pulsa el botón de pasar canción | |
| Dependencias | Dependiendo de si se encuentra en la ventana “Selección” o en la de “Temporada” los casos necesarios para llegar a ellas | |
| Precondición | Ninguna | |
| Descripción | El sistema debería de cambiar la canción que suena de fondo | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario pulsa el botón |
| 2 | El sistema pasa a la canción siguiente |
| Postcondición | La canción que suena de fondo cambia | |
| Excepciones | Ninguna | |

|  |  |  |
| --- | --- | --- |
| CASO 4 | El usuario pulsa el botón de mute | |
| Dependencias | Dependiendo de si se encuentra en la ventana “Selección” o en la de “Temporada” los casos necesarios para llegar a ellas | |
| Precondición | Ninguna | |
| Descripción | El sistema debería de para la canción que suena de fondo | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario pulsa el botón |
| 2 | El sistema pasa a la canción siguiente |
| Postcondición | La canción que suena de fondo se silencia. | |
| Excepciones | Ninguna | |

|  |  |  |
| --- | --- | --- |
| CASO 5 | El usuario selecciona algún equipo | |
| Dependencias | Caso 1 | |
| Precondición | El usuario ha decidido iniciar el juego en la pantalla de inicio | |
| Descripción | El sistema redirigirá al usuario a la pantalla temporada | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario pulsa el botón |
| 2 | El sistema envía al usuario a la ventana “Temporada” |
| Postcondición | La ventana “Temporada” tendrá toda la información del equipo seleccionado. | |
| Excepciones | Excepción por perdida de conexión a la BBDD y error de invocación | |

|  |  |  |
| --- | --- | --- |
| CASO 5 | El usuario selecciona algún equipo | |
| Dependencias | Caso 1 | |
| Precondición | El usuario ha decidido iniciar el juego en la pantalla de inicio | |
| Descripción | El sistema redirigirá al usuario a la pantalla temporada | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario pulsa el botón |
| 2 | El sistema envía al usuario a la ventana “Temporada” |
| Postcondición | La ventana “Temporada” tendrá toda la información del equipo seleccionado. | |
| Excepciones | Excepción por perdida de conexión a la BBDD y error de invocación | |

|  |  |  |
| --- | --- | --- |
| CASO 6 | El usuario pulsa el botón convocar | |
| Dependencias | Caso 1 y Caso 5 | |
| Precondición | El usuario ha decidido seleccionar un equipo para jugar | |
| Descripción | El sistema actualizara el estado del jugador seleccionado de “Suplente” a “Titular” | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario pulsa el botón |
| 2 | El sistema se conecta a la BBDD |
| 3 | El sistema actualiza el estado del jugador |
| Postcondición | La ventana “Temporada” mostrara el estado actualizado del jugador seleccionado. | |
| Excepciones | Excepción por perdida de conexión a la BBDD. | |

|  |  |  |
| --- | --- | --- |
| CASO 7 | El usuario pulsa el botón Menú | |
| Dependencias | Caso 1 y Caso 5 | |
| Precondición | El usuario ha decidido seleccionar un equipo para jugar | |
| Descripción | El sistema te redirigirá a la ventana menú permitiendo volver a seleccionar equipo | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario pulsa el botón |
| 2 | El sistema vuelve a la ventana “Menú” |
| Postcondición | El usuario volverá al menú y este podrá volver a elegir el equipo con el que jugar. | |
| Excepciones | Ninguna | |

|  |  |  |
| --- | --- | --- |
| CASO 8 | El usuario convoca más o menos de 11 jugadores | |
| Dependencias | Caso 1, Caso 5 y Caso 6 | |
| Precondición | El usuario ha decidido seleccionar un equipo para jugar y ha convocado un numero distinto a 11 jugadores. | |
| Descripción | El sistema mostrara un alert al intentar iniciar el partido | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario convoca más o menos de 11 jugadores |
| 2 | El usuario pulsa el botón de iniciar partido |
| 3 | El sistema muestra una alerta advirtiendo del error |
| Postcondición | No podrá avanzar de la ventana temporada a la de partido | |
| Excepciones |  | |

|  |  |  |
| --- | --- | --- |
| CASO 9 | El usuario decide iniciar el partido con el equipo seleccionado | |
| Dependencias | Caso 1 y Caso 5 y Caso 6 | |
| Precondición | El usuario ha decidido iniciar el juego en la pantalla de inicio y ha escogido un equipo | |
| Descripción | El sistema redirigirá al usuario a la ventana “Partido” | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario pulsa el botón |
| 2 | El sistema envía al usuario a la ventana “Partido” |
| Postcondición | El partido comienza su ejecución y con el equipo que el usuario ha seleccionado en un inicio. | |
| Excepciones | Excepción por perdida de conexión a la BBDD y error de invocación | |

|  |  |  |
| --- | --- | --- |
| CASO 10 | El usuario decide continuar el partido cuando acaba la primera parte | |
| Dependencias | Caso 1, Caso 5, Caso 6 y Caso 8 | |
| Precondición | El usuario ha decidido continuar con el partido cuando el JAletr ha saltado anunciando el fin de la primera parte. | |
| Descripción | El sistema reanudara la ejecución del partido con el inicio de la segunda parte | |
| Secuencia normal | Paso | Acción |
| 1 | El sistema lanza el JAlert |
| 2 | El usuario le da al botón continuar |
| 3 | El sistema comienza la segunda parte del partido |
| Postcondición | El partido finaliza al acabar la segunda parte y el sistema envia un segundo JAlert para anunciar del fin de la misma | |

|  |  |  |
| --- | --- | --- |
| CASO 11 | El usuario pulsa el botón volver a temporada de la ventana “Ganador” | |
| Dependencias | Caso 1, Caso 5, Caso 6, Caso 8 y Caso 9 | |
| Precondición | El usuario ha decidido continuar con el partido cuando el JAlert ha saltado anunciando el fin de la primera parte. | |
| Descripción | El sistema redirige al usuario a la ventana temporada y en esta se ven los resultados de los partidos actualizados en las tablas de la misma | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario pulsa el botón volver a temporada |
| 2 | El sistema actualiza los datos en la BBDD |
| 3 | El sistema envía al usuario a la ventana “Temporada” |
| Postcondición | La tabla de temporada de la ventana “Temporada” tiene que haber sido actualizada en base a los resultados de los enfrentamientos ocurridos en esa fase | |
| Excepciones | Excepción por perdida de conexión a la BBDD y error de invocación | |

# Diseño del proyecto

Nuestra aplicación comienza con una pantalla inicial compuesta de un fondo animado, el logo en la parte superior y los 3 botones que nos permitirán, iniciar partida, cargar una jugada anteriormente y por último salir de la aplicación si se desea (ver en [Anexo](#_12.1_Imágenes_diseño)). Estos botones cuentan con animaciones a la hora de pasar el ratón por encima y al pulsar sobre ellos.

Una vez iniciada la partida pasamos a la segunda pantalla, la selección de equipos. Esta pantalla está compuesta por diez botones, cada uno para su referente equipo con un logo que los identifica. Estos logos son animados para los equipos de mayor numero de estrellas. Todos los botones tienen una animación al pasar el ratón por encima.

A la derecha de los botones encontramos el número de estrellas que tiene el equipo en particular. De nuevo, cuando estos equipos son de 5 estrellas estos se diferencias por contar una pequeña animación en las estrellas. También tienen un JLabel que indicará el nombre del equipo y este será dorado para los equipos de 5 estrellas (ver en [Anexo](#_12.1_Imágenes_diseño)).

Una vez seleccionado el equipo, pasamos a la ventana temporada. En esta ventana tenemos por un lado en la esquina superior izquierda el logo del equipo que hemos seleccionado, debajo su alineación, a la derecha un resumen de la temporada que estamos jugando y por último en la parte inferior derecha los botones de iniciar partido (que comenzara el siguiente partido de la temporada) y el botón salir (este cerrara la aplicación). La pantalla cuenta con un fondo personalizado con el logo de la aplicación (ver en [Anexo](#_12.1_Imágenes_diseño)).

