UNIVERSIDAD POLITÉCNICA DE VALENCIA
ESCUELA TÉCNICA SUPERIOR DE INFORMÁTICA APLICADA

DESARROLLO DE APLICACIÓN GRÁFICA AUTOMATIZADA PARA TELÉFONOS MÓVILES

PROYECTO FIN DE CARRERA

Iván Gómez Badenes

Antonio Cano Gómez

29 de Junio de 2010

ÍNDICE DE CONTENIDOS

1. INTRODUCIÓN

1.1. DESCRIPCIÓN DE LA APLICACIÓN

El desarrollo de la aplicación basada en un dispositivo móvil con sistema operativo Maemo se enmarca dentro del proceso de realización del proyecto final de carrera por el alumno Iván Gómez Badenes.

La aplicación móvil a desarrollar y cuyo desarrollo en su totalidad se especificará a lo largo de este documento se corresponde con el proyecto II/LD-A-DSIC- 25/09 dirigido por el profesor del departamento DSIC Antonio Cano Gómez.

El software desarrollado consiste en una aplicación de gestión de un equipo deportivo, en este caso, un equipo de fútbol en la cual, el usuario podrá gestionar los datos relativos a los jugadores de su equipo, las incidencias durante el partido, los pormenores de los entrenamientos, etc. con el fin de almacenar toda la información que considere necesaria para optimizar el rendimiento de su equipo en base a unas estadísticas y unos datos evolutivos del mismo tomados a lo largo del tiempo y en base a ellos tomar las decisiones de trabajo que se consideren adecuadas.

1.2. OBJETIVOS

Se plantean como objetivos a satisfacer con el desarrollo de esta aplicación el facilitar la labor de gestión de un equipo de fútbol a su usuario final. La aplicación pretende presentar un método práctico de tener toda la información relativa al equipo de fútbol en cuestión en el dispositivo móvil para así poder analizarla en cualquier momento que se desee, huyendo así de la restricción que supone la necesidad de ir cargado con papeles llenos de estadísticas para poder analizar las cuestiones que se deseen.

Así pues, la ventaja que presenta esta aplicación respecto a la toma de datos en papel o en un PC o un portátil, es la portabilidad. La información se toma sobre el propio dispositivo a través de la interfaz del programa, y se analiza también desde el propio dispositivo, pudiendo ser consultada en cualquier momento y en cualquier lugar con el único y obvio requisito de llevar el dispositivo móvil encima. Asimismo, se garantiza una arquitectura de la información clara y estructurada siguiendo el esquema de una máquina de estados en la cual estarán bien definidos los caminos a seguir para encontrar según qué información, beneficio éste respecto al mantenimiento de información en papel tomada a pie de campo, ya que ésta se puede desordenar, los folios se pueden traspapelar o incluso perder.

Por otra parte, destacar que el objetivo de la aplicación es proporcionar un soporte al usuario en distintos ámbitos.

El primero de ellos sería a la hora de entrenar, en la cual, se podrá gestionar la asistencia de los jugadores a los entrenamientos, las incidencias acontecidas, la duración del entrenamiento y tener una lista de ejercicios llevados a cabo en cada entrenamiento y la duración de estos ejercicios para así poder gestionar y planificar los entrenamientos en base a lo realizad en los anteriores entrenamientos o en base al criterio que el usuario decida. Así pues, esta parte de la aplicación, podría corresponderse con el contenido de la Tabla 1, en caso de que esta información fuese tomada en papel en lugar de hacerse con el dispositivo móvil.

Duración del entrenamiento	**1 hora 35 minutos**

Ejercicio	**Duración**
Calentamiento	30 minutos
Centros al área y remates	20 minutos
Lanzamientos a puerta	20 minutos
Tanda de penaltis	15 minutos
Estiramientos	10 minutos

Jugador	**Rendimiento**	**Incidencias**	**Comentarios**
Javier García	Bueno	-	Debe trabajar los centros al área
Rubén Mora	Regular	Se retira con molestias a los 35 minutos	-

- Tabla 1 -

En segundo lugar la aplicación proporcionaría una interfaz de toma de datos a pie de campo durante el partido, estos datos acerca de los jugadores, del rendimiento del equipo (goles a favor, goles en contra, en qué minutos el equipo rinde más y en cuáles rinde menos) se almacenan y se usarán para elaborar unas estadísticas acerca del rendimiento del equipo para así determinar cuáles son las facetas a potenciar y cuáles las que deben ser corregidas. La toma de datos a pie de campo se correspondería si se realizara en papel, de modo análogo a lo estudiado para el caso anterior, con algo similar al contenido de la Tabla 2.