Por último, estas 3 pantallas cuentan con una canción de fondo para darle un poco más de vida a la aplicación. Esta canción se detendrá al cambiar a la pantalla del partido y por ende, comenzará la canción del partido.

Pasamos ahora con la ventana partido. Esta ventana comienza en la parte superior con los logos de los equipos que se van a enfrentar, uno en cada esquina. En el medio un diseño personal para simbolizar el “Versus”. Debajo del mismo encontramos el temporizador del partido y el marcador del mismo.

Por último, tenemos el JTextArea donde se mostrarán todos los acontecimientos que ocurrirán en el partido. De fondo esta ventana cuenta con un diseño basado en un color azul degradado. La ventana partido cuenta con su propia canción para darle un toque de acción, así como sonidos de silbatos para el comienzo de cada parte del partido. (ver en [Anexo](#_12.1_Imágenes_diseño)).

Por último, la ventana final. Esta está compuesta por el logo del ganador del partido, un gif animado debajo y un botón para volver a la ventana temporada donde podremos seguir jugando (ver en [Anexo](#_12.1_Imágenes_diseño)). Esta ventana tendrá un sonido que se escuchará nada más aparecer la ventana.

## Implementación

Lo primero de todo que se pensó en hacer fue crear la plantilla de la estructura del proyecto con los diferentes paquetes de java que lo compondrán, los cuales mencionaremos a continuación.



**Estructura proyecto Champions Simulator**

Comenzando por el paquete “Auxiliares”, este almacena los archivos encargados de realizar la conexión con la BBDD, los CSS que estilizan la aplicación y las clases encargadas de gestionar los sonidos que se reproducirán a lo largo de la ejecución.

El segundo paquete “ChampionsSimulator”, contiene la clase que inicializa la aplicación.

El tercer paquete “Controladores”, contiene los controladores con toda la programación de las vistas que componen la aplicación.

El cuarto paquete lo componen las imágenes, todas las imágenes que hemos utilizado en todas las vistas de la aplicación se encuentran en este paquete.

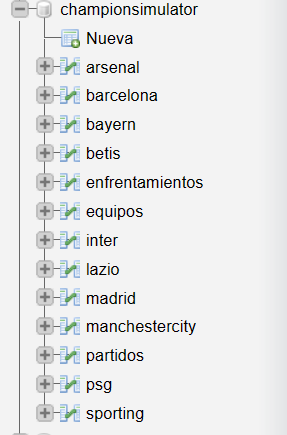
El quinto paquete contiene los modelos de los objetos que utilizaremos para almacenar la información a lo largo de la ejecución de la aplicación.

El paquete “Música” almacena todos los sonidos y canciones que sonaran a lo largo de la ejecución del programa.

El paquete “Útiles” contiene la programación de los Hilos que se ejecutaran durante el partido.

Por último, el paquete “Vistas” contiene todas las vistas de las ventanas que conforman la aplicación.

A continuación, lo que se realizo fue la creación de la base de datos la cual se compone de las diferentes tablas:



**Estructura BBDD Champions Simulator**

Por un lado, tenemos la tabla “Equipos” que almacena todos los equipos que tiene nuestra aplicación. Después tenemos la tabla “Enfrentamientos” que almacena todos los posibles partidos que pueden surgir a lo largo de la ejecución del programa.

Por último, respecto al resto de tablas, estas tienen la función de almacenar la información de los jugadores de cada equipo.

Mas tarde con la creación de los Junit, nos dimos cuenta de que al realizar los Test unitarios necesarios para la comprobación del buen funcionamiento de todos y cada uno de los métodos utilizados en las diferentes clases de java, se estaban alterando los campos de la base de datos principal y este acontecimiento no podría darse ya que es una nefasta práctica. La solución que se dio a este problema fue la creación de otra base de datos específicamente para los Test Unitarios de java que necesiten alterar la base de datos. Esta base de datos es exactamente igual a la anterior.

Se siguió con añadir las clases de java que son las siguientes:

En el paquete de Auxiliares se añadieron las clases:

* **ElegirCSS:** Este archivo CSS sirve para darle estilo a la ventana donde el usuario elige el equipo.
* **GanadorCSS:** Este archivo CSS sirve para darle estilo a la ventana donde se muestra el ganador del partido.
* **InicioCss**: Este archivo CSS sirve para darle estilo a la pantalla de inicio de la aplicación.
* **TemporadaCSS:** Este archivo CSS sirve para darle estilo a la ventana Temporada donde se encontrará la alineación y la clasificación de la liga.
* **Sonido:** Esta clase sirve para desarrollar toda la programación de una instancia de sonido.
* **SonidoManager:** Esta clase sirve para el manejo de los posibles diferentes sonidos que tendrá nuestra aplicación.
* **Conexiones:** Sirve para las conexiones entre la aplicación con las diferentes bases de datos.

El paquete de “ChampionSimulator” solo se compone de una clase Java:

* **ChampionSimulator.java:** Esta clase sirve para ejecutar la aplicación.

En el paquete de “Controladores” se añadieron las siguientes clases para el control de las vistas:

* **FXML\_VentanaElegirController:** Esta clase contiene la lógica de la ventana de la selección de equipo.
* **FXML\_VentanaGanadorController:** Este archivo se compone de toda la lógica de la ventana Ganador.
* **FXML\_VentanaInicioController:** Esta clase contiene la lógica de los botones de la ventana “Inicio”.
* **FXML\_VentanaPartidoController:** Esta clase contiene parte de la lógica de la ventana partido, esta se verá complementada por las clases contenidas en el paquete “Útiles”
* **FXML\_VistaTemporadaController:** Esta clase contiene la lógica de los botones y las tablas de la ventana Temporada.

En el paquete “Modelo” se añadieron las siguientes clases:

* **Equipo:** La clase para generar los objetos con la información del equipo.
* **Jugador:** La clase para generar los objetos con la información de un jugador.
* **Partidos:** La clase para generar los objetos con la información de cada partido.

En el paquete “Útiles” se añadieron las siguientes clases:

* **HiloTiempo:** Esta clase contiene la lógica empleada para hacer funcionar el contador de tiempo del partido
* **HiloPartido:** Esta clase contiene la lógica empleada para hacer funcionar la ejecución del partido.

Por último, como se ha mencionado antes, los paquetes “Música” e “Imágenes” almacenan todo el contenido multimedia utilizado en la programación de la aplicación. El paquete “Vistas” todas y cada una de las vistas de las ventanas que componen el programa.

Ya mostradas las diferentes clases vamos a continuar por explicar que métodos existen en estas clases y cuál es su funcionalidad:

**Sonido:**

* **public Sonido (String \_url) :** Este método es el método constructor de esta clase que necesita una url por parámetro.
* **public void ReproducirSonido() :** Este método es sirve para reproducir el sonido.
* **public void PararSonido() :** Este método sirve para parar el sonido.
* **public void reset() :** Este método sirve para reiniciar el sonido.

**SonidoManager:**

* **public static SonidoManager getInstance() :** Este método sirve para devolver la instancia de la clase tanto si existe como si no existe.
* **public SonidoManager() :** Este método es el constructor de esta clase, no pide ningún parámetro, pero crea un nuevo hashmap al llamarlo.
* **public Sonido getSonido(String \_clave) :** Este método sirve para devolver un sonido según la clave que le pases por parámetro.
* **public Sonido createSonido(String \_clave, String \_url) :** Este método sirve para crear un sonido y para ello se le pasa tanto la clave como la url por parámetro.

**ChampionSimulator:**

* **public void start(Stage stage) :** Este método sirve para iniciar la pantalla principal.
* **public static void main(String[] args ):** Este método main ejecuta el programa.
* **public void creacionSonidos(SonidoManager \_m) :** Este método sirve para crear todos los sonidos que hay en el programa.
* **public void limpiarBBDD() :** Este método sirve para limpiar la base de datos quiere decir dejar todos los campos que queramos en 0 para la ejecución de una nueva temporada.

**FXML\_VentanaElegirController:**

* **public void initialize(URL url, ResourceBundle rb) :** Este método sirve para inicializar la ventana ElegirController.
* **private void escogerRM(ActionEvent event) :** Este método sirve para que al tocar el botón del real Madrid se te configure la temporada para que tu equipo sea el real Madrid.
* **private void escogerSport(ActionEvent event) :** Este método sirve para que al tocar el botón del Sporting de Gijón se te configure la temporada para que tu equipo sea el Sporting de Gijón.
* **private void escogerInter(ActionEvent event) :** Este método sirve para que a la imagen del Inter de Milán se te configure la temporada para que tu equipo sea el Inter de Milán.
* **private void escogerBarsa(ActionEvent event) :** Este método sirve para que al tocar el botón del Barcelona se te configure la temporada para que tu equipo sea el Barcelona.
* **private void escogerArsenal(ActionEvent event) :** Este método sirve para que al tocar el botón del Arsenal se te configure la temporada para que tu equipo sea el Arsenal.
* **private void escogerLazio(ActionEvent event)** **:** Este método sirve para que al tocar el boton del Lazio se te configure la temporada para que tu equipo sea el Lazio,
* **private void escogerPSG(ActionEvent event)** **:** Este método sirve para que al tocar el botón del PSG se te configure la temporada para que tu equipo sea el PSG.
* **private void escogerCity(ActionEvent event) :** Este método sirve para que al tocar el botón del Manchester City se te configure la temporada para que tu equipo sea el Manchester City.
* **private void escogerBayern(ActionEvent event) :** Este método sirve para que al tocar el botón del Bayern Munich se te configure la temporada para que tu equipo sea el Bayern Munich.
* **private void escogerBETIS(ActionEvent event) :** Este método sirve para que al tocar el boton del Betis se te configure la temporada para que tu equipo sea el Betis.
* **private void pasarcanción(ActionEvent event) :** Este método sirve para pasar de canción al pulsar el boton indicando para ello.
* **private void mute (ActionEvent event) :** Este método sirve para mutear la música al tocar el botón correspondiente.
* **private void colocarImagenBotones() :** Este método sirve para colocar las imágenes de los botones de play y mute.

**FXML\_VentanaGanadorController:**

* **public void initialize(URL url, ResourceBundle rb) :** Este método inicializa el controlador de la ventana FXML\_VentanaGanador.
* **private void regresoTemporada(ActionEvent event) :** Este método sirve que para volver a la ventana temporada al pulsar el botón correspondiente.
* **public void cambioNombre() :** Este método sirve para cambiar la variable nombre.

**FXML\_VentanaInicioController:**

* **public void initialize(URL url, ResourceBundle rb)** : Este método inicializa el controlador de la ventana FXML\_VentanaInicio.
* **private void botonInicioClick(ActionEvent event) :** Este método sirve para que al pulsar el botón Inicio te lleve a la siguiente ventana que será para elegir a tu equipo
* **private void botonSalirClick(ActionEvent event) :** Este método sirve para cerrar la aplicación.
* **private void pasarcanción(ActionEvent event) :** Este método sirve para pasar canción al pulsar el boton play.
* **private void mute(ActionEvent event) :** Este método sirve para mutear la música al tocar el botón correspondiente.
* **private void colocarImagenBotones() :** Este método sirve para colocar las imágenes de los botones de play y mute.

**FXML\_VentanaPartidoController:**

* **public synchronized void start() :** Este metodo sirve para sincronizar los hilos de esta Ventana.
* **public void initialize(URL url, ResourceBundle rb) :** Este método inicializa el controlador de la ventana FXML\_VentanaPartido y dentro es donde se inicializan todo los métodos que no le pasen el parámetro evento y en este caso también inicializa los hilos.
* **public void getEquipos(ArrayList<Equipo> lista) :** Este método realiza una consulta a la tabla equipos para recoger toda la información de todos los campos que contiene esta tabla y agregarlos a un Arraylist que se le pasa por parámetro.
* **public void getVisitante() :** Este método sirve para recibir el visitante aleatorio que tiene que jugar contra nuestro equipo recoger sus estrellas y mostrar su escudo en la pantalla.
* **public void getEstrellasEquipos() :** Este método sirve para recoger las estrellas de los equipos tanto local como visitante.
* **public void cambioNombre() :** Este método Sirve para cambiar el valor de la variable nombre.
* **public void cambioVisitante() :** Este método sirve para cambiar el valor de la variable visitante.
* **public void setGOLESPARTIDO() :** Este método sirve para ir cambiando los label de valor según van marcando goles los equipos.

**FXML\_VistaTemporadaController:**

* **public void initialize(URL url, ResourceBundle rb) :** Este método inicializa el controlador de la ventana FXML\_VentanaTemporada, es donde se inicializan todo los métodos que no le pasen el parámetro evento y se cambia el logo según el equipo elegido.
* **private void FuncionIniciar(ActionEvent event) :** Este método sirve para que cuando pulses el botón iniciar inicie el partido con su vista correspondiente. También tiene encuentra si estan los 11 jugadores del 11 inicial.
* **private void FuncionSalir(ActionEvent event)** : Este método sirve para que cuando pulses el botón salir, se cierre el programa.
* **public void recibirParametro(String parametro) :** Este método sirve para recoger que equipo es el elegido por el usuario anteriormente y que se le ponga la ventana/escena configurada con su equipo.
* **public static void getTodosEquipos(ObservableList<Equipo> \_Eqlist) :** Este método devuelve la tabla equipos que será la tabla que tendrá la clasificación de esta liga.
* **public void getTodosJugadores(String \_nombre) :** Este método devuelve todos los jugadores correspondientes del equipo que ha seleccionado el jugador.
* **public void getPartidosTemporada(String nombre) :** Este método devuelve los partidos que tienen que jugarse esa jornada para ello tiene que coger todos los enfrentamientos posibles que puedan pasar almacenarlos en una lista, a continuación coger un enfrentamiento aleatorio el cual el local es el valor del parámetro nombre y luego aleatoriamente cojera 4 enfrentamientos más para esa jornada almacenándolos en otra lista.
* **private void FuncionMenu(ActionEvent event) :** Este método sirve para volver a la ventana inicio y e iniciar el juego otra vez o salir de la aplicación.
* **private void pasarcanción(ActionEvent event) :** Este método sirve para pasar canción al pulsar el botón play.
* **private void mute(ActionEvent event) :** Este método sirve para mutear la música al tocar el botón correspondiente.
* **private void colocarImagenBotones() :** Este método sirve para colocar las imágenes de los botones de play y mute.
* **private void Convocar(ActionEvent event) :** Este método lo que realiza es una ejecución en la consulta para recoger el valor del campo titularidad del futbolista seleccionado por el usuario y si es 0 se cambia 1 (0 quiere decir que esta convocado), si es 1 se cambia a 0 y a la vez se actualiza la base de datos para que en la tabla de alineación se actualice instantáneamente.

**Equipo:**

* **public public Equipo(int \_id, String \_nombre, int \_victorias, int \_derrotas, int \_goles, int \_golesc, int \_golesdiff, int \_estrellas) :** Constructor de la clase el cual al instanciarlo hay que pasarle 8 parámetro: id, nombre, victorias, derrotas, goles, goles en contra, goles de diferencia y estrellas. Y las guarde en variables de la clase.
* **public int getId() :** Este método sirve para recoger el id del equipo.
* **public String getNombre() :** Este método sirve para recoger el Nombre del equipo.
* **public int getVictorias() :** Este método sirve para recoger las Victorias del equipo.
* **public int getDerrotas() :** Este método sirve para recoger el Derrotas del equipo.
* **public int getGoles() :** Este método sirve para recoger los goles del equipo.
* **public int getGolesc() :** Este método sirve para recoger los goles en contra del equipo.
* **public int getGolesdiff() :** Este método sirve para recoger los goles de diferencia que tiene el equipo del equipo.
* **public int getEstrellas() :** Este método sirve para recoger las estrellas del equipo.
* **public void setId(int id) :** Este método sirve para cambiar el valor de la variable id.
* **public void setNombre(String nombre) :** Este método sirve para cambiar el valor de la variable nombre.
* **public void setVictorias(int victorias) :** Este método sirve para cambiar el valor de la variable victorias.
* **public void setDerrotas(int derrotas) :** Este método sirve para cambiar el valor de la variable derrotas.
* **public void setGoles(int goles) :** Este método sirve para cambiar el valor de la variable goles.
* **public void setGolesc(int golesc) :** Este método sirve para cambiar el valor de la variable golesc.
* **public void setGolesdiff(int golesdiff) :** Este método sirve para cambiar el valor de la variable golesdiff.
* **public void setEstrellas(int estrellas) :** Este método sirve para cambiar el valor de la variable estrellas.