Equipo Rival: Lepe C.F.	Clasificación del rival: 9º

Alineación inicial	Suplentes
Jugador 1	Jugador 12
...	...
Jugador 11	Jugador 18

Goles a favor	
Autor	**Minuto**
Jugador 7	25
Jugador 9	74
Jugador 10	87

Goles en contra	
Minuto	**Comentario**
10	Jugada a balón parado desde la banda
82	Error de marcaje de Jugador4

Tarjetas		
Jugador	**Amonestación**	**Minuto**
Jugador3	Amarilla	65

Tarjetas provocadas		
Jugador	**Amonestación provocada**	**Minuto**
Jugador7	Amarilla	40

Sustituciones		
Jugador sustituido	**Jugador que entra al campo**	**Minuto**
Jugador9	Jugador14	81

Incidencias		
Jugador	**Minuto**	**Comentarios**
Jugador9	81	Sustituido por molestias en el gemelo derecho

- Tabla 2-

Será también objetivo de la aplicación poder usar y manipular todos los datos recogidos del modo análogo a cómo se tomarían en papel para elaborar una serie de estadísticas generales que permitieran al usuario estar al tanto de cualquier detalle, por mínimo que éste sea, como, por ejemplo, el jugador más amonestado, el que más tarjetas provoca a los contrarios, el jugador más convocado, el más sustituido, promedios goleadores, etc.

1.3. PÚBLICO OBJETIVO

Una vez descrita la aplicación y los objetivos que con su implementación se desean cubrir y satisfacer, es momento de definir a qué sector de la población está destinada o cuáles serán sus potenciales usuarios finales.

Es de gran importancia este punto, ya que el diseño y posterior desarrollo de la aplicación dependerá de quién sea ese público objetivo para poder crear un software que se adecúe lo máximo a las necesidades de estos usuarios. Para obtener los resultados óptimos, será necesario saber el nivel de conocimientos tanto informáticos como deportivos, por ser éste el marco en el cual se desarrolla este proyecto , que tendrá un entrenador o una entrenadora de un equipo de fútbol.

Así pues, se determina como público objetivo al sector de la sociedad cuyo cometido, tanto si es a tiempo total o a tiempo parcial, es entrenar un equipo de fútbol de cualquier categoría (alevín, infantil, cadete, juvenil...). Por supuesto, estos entrenadores deberán disponer de un dispositivo móvil Maemo para poder integrar en él la aplicación.

Basándonos en el público objetivo y poniéndonos en la piel de un entrenador y del uso que éste haría del software, deducimos que la aplicación, por el hecho de ser para un uso a pie de campo a modo de "libreta" donde apuntar los acontecimientos que en un partido y/o entrenamiento acaecen, deberá ser de un uso fácil e intuitivo. La aplicación deberá tener una arquitectura de la información muy bien definida dónde sea fácil encontrar la información que el usuario final desee obtener o dónde encontrar el formulario para introducir los datos de una manera inequívoca de modo que se le dé robustez al software.

Por otra parte, deberemos distinguir a la hora de la implementación, qué casos de uso serán utilizados en según qué circunstancia. Por ejemplo, no se deberá afrontar del mismo modo la gestión inicial de los jugadores (Añadir nuevo jugador al equipo, editar datos de un jugador...) que, por ejemplo, la gestión e introducción de datos a pie de campo o en un partido (Registrar gol a favor, registrar amonestación, introducir alineación inicial...), ya que en el primero de los casos, el usuario tendrá que tomarse un tiempo para rellenar los campos correspondientes detenimiento, mientras que, por otra parte, en el segundo de los casos, la información que se tomará durante los partidos y/o entrenamientos deberá ser fácilmente localizable dónde apuntarla, y rápido el poder llevar a cabo este registro, ya que en un partido suceden muchas incidencias y el entrenador no puede estar dedicando un tiempo excesivo a anotar, por ejemplo una tarjeta amarilla.

1.4. LA APARICIÓN DE MeeGo

Durante el proceso de desarrollo del proyecto que nos ocupa, ha aparecido la noticia de que Nokia e Intel han anunciado la aparición de un nueva plataforma open source basada en el sistema operativo Linux que se llamará MeeGo, creada para Smartphone, netbooks y otros dispositivos destinados a la comunicación móvil.

Según se comunicó en la rueda de presa de la pasada edición del *Mobile World Congress*, Moblin (basado en Fedora) se unirá con el sistema Maemo que Nokia utiliza en su Smartphone Nokia N900, y se esperaba que la nueva plataforma (MeeGo) se introduzca en el mercado durante el segundo cuarto del presente año, es decir, entre primeros abril y finales de junio de 2010. El fin de esta plataforma no es remplazar el actual sistema operativo Symbian según Nokia, si bien es cierto que éste quedará operativo para una cantidad más reducida de dispositivo cuando la nueva plataforma sea lanzada.

El nuevo sistema operativo combina el núcleo Moblin, desarrollado por Intel, y el toolkit de interfaces gráficas de Maemo, y soportará tanto la arquitectura Atom de Intel, como la arquitectura ARM.

Previamente, Nokia había estado desarrollando de manera independiente su plataforma Maemo para sus dispositivos, mientras que Intel Moblin ha sido diseñado principalmente para netbooks. Aunque las dos plataformas basadas en Linux tienen mucho en común, hay un número de diferencias técnicas que podrán ser a priori poco sencillas de “reconciliar” por parte de los desarrolladores

Asimismo, Nokia se ha posicionado afirmando que en lo sucesivo lanzará teléfonos móviles con MeeGo. Sin embargo, según Intel, más compañías y operadores anunciarán en las próximas semanas (citado el 18 de febrero de 2010) su apoyo y planes de producción para MeeGo.