**Jugador:**

* **public Jugador(int \_id, String \_nombre, String \_posicion, int \_titular) :** Constructor de la clase el cual al instanciarlo hay que pasarle 4 parámetro, id , nombre, posición y titular y estos parámetros se guardan en variables de clase.
* **public int getId() :** Este método sirve para devolver el valor de la variable id.
* **public String getNombre() :** Este método sirve para devolver el valor de la variable nombre.
* **public String getPosicion()** : Este método sirve para devolver el valor de la variable posición.
* **public int getTitular() :** Este método sirve para devolver el valor de la variable Titular.
* **public void setId(int id) :** Este método sirve para cambiar el valor de la variable id.
* **public void setNombre(String nombre) :** Este método sirve para cambiar el valor de la variable nombre.
* **public void setPosicion(String posicion) :** Este método sirve para cambiar el valor de la varible posicion.
* **public void setTitular(int titular) :** Este método sirve para cambiar el valor de la variable Titular.

**Partidos:**

* **public Partidos (String \_local,String \_visitante) :** Constructor de la clase el cual al instanciarlo hay que pasarle 2 parametros, local y visitante que se almacenaran en variables de clase.
* **public String getLocal() :** Este método sirve para devolver el valor de local.
* **public String getVisitante() :** Este método sirve para para devolver el valor de visitante.
* **public void setLocal(String local) :** Este método para settear el valor de la variable local.
* **public void setVisitante(String visitante) : Este** método para settear el valor de la variable visitante.

**HiloPartido:**

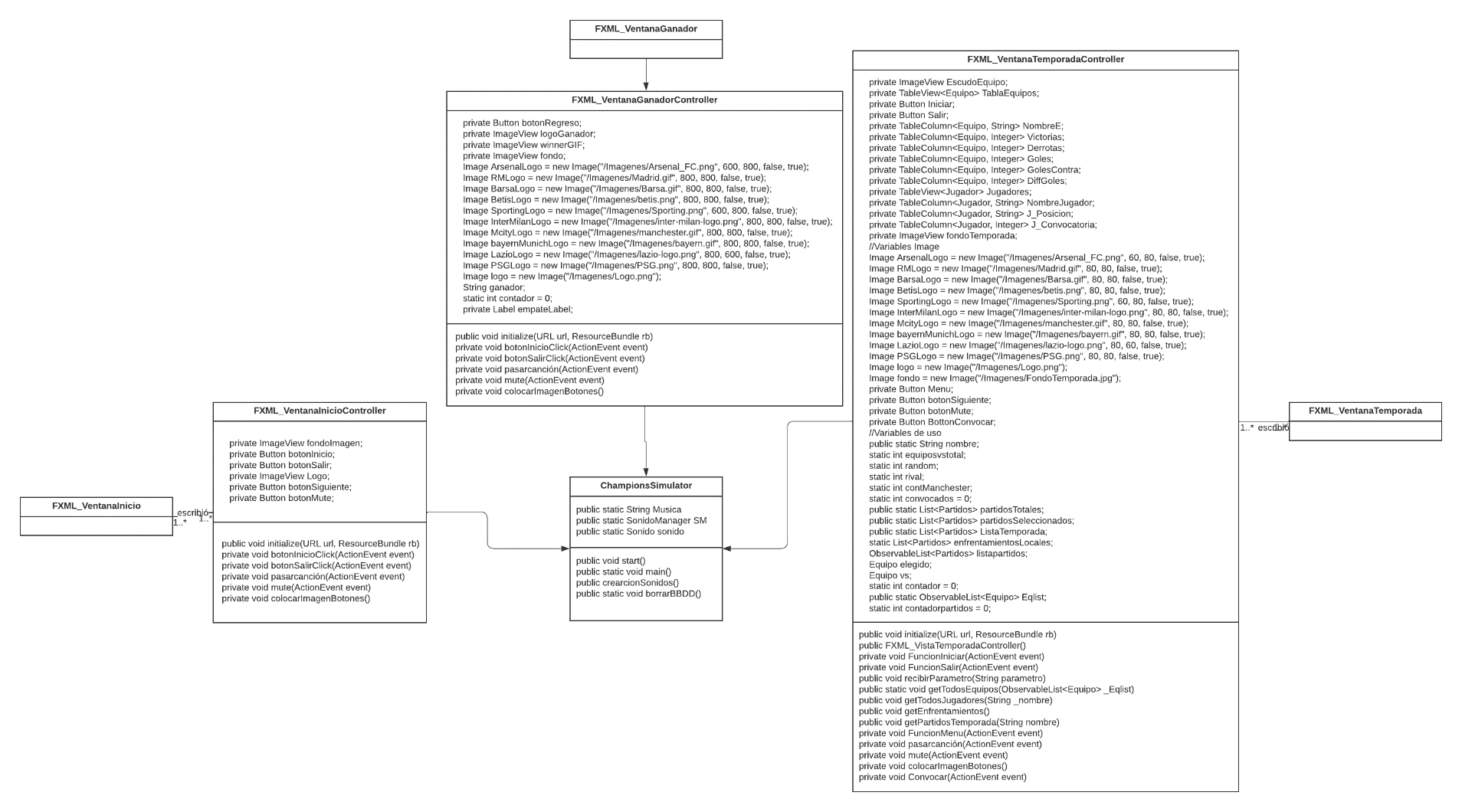
* **public HiloPartido(int estrellasEquipoLocal, int estrellasEquipoVisitante, Label lblGolesLocal, Label lblGolesVisitante, HiloTiempo ht, TextArea texto) :** Constructor de la clase el cual al instanciarlo hay que pasarle 6 parámetros, estrellasEquipoLocal, estrellasEquipoVisitante, lblGolesLocal, lblGolesVisitante, ht, texto y se guardan en variables de clase.
* **public void run() :** Este método run sirve para inicializar el timeline del partido que settea un textarea y dos label según los goles que marca un equipo y otro según la posibilidades que tengan para marcar gol según sus condiciones.
* **public int getCont1() :** Este método sirve para devolver el valor de cont1.
* **public int getCont2() :** Este método sirve para devolver el valor de cont2.
* **public void setCont1(int cont1) :** Este método sirve para cambiar el valor de la variable con1.
* **public void setCont2(int cont2) :** Este método sirve para cambiar el valor de la variable con2.

**HiloTiempo:**

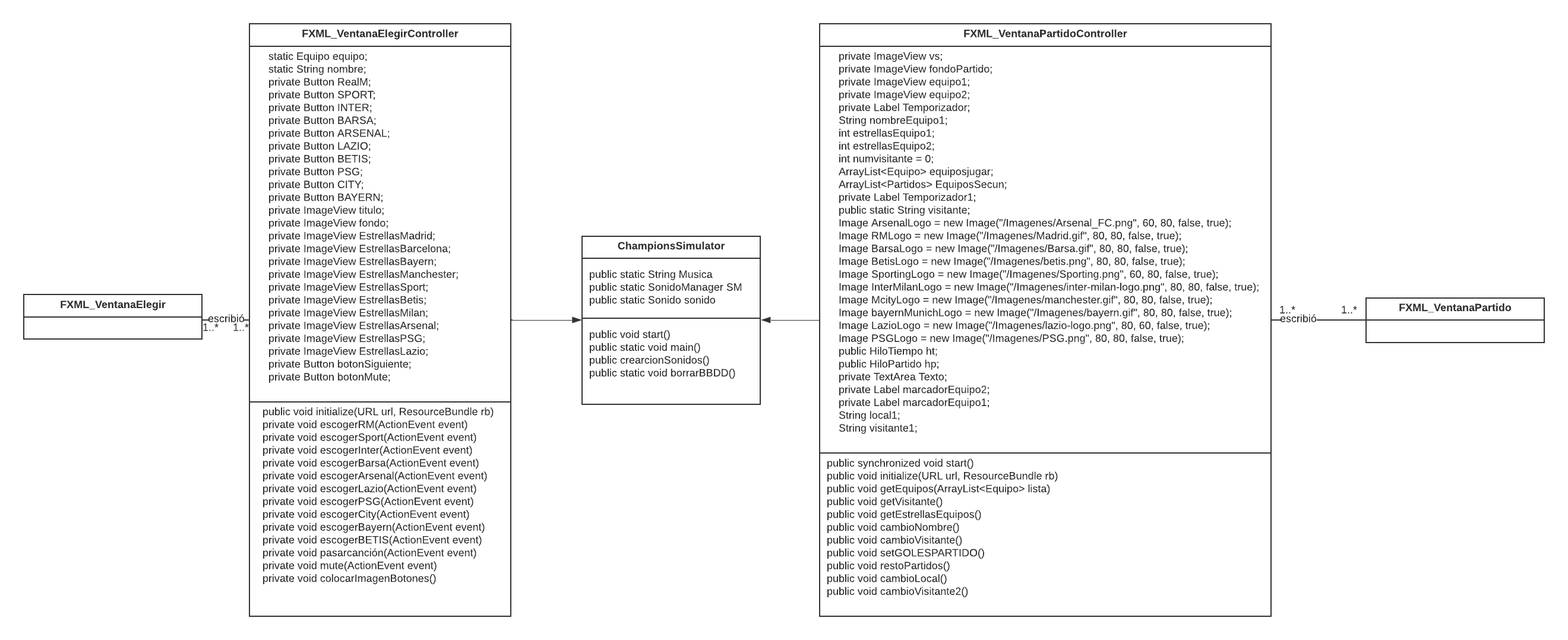
* **public HiloTiempo(Label \_label) :** Constructor de la clase el cual al instanciarlo hay que pasarle 1 parámetro, label que se guarda en una variable de clase.
* **public void run()** : Este método run sirve para llevar el tiempo del partido y pararlo cuando llegue el minuto 45 y reiniciarlo cuando en la alert le den a continuar. Cuando llega al final de la segunda parte inicializa la ventana Ganador