Así pues, con la aparición de esta nueva plataforma aparece una gran ventaja para los desarrolladores que trabajan con librerías Qt de Nokia, como sucede en este caso, que es la creación de aplicaciones multiplataforma, ya que creando la aplicación en PC con el entorno Qt Creator y con los ficheros fuentes, podremos migrar los códigos de los proyectos no sólo a dispositivos Maemo, sino también a Pocketables, dispositivos In-vehicle (de uso en el interior de los vehículos), netbook, televisión y media phones, así como en móviles con sistema Symbian.

Las últimas noticias aparecidas a propósito de esta nueva plataforma datan del 31 de mayo de 2010, (cumpliendo con los plazos previstos meses atrás) en las cuales se comunica que Nokia e Intel han lanzado MeeGo v1.0, una versión para netbooks y terminales móviles que estará presumiblemente disponibe para tablets en octubre de este mismo año. Según se comunica, versión será MeeGo NetBook, y avanzado el mes de junio, periodo en el cual se está redactando el presente documento, estará disponible la de móviles, MeeGoandset, que en principio estaría disponible únicamente para los terminales Nokia N900 con vistas a extenderse al resto de modelos. MeeGo v1.0 incorpora la nueva API de MeeGo, la cual incluye la plataforma de desarrollo Qt 4.6 (plataforma bajo la cual se enmarca y con la que se ha realizado el desarrollo de este proyecto), el nuevo SDK de MeeGo con un entorno de desarrollo integrado y varias herramientas más, aún por hacerse públicas, para otros sistemas operativos.

Asimismo, se ha hecho pública la fecha de la nueva versión MeeGo v1.1, preparada para octubre de 2010 y que incluirá soporte para dispositivos táctiles tales como terminales móviles, tablets o sistemas de informacióny entretenimiento para coches. En caso de ser estas afirmaciones ciertas, el tablet de Nokia, cuya salida a mercado se espera para otoño del presente añoia incorporar MeeGo en vez de

Windows, tal y como se había anunciado a principios del mes de abril de 2010. El 7 de abril de 2010, la agencia Reuters se hizo eco de las afirmaciones vertidas por el analista Ashok Kumar (Rodman and Renshaw) en las cuales se decía que, tras un análisisde los últimos signos y pasos que estaba dando la compañía finlandesa, concluyó que "Ahora mismo la cadena de suministro está siendo preparada para un lanzamiento en otoño. Será a mediados de septiembre-octubre para responder a la demanda en Navidades". Por aquellas fechas se presumía que Nokia no quería quedar atrás en la carrera por el mercado tableta, especialmente tras el éxito de Apple y su iPad, así como que el sistema operativo elegido sería Windows debido a que MeeGo no estaba suficientemente desarrollado como para poder cubir las necesidades de un equipo de las caracterísicas de un tablet.

Ahora pues, es momento de esperar acontecimientos con las expectativas puestas en el crecimiento de la plataforma que nos ocupa en este proyecto, y disponernos a ser testigos la batalla mercantil tento en el sector de la telefonía mo´vil como en el de los tablets, ya que en este último, varios fabricantes como Samsung o Hp parece han apostado por este mercado. Aunque de momento podemos destacar de MeeGo v1.0 su simplicidad y su agradable interfaz de usuario con colores arctivos y isualmente bastant atractiva, con iconos muy trabajados. Desde un punto de vista más técnico podría en un principio tacharse al sistema como de rápido arranque y conexión a Internet secundario, para momentos puntuales, si bien, se destaca que puede funcionar como sistema operativo principal de la computadora sin excesivos problemas.

De momento, abierto a todo el publico, podemos encontrar toda la información sobre MeeGo v1.0 y sus herramientas de desarrollo en:

http://meego.com/community/blogs/imad/2010/meego-v1.0-core-software-platform-netbook-user-experience-project-release.

- MeeGo v1.0. Imagen tomada de la página oficial de MeeGo: http://meego.com -

2. FASE DE DISEÑO

Llegados a este punto y teniendo en cuenta todo lo anteriormente expuesto, nos encontramos, ahora sí, en disposición de poder elaborar un estudio previo de cómo será la aplicación, el aspecto de las pantallas, de cuántas estará compuesto el sistema, y determinar qué información será relevante almacenar en la misma y cuáles serán los datos a procesar y qué resultados desearemos obtener de ese procesado, apoyado todo esto con un diagrama de clases inicial y un modelo a priori de casos de uso, ambos susceptibles de ser modificados a lo largo del proceso de implementación.

Asimismo, se definirá en los puntos sucesivos, y mediante las diversas pantallas, la arquitectura de la información y los accesos a la misma, modelizado todo esto como una máquina de estados, que sostendrá el correcto funcionamiento de la aplicación.

2.1. DIAGRAMA DE CLASES

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

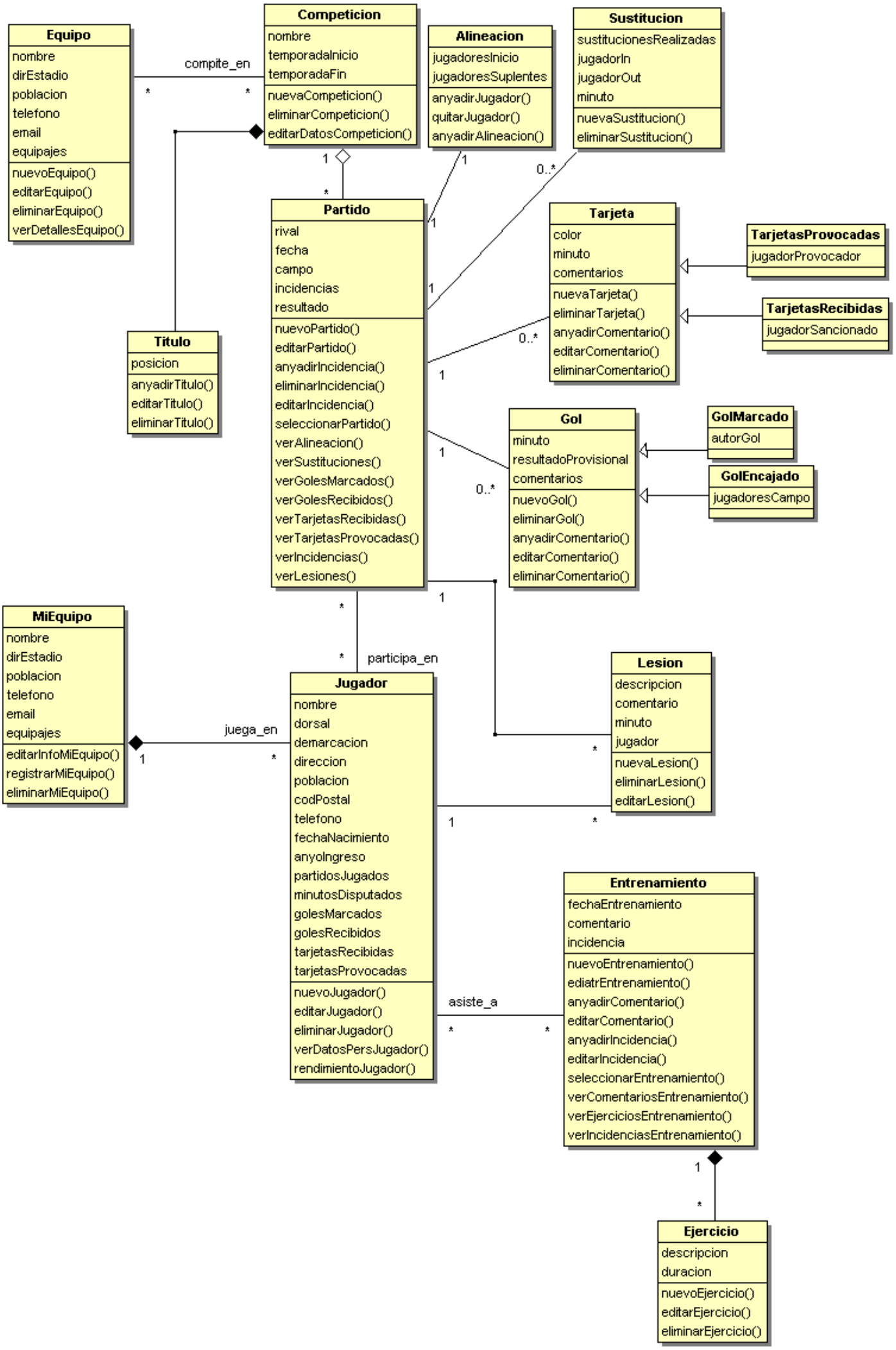
Por tanto, es importante elaborar un buen diagrama de clases que nos facilite la labor de implementación de la misma basándonos en un estudio previo de las necesidades que tendremos a la hora de programas y generar código C++ .

El diagrama de clases que se muestra en su totalidad en la página siguiente para una cómoda lectura e interpretación, encontramos las clases necesarias y las relaciones de asociación, especialización/generalización y agregaciones tanto inclusivas (en las que cada componente puede pertenecer a lo sumo a un compuesto y la destrucción del compuesto implica la destrucción del componente) como referenciales o de catálogo (en las cuales los componentes son reutilizables a lo largo de diferentes compuestos y los tiempos de vida no están relacionados) que para el correcto funcionamiento de la aplicación se requieren.

Nótese que este diagrama de clases puede verse modificado a lo largo de la programación de la aplicación, en su forma, añadiendo nuevas clases modificando atributos métodos de las ya diseñada, etc.

El fin de este diseño es aproximar la estructura de los datos, la forma en que la información será almacenada y proporcionar una primer detalle de cómo estos podrán ser accedidos, desde dónde y en qué puntos se podrá disponer de según qué información.

A partir del diagrama de clases obtenemos de una manera rápida el modelo relacional de la base de datos que da cobertura a las transacciones y operaciones que la aplicación debe ser capaz de realizar.