## Diagrama de clases

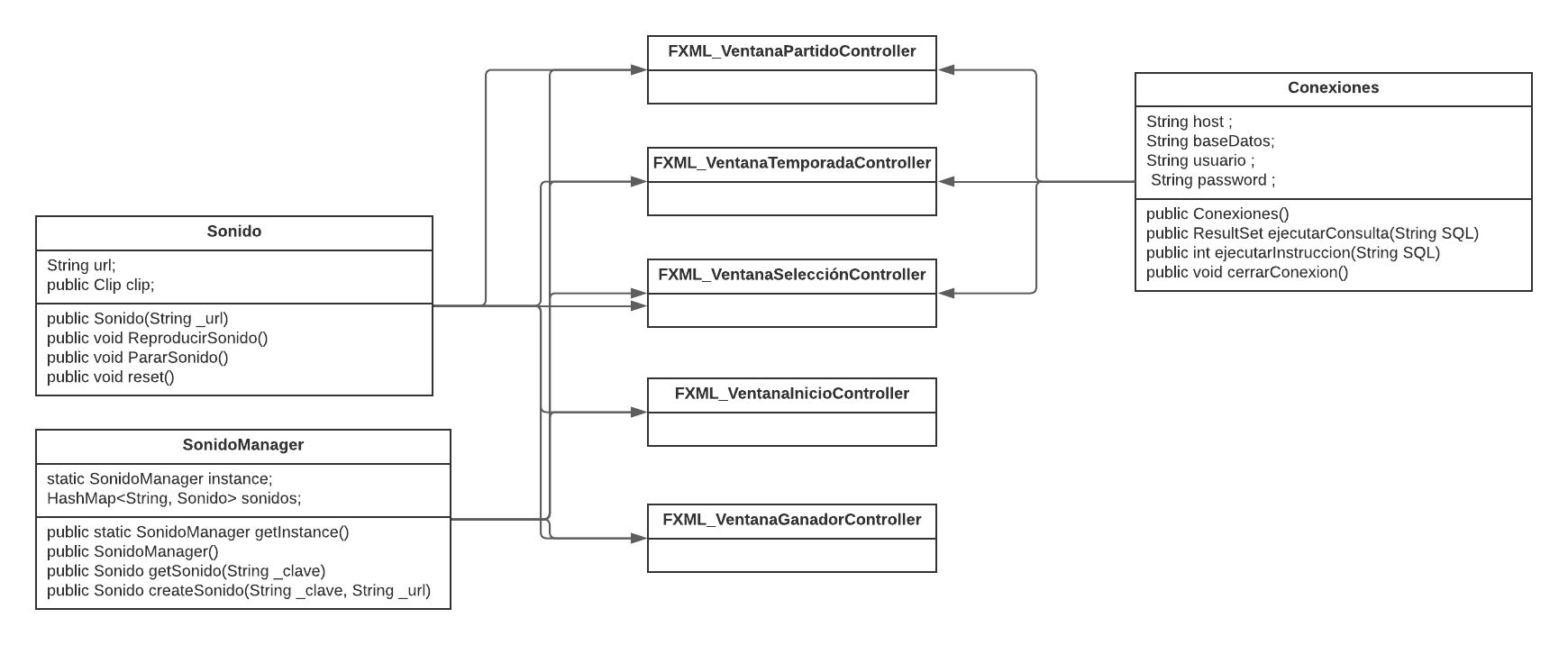




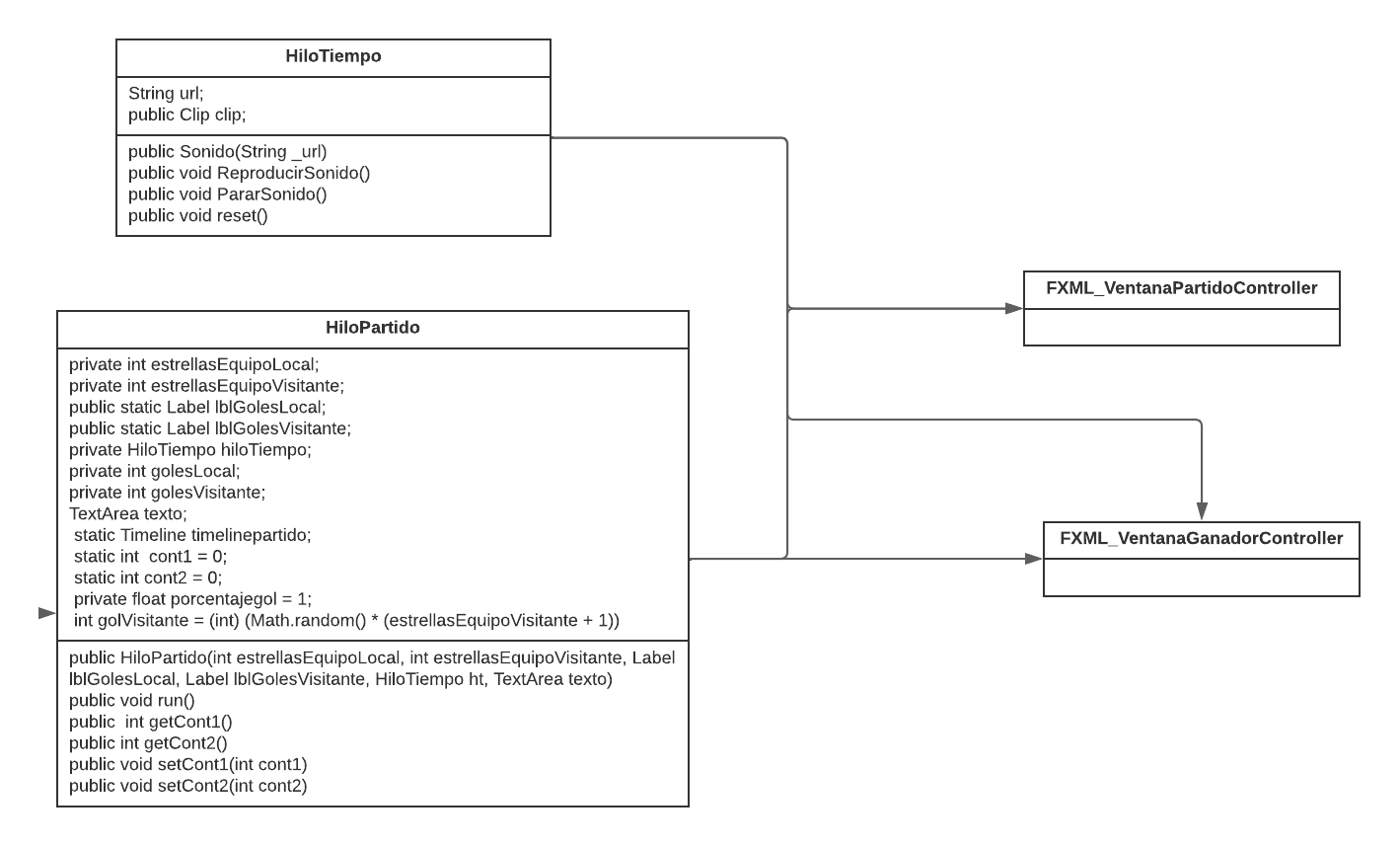
**Diagrama de Clases (Vistas 1)**

****

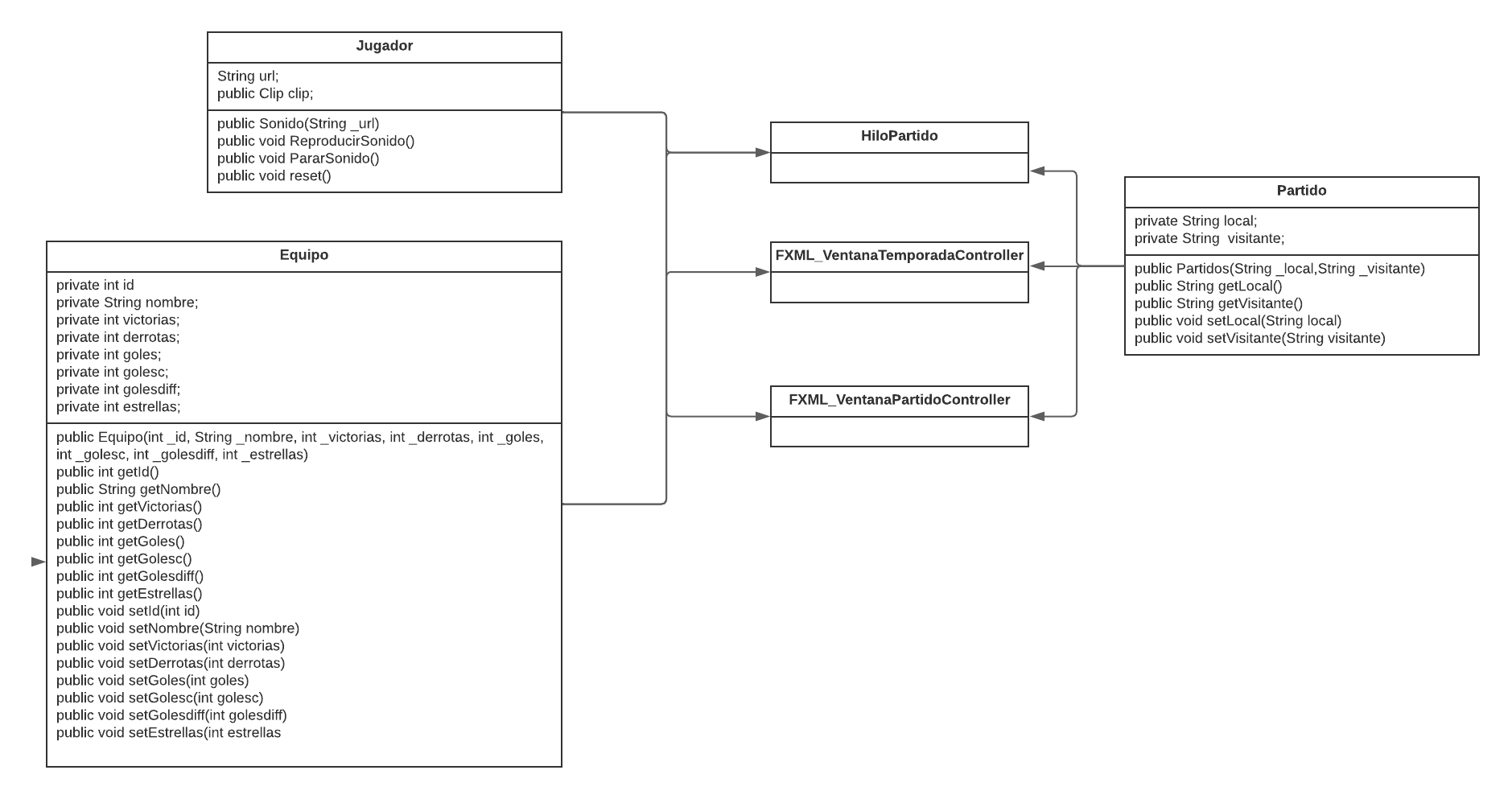
**Diagrama de Clases (Vistas 2)**



**Diagrama de Clases (Auxiliares)**

****

**Diagrama de Clases (Hilos)**



**Diagrama de Clases (Modelos)**

# Despliegue y pruebas

|  |  |
| --- | --- |
| Nº | **ESPECIFIACIÓN DE PRUEBAS** |
| 1º | **Objetivo:** Avanzar de la pantalla inicial a la de selección de equipo  **Pruebas realizadas:** Se pulsa el botón de inicio al ejecutar la aplicación, comprobamos que efectivamente, pasa a la pantalla de selección de equipo y la consola de ejecución no da ningún error. |
| 2º | **Objetivo:** Comprobar que el botón Salir cierra la aplicación  **Pruebas realizadas:** Se pulsa el botón de salir al ejecutar la aplicación y comprobamos que esta se cierra y termina su ejecución |
| 3º | **Objetivo:** Comprobar que cada botón de equipo es funcional y selecciona el equipo deseado una vez iniciada la partida  **Pruebas realizadas:** Se pulsan todos los botones de la pantalla selección y comprobamos que en la ventana temporada aparece la información del equipo seleccionado. |
| 4º | **Objetivo:** Comprobar que las tablas muestran el contenido verídico y actualizado de la BBDD  **Pruebas realizadas:** Al pasar a la pantalla temporada, se compara el contenido de las tablas mostradas en la aplicación con el que se encuentra en la BBDD y si son iguales es correcto. |
| 5º | **Objetivo:** Comprobar que el botón iniciar partido nos envía a la ventana partido y que el equipo que habíamos seleccionado anteriormente es el que aparece en la pantalla.  **Pruebas realizadas:** Una vez estemos en la ventana temporada, pulsamos el botón iniciar partido y comprobamos que nos envía a la ventana partido y que el equipo que sale a la izquierda (que sería el nuestro) es el que habíamos seleccionado en un inicio |