Equipo
nombre
dirEstadio
poblacion
telefono
email
equipajes
nuevoEquipo()
editarEquipo()
eliminarEquipo()
verDetallesEquipo()
compite_en
Competicion
nombre
temporadaInicio
temporadaFin
nuevaCompeticion()
eliminarCompeticion()
editarDatosCompeticion()
Alineacion
jugadoresInicio
jugadoresSuplentes
anyadirJugador()
quitarJugador()
anyadirAlineacion()
Sustitucion
sustitucionesRealizadas
jugadorIn
jugadorOut
minuto
nuevaSustitucion()
eliminarSustitucion()
Partido
rival
fecha
campo
incidencias
resultado
nuevoPartido()
editarPartido()
anyadirIncidencia()
eliminarIncidencia()
editarIncidencia()
seleccionarPartido()
verAlineacion()
verSustituciones()
verGolesMarcados()
verGolesRecibidos()
verTarjetasRecibidas()
verTarjetasProvocadas()
verIncidencias()
verLesiones()
Tarjeta
color
minuto
comentarios
nuevaTarjeta()
eliminarTarjeta()
anyadirComentario()
editarComentario()
eliminarComentario()
TarjetasProvocadas
jugadorProvocador
TarjetasRecibidas
jugadorSancionado
Titulo
posicion
anyadirTitulo()
editarTitulo()
eliminarTitulo()
Gol
minuto
resultadoProvisional
comentarios
nuevoGol()
eliminarGol()
anyadirComentario()
editarComentario()
eliminarComentario()
GolMarcado
autorGol
GolEncajado
jugadoresCampo
MiEquipo
nombre
dirEstadio
poblacion
telefono
email
equipajes
editarInfoMiEquipo()
registrarMiEquipo()
eliminarMiEquipo()
juega_en
participa_en
Jugador
nombre
dorsal
demarcacion
direccion
poblacion
codPostal
telefono
fechaNacimiento
anyoIngreso
partidosJugados
minutosDisputados
golesMarcados
golesRecibidos
tarjetasRecibidas
tarjetasProvocadas
nuevoJugador()
editarJugador()
eliminarJugador()
verDatosPersJugador()
rendimientoJugador()
Lesion
descripcion
comentario
minuto
jugador
nuevaLesion()
eliminarLesion()
editarLesion()
asiste_a
Entrenamiento
fechaEntrenamiento
comentario
incidencia
nuevoEntrenamiento()
ediatrEntrenamiento()
anyadirComentario()
editarComentario()
anyadirIncidencia()
editarIncidencia()
seleccionarEntrenamiento()
verComentariosEntrenamiento()
verEjerciciosEntrenamiento()
verIncidenciasEntrenamiento()
Ejercicio
descripcion
duracion
nuevoEjercicio()
editarEjercicio()
eliminarEjercicio()

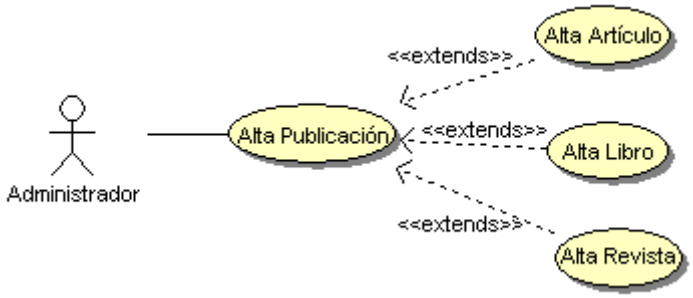
2.2. MODELO DE CASOS DE USO

Los casos de uso son la técnica que emplearemos en el proceso de análisis previo al desarrollo del proyecto para la captura de requisitos potenciales de la aplicación software a implementar. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario para conseguir un objetivo específico.

Cada caso de uso representará una secuencia de interacciones que se desarrollarán entre el sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. En este caso particular, por el hecho de no interactuar con otros sistemas, el único actor que se halla es el usuario final. Así pues, utilizamos el modelo de casos de uso con el fin de especificar la comunicación y el comportamiento de un sistema mediante su interacción con el usuario.

En las próximas páginas encontraremos el diagrama de casos de uso, así como las tablas que describen a cada uno de ellos agrupados según grupos de interés relacionados a criterio del programador, independientes de la máquina de estados que representa la aplicación, el diagrama de clases, o el propio diagrama que a continuación se presenta.

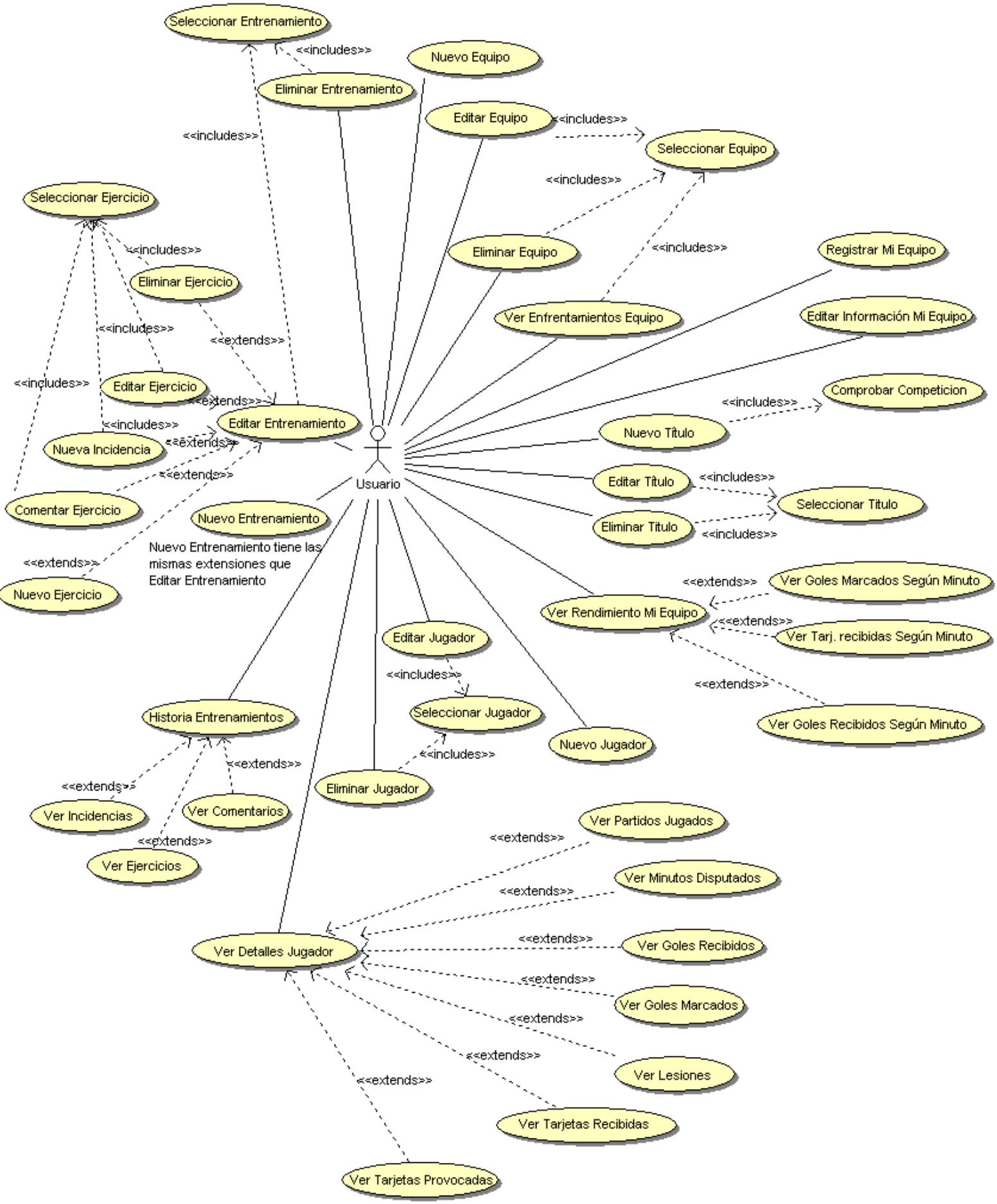
Las relaciones de extensión, etiquetadas con el texto *<<extends>>* en el diagrama, indican una relación de dependencia en la cual se denota una especialización de uno de los casos de uso sobre el otro, ya que un caso de uso extiende a otro cuando sin alterar a éste, se incorpora su funcionalidad como parte integral del primero. A nivel de flujo de eventos, se podría decir que el flujo principal del caso base no se ve alterado, pero que en cambio, el flujo de eventos del caso extendido hace referencia al primero, de manera tal que no puede ser entendido en ausencia de los pasos del caso base (Ver Figura 1).