|  |  |
| --- | --- |
| 6º | **Objetivo:** Comprobar que la aplicación no te deja avanzar al partido si no tienes exactamente 11 jugadores convocados  **Pruebas realizadas:** Una vez en la ventana “Temporada”, convocamos un numero distinto de 11 jugadores con el botón convocar e intentamos avanzar hacia la ventana partido. Comprobamos entonces que el sistema muestra un alert y que no nos deja avanzar hacia la ventana partido |
| 7º | **Objetivo:** Comprobar que el botón Menú regresa al usuario al menú para poder reseleccionar equipo.  **Pruebas realizadas:** Se pulsa el botón de Menú y se comprueba que la aplicación te regresa a la ventana Menú. |
| 8º | **Objetivo:** Comprobar que el botón Salir de la ventana temporada cierra la aplicación  **Pruebas realizadas:** Se pulsa el botón de salir al llegar a la ventana temporada y comprobamos que esta se cierra y termina su ejecución |
| 9º | **Objetivo:** Comprobar que el temporizador del partido funciona  **Pruebas realizadas:** En cuanto llegamos a la ventana partido y este comienza, verificamos que el tiempo avanza, que al llegar a “45 mins” sale un alert indicando que hemos acabado la primera parte, que al pulsar en aceptar se reinicia y reanuda el temporizador comenzando la segunda parte y por último, al finalizar esta segunda parte, que aparezca un último alert que nos indique que el partido ha terminado. |
| 10º | **Objetivo:** Comprobar que el marcador de goles funciona  **Pruebas realizadas:** Se espera a que alguno de los dos equipos enfrentados marque un gol y este se simbolice en el marcador de goles y en el equipo correspondiente. |
| 11º | **Objetivo:** Comprobar que el JTextField muestra el progreso del partido  **Pruebas realizadas:** Vemos la ejecución del partido que se está disputando y comprobamos que tanto goles como tarjetas se simbolizan en el JTextField. |

|  |  |
| --- | --- |
| 12ª | **Objetivo:** Comprobar que el JTextField no deja ser modificado por el usuario.  **Pruebas realizadas:** Vemos la ejecución del partido que se está disputando y comprobamos que el Text Field no puede ser modificado. |
| 13º | **Objetivo:** Comprobar que, una vez acabado el partido, nos envía a la pantalla ganador y que esta muestra el equipo victorioso.  **Pruebas realizadas:** Vemos la ejecución completa del partido y cuando esta termina, verificamos que la aplicación nos lleva a la ventana ganador y que esta muestra el equipo que ha ganado la partida. |
| 14º | **Objetivo:** Comprobar que el botón volver a Temporada nos reenvía a la ventana Temporada y que los datos de la tabla temporada han sido actualizados en base a los resultados de los partidos disputados  **Pruebas realizadas:** Una vez llegados a la ventana ganador, pulsamos el botón “Regresar a Temporada” y vemos que la aplicación nos regresa a la ventana temporada y que los datos de la tabla temporada han sido actualizados correctamente. |
| 15º | **Objetivo:** Comprobar que, una vez acabado el primer partido, podemos volver a jugar más partidos y que estos no se repiten  **Pruebas realizadas:** Cuando acabamos la ejecución completa de un partido, comenzamos uno nuevo, vemos que este funciona correctamente y que además el contrincante es diferente al que teníamos en el primer partido. Lo comprobamos todas las veces que necesitamos hasta acabar y comprobar los 9 enfrentamientos disponibles. |
| 16º | **Objetivo:** Comprobación efectos de sonido  **Pruebas realizadas:** Comprobamos a lo largo de la ejecución del partido que suenan los silbatos al comienzo de cada parte y que cuando acaba el partido y sale el ganador, suena el efecto de sonido asignado al ganador. |

|  |  |
| --- | --- |
| 17º | **Objetivo:** Comprobación del botón pasar canción  **Pruebas realizadas:** En las tres pantallas en las que se encuentra este botón, verificamos que funciona correctamente y pasa a la canción siguiente una vez pulsado |
| 18º | **Objetivo:** Comprobación del botón mute  **Pruebas realizadas:** En las tres pantallas en las que se encuentra este botón, verificamos que funciona correctamente y quita la canción una vez pulsado. |
| 19º | **Objetivo:** Comprobación del funcionamiento de los botones que controlan la música una vez finalizada el partido y habiendo regresado a la ventana “Temporada”  **Pruebas realizadas:** Realizamos una ejecución completa del partido y una vez que llegamos a la ventana “Ganador”, pulsamos el botón de volver a temporada y una vez aquí, pulsamos ambos botones para comprobar que los controles de la música funcionan correctamente. |

# Conclusiones

Estas son las conclusiones que hemos obtenido una vez finalizado el desarrollo de nuestro proyecto durante los últimos 3 meses.

Nos ha encantado realizar este proyecto ya que, gracias a él, hemos aprendido mucho en muchos ámbitos de la programación. Nuevas funcionalidades de CSS, más opciones en JavaFX, el uso de GitHub y como organizar un proyecto de forma correcta mediante Trello. Creemos que ha sido una manera de avanzar un poco más como programadores y prepararnos para nuestro futuro laboral.

Hemos encontrado numerosas dificultades a la hora de realizar el proyecto debido a que, los conocimientos aprendidos en clase no nos eran suficientes para realizar todo lo implementado en el proyecto. Tuvimos que investigar bastante porque muchas veces era complicado encontrar una solución exacta para la duda que teníamos.

Estamos contentos con la versión final de nuestro proyecto ya que ha superado las expectativas que teníamos previstas en un inicio, según íbamos desarrollando la aplicación iban surgiendo ideas nuevas que implementar. También el habernos dado cuenta que se nos da bien el trabajo en equipo de manera organizada mediante el uso de la metodología Kanban.

Por último, debido al poco margen de tiempo con el que hemos contado y la complejidad de la programación que hemos desarrollado, nos ha sido imposible implementar alguna de las funciones que teníamos pensadas en un inicio como ha podido ser el guardado y cargado de la partida.

# Vías futuras

Con un poco más de tiempo para aprender y desarrollar la aplicación nos hubiera gustado introducir:

* **Menús dinámicos:** A la hora de seleccionar el equipo, hacer una interfaz animada que permitiera mediante el uso de flechas seleccionar el equipo que quieras utilizar.
* **Mercado de fichajes:** Implementar un mercado donde puedas comprar jugadores y elaborar un equipo a tu gusto.
* **Estadísticas de jugadores:** De la mano de la anterior propuesta, sería interesante que cada jugador tuviera sus estadísticas y a su vez, que estas influyan con el resto de jugadores, haciendo más interesante elaborar un equipo equilibrado para que tenga más posibilidades de ganar.
* **Más equipos, distintas copas:** Nos gustaría introducir más equipos incluyendo las selecciones para poder hacer otros modos de juego como podría ser el mundial.
* **Creación de usuarios para el modo online:** elaborar una ventana de registro e inicio de sesión para poder almacenar los datos de los jugadores.
* **Elaborar un Ranking online:** Con la implementación de la posibilidad de crear nuestros propios equipos también nos gustaría crear un ranking online donde la gente pueda competir por un primer puesto en cuanto a número de partidos ganados, goles, copas ganadas etc.
* **Elaborar un sistema de guardado y cargado de partida:** Debido a la complejidad de la programación y al poco tiempo del que disponemos, no tuvimos tiempo de elaborar un sistema eficaz y funcional de guardado y cargado de partida.