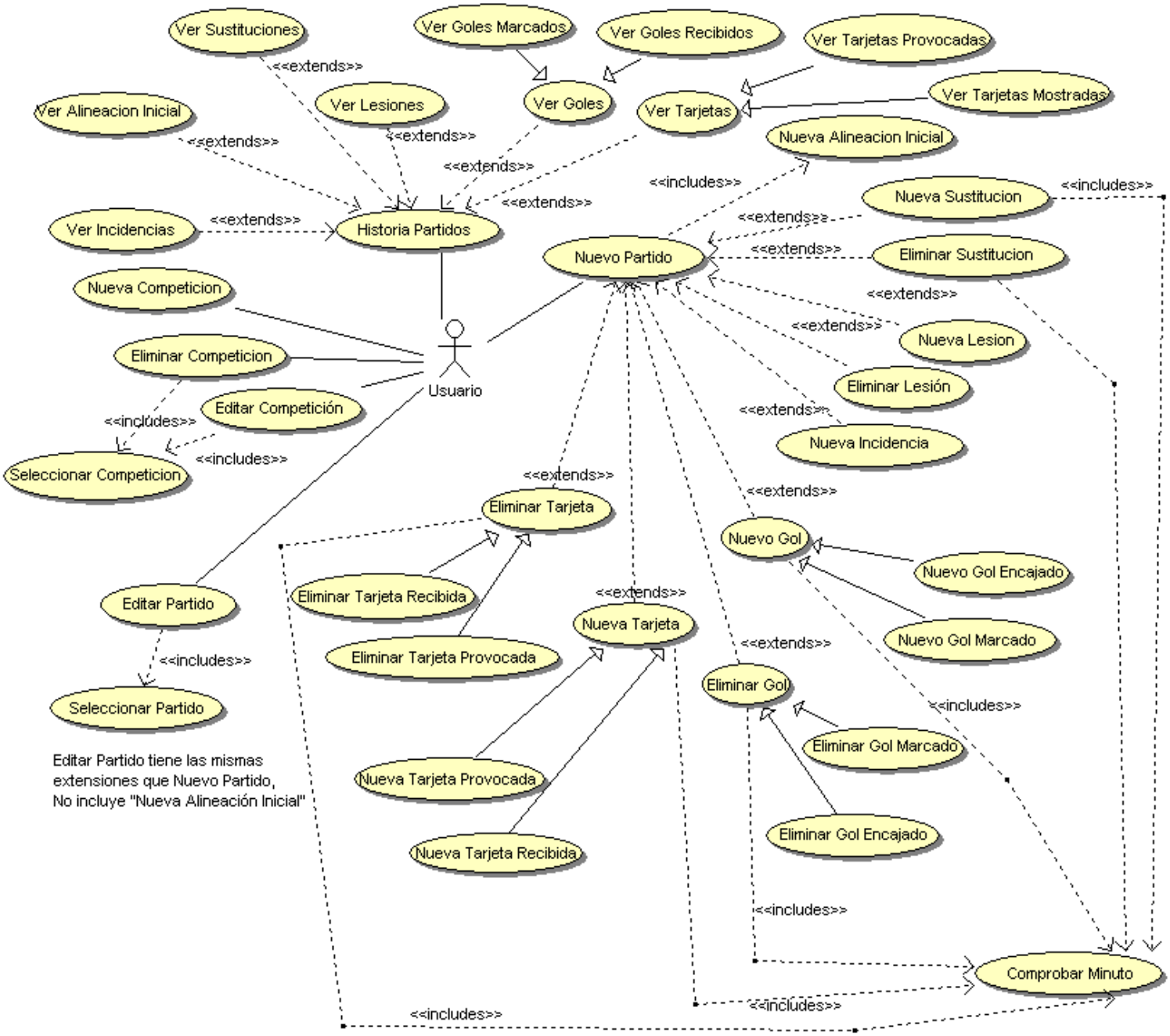


- Figura 1 -

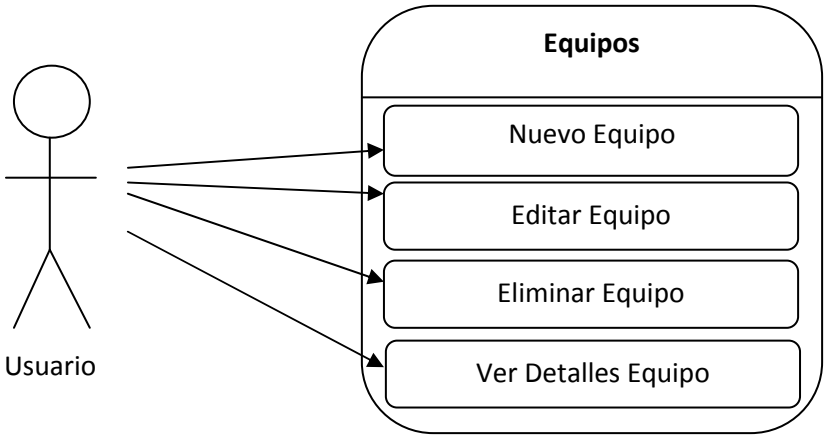
Las relaciones de inclusión, etiquetadas con el texto *<<includes>>*, indican la relación de dependencia en la cual un caso de uso es incluido dentro de otro. Un caso de uso concreto incluye a otro caso de uso, cuando como parte de su descripción breve o su flujo de eventos, se hace referencia al código del caso de uso incluído; de froma tal que lo dicho en el caso de uso incluído pasa a ser parte de la especificación del caso de uso (Figura 2).

- Figura 2 –


Seleccionar Entrenamiento
<<includes>>
Eliminar Entrenamiento
Nuevo Equipo
Editar Equipo
<<includes>>
Seleccionar Equipo
<<includes>>
Eliminar Equipo
<<includes>>
Registrar Mi Equipo
Ver Enfrentamientos Equipo
Editar Información Mi Equipo
Seleccionar Ejercicio
<<includes>>
Eliminar Ejercicio
<<includes>>
<<extends>>
<<includes>>
Editar Ejercicio
<<extends>>
Editar Entrenamiento
<<includes>>
Nueva Incidencia
<<extends>>
<<extends>>
Comentar Ejercicio
Nuevo Entrenamiento
Nuevo Entrenamiento tiene las mismas extensiones que Editar Entrenamiento
<<extends>>
Nuevo Ejercicio
Usuario
Comprobar Competicion
<<includes>>
Nuevo Título
Editar Título
<<includes>>
Seleccionar Título
Eliminar Titulo
<<includes>>
<<extends>>
Ver Goles Marcados Según Minuto
Ver Rendimiento Mi Equipo
<<extends>>
Ver Tarj. recibidas Según Minuto
<<extends>>
Ver Goles Recibidos Según Minuto
Editar Jugador
<<includes>>
Seleccionar Jugador
Nuevo Jugador
<<includes>>
Eliminar Jugador
Historia Entrenamientos
<<extends>>
<<extends>>
Ver Incidencias
Ver Comentarios
<<extends>>
Ver Ejercicios
Ver Partidos Jugados
<<extends>>
Ver Minutos Disputados
<<extends>>
Ver Detalles Jugador
<<extends>>
Ver Goles Recibidos
<<extends>>
Ver Goles Marcados
<<extends>>
Ver Lesiones
<<extends>>
Ver Tarjetas Recibidas
<<extends>>
Ver Tarjetas Provocadas