# Glosario

* **IDE:** Un entorno de desarrollo integrado​​ o entorno de desarrollo interactivo, es una aplicación informática que proporciona servicios integrales para facilitar al desarrollador o programador el desarrollo de software.
* **Sony Vegas:** Vegas Pro, anteriormente conocido como Sony Vegas, es un software de edición de video de tipo edición no lineal y editor de audio diseñado para PC. Está orientado tanto a la edición de vídeos profesionales como al mercado de consumidores.
* **Photoshop:** Adobe Photoshop es un editor de fotografías desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos
* **Java:** Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems.
* **JavaFX**: JavaFX es una familia de productos y tecnologías de Oracle Corporation, para la creación de Rich Internet Applications, esto es, aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio, incluyendo aplicaciones multimedia interactiva.
* **BBDD**: Una base de datos es un conjunto de datos organizado de manera que permite obtener rápidamente la información que se necesita en cada momento.
* **Método:** En programación, método se le llama a un bloque de código que tiene definido en su interior un conjunto de instrucciones, estas instrucciones realizan una determinada tarea.
* **POO:** La Programación Orientada a Objetos es un [paradigma de programación](https://profile.es/blog/que-son-los-paradigmas-de-programacion/), es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él. Se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

# Bibliografía/Webgrafía

## Información

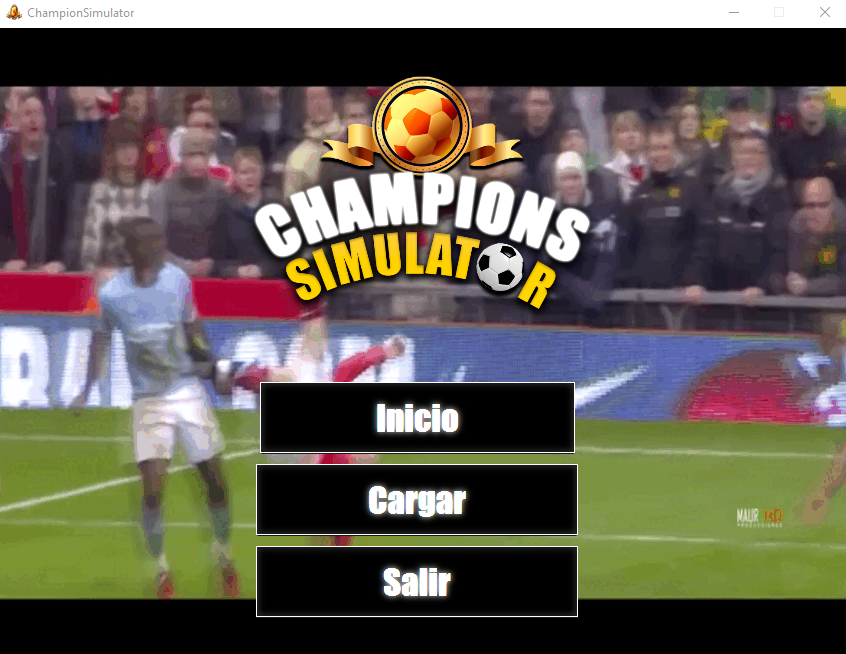
* Colaboradores de Wikipedia. (2023). Entorno de desarrollo integrado. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado>.
* Contenidos. (2020, 27 agosto). ¿Qué es Photoshop y para qué sirve? - Neo Wiki | NeoAttack. NeoAttack. <https://neoattack.com/neowiki/photoshop/>
* Colaboradores de Wikipedia. (2023a, marzo 23). Base de datos. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Base_de_datos>
* School, T. (2022). ¿Qué es JavaFX y para que se utiliza? Tokio School. <https://www.tokioschool.com/noticias/que-es-javafx-usos/>
* Colaboradores de Wikipedia. (2023a). VEGAS Pro. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/VEGAS_Pro>
* Colaboradores de Wikipedia. (2023d). Java (lenguaje de programación). Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)>
* Canelo, M. M. (2022). ¿Qué es la Programación Orientada a Objetos? Profile Software Services. <https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/>
* GameDevTraum. (2023). Qué es un "Método en programación. <https://gamedevtraum.com/es/programacion-informatica/teoria-de-programacion/que-es-metodo-programacion/#:~:text=En%20programaci%C3%B3n%2C%20m%C3%A9todo%20se%20le,instrucciones%20realizan%20una%20determinada%20tarea>.

## Código

* Midudev. (2022, 21 noviembre). ESPECTACULAR efecto para tu BOTÓN con sólo CSS [Vídeo]. YouTube. <https://www.youtube.com/watch?v=QR8CVkmJhLU>
* Platzi. (s. f.). JavaFX + CSS  (Aplicar Estilos CSS en JavaFX). <https://platzi.com/tutoriales/1631-java-basico/9242-javafx-css-aplicar-estilos-css-en-javafx/>
* De Programación, T. (2016, 15 marzo). Introducción a CSS en JavaFX. Tutor de programación. <http://acodigo.blogspot.com/2016/03/introduccion-css-en-javafx.html#:~:text=Agregar%20Archivo%20CSS%20a%20JavaFX&text=En%20c%C3%B3digo%20Java%20tenemos%20que,css%E2%80%9D>).
* Jakob, M. (2014, 17 septiembre). Parte 4: Hojas de estilo CSS | Tutorial JavaFX (Español) | code.makery.ch. <https://code.makery.ch/es/library/javafx-tutorial/part4/>
* De Programación, T. (2015, 12 noviembre). Estilos CSS para TableView JavaFX. Tutor de programación. <http://acodigo.blogspot.com/2015/11/estilos-css-para-tableview-javafx.html>

# Anexos

## 12.1 Imágenes diseño aplicación

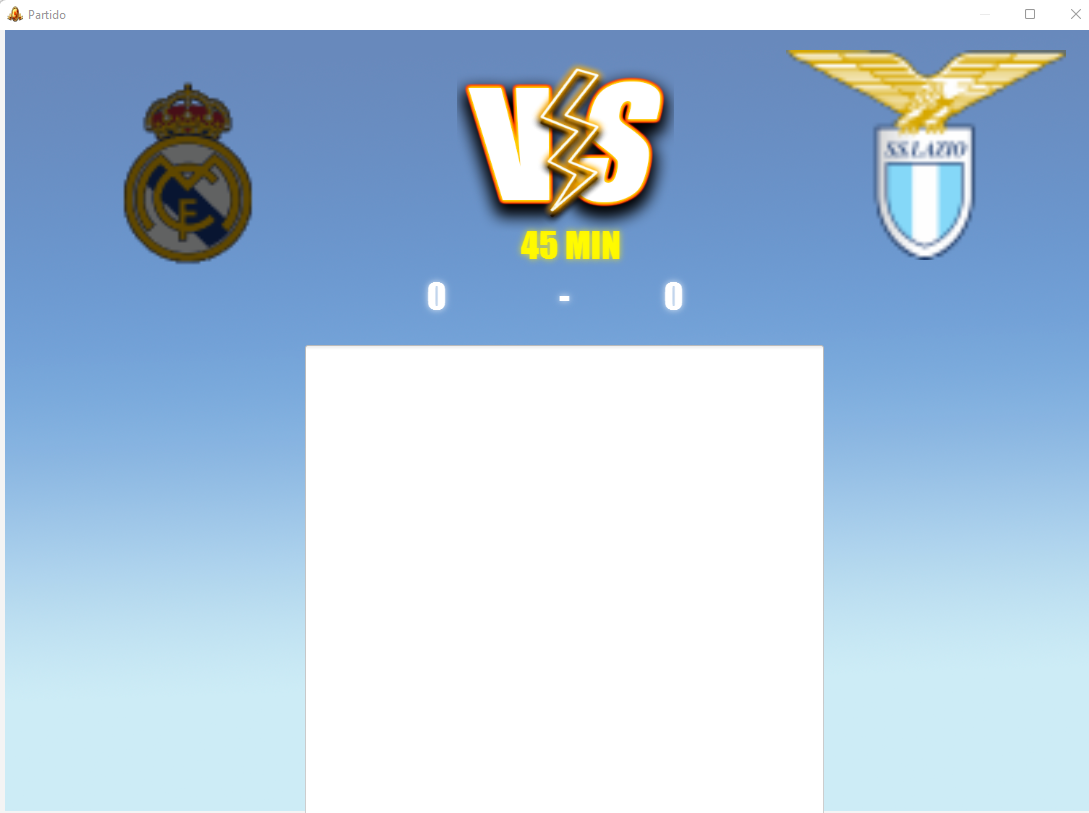




**Diseño segunda pantalla**

**Diseño primera pantalla**

**Diseño ventana Temporada**



**Diseño ventana Partido**