Ver Sustituciones
Ver Goles Marcados
Ver Goles Recibidos
Ver Tarjetas Provocadas
Ver Alineacion Inicial
Ver Lesiones
Ver Goles
Ver Tarjetas
Ver Tarjetas Mostradas
Nueva Alineacion Inicial
Nueva Sustitucion
Ver Incidencias
Historia Partidos
Nuevo Partido
Eliminar Sustitucion
Nueva Competicion
Nueva Lesion
Eliminar Competicion
Usuario
Eliminar Lesión
Editar Competición
Nueva Incidencia
Seleccionar Competicion
Eliminar Tarjeta
Nuevo Gol
Nuevo Gol Encajado
Editar Partido
Eliminar Tarjeta Recibida
Nueva Tarjeta
Nuevo Gol Marcado
Eliminar Tarjeta Provocada
Eliminar Gol
Seleccionar Partido
Eliminar Gol Marcado
Nueva Tarjeta Provocada
Eliminar Gol Encajado
Nueva Tarjeta Recibida
Comprobar Minuto
<<extends>>
<<includes>>
Editar Partido tiene las mismas
extensiones que Nuevo Partido,
No incluye "Nueva Alineación Inicial"

2.2.1. CASOS DE USO EQUIPOS



El usuario gestiona la información relativa a los equipos que se enfrentarán a su equipo.

Detalles del caso de uso: Nuevo Equipo

Nombre: Nuevo Equipo.

Actores: Usuario.

Precondiciones: Ninguna.

Postcondiciones: El equipo dado de alta queda registrado en el sistema y se muestra en la tabla de equipos de la interfaz.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario introduce los datos del equipo que va a ser dado de alta (nombre, dirección, población, teléfono, e-mail y equipajes)	
2.	El usuario pulsa el botón de Finalizar.	
3.		El sistema valida el formato de los datos introducidos.
4.		El sistema almacena todos los datos del nuevo equipo que ya queda registrado.
5.		El sistema muestra en la tabla de equipos el nombre del nuevo equipo.

Extensiones síncronas

#1 En 1 y 2, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de equipos y ningún cambio queda registrado en el sistema.

#2 En 3, si alguno de los datos tiene un formato incorrecto (por ejemplo Teléfono: 96355ABC5), el sistema muestra un mensaje de error para que el usuario corrija el fallo. No se modifica el estado del sistema hasta que éste no sea corregido y la acción sea finalizada.

Detalles del caso de uso: Editar Equipo

Nombre: Editar Equipo.

Actores: Usuario.

Precondiciones: El equipo a modificar debe estar dado de alta en el sistema.

Postcondiciones: Las modificaciones quedan registrado en el sistema sobrescribiendo toda la información anterior relativa al mismo equipo.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona un equipo de la tabla de equipos haciendo clic sobre la fila correspondiente.	El Sistema muestra la ventana correspondiente a la información del equipo seleccionado
2.	El usuario modifica los datos pertinentes (nombre, dirección, población, teléfono, e-mail y/o equipajes)	
3.	El usuario pulsa el botón de Finalizar.	El sistema valida el formato de los datos introducidos.
4.		El sistema almacena todos los nuevos datos del equipo.

Extensiones síncronas

#1 En 2 y 3, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de equipos y ningún cambio queda registrado en el sistema.

#2 En 4, si alguno de los datos tiene un formato incorrecto (por ejemplo Teléfono: 96355ABC5), el sistema muestra un mensaje de error para que el usuario corrija el fallo. No se modifica el estado del sistema hasta que éste no sea corregido y la acción sea finalizada.

Detalles del caso de uso: Eliminar Equipo

Nombre: Eliminar Equipo.

Actores: Usuario.

Precondiciones: El equipo a eliminar debe estar dado de alta en el sistema.

Postcondiciones: El equipo y toda la información relativa al mismo en el sistema queda eliminada del mismo

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona un equipo de la tabla de equipos haciendo clic sobre la fila correspondiente.	
2.	El usuario pulsa el botón “Eliminar Equipo”	
3.		El sistema muestra un mensaje de advertencia al usuario para que sea consciente de la operación que va a realizar y sus consecuencias en el sistema.
4.	El usuario pulsa el botón de Aceptar.	Se elimina el equipo y todos los datos relativos a él del sistema.

Extensiones síncronas

#1 En 4, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de equipos y ningún cambio queda registrado en el sistema. No se elimina el equipo previamente seleccionado.

Detalles del caso de uso: Ver Detalles Equipo

Nombre: Ver Detalles Equipo.

Actores: Usuario.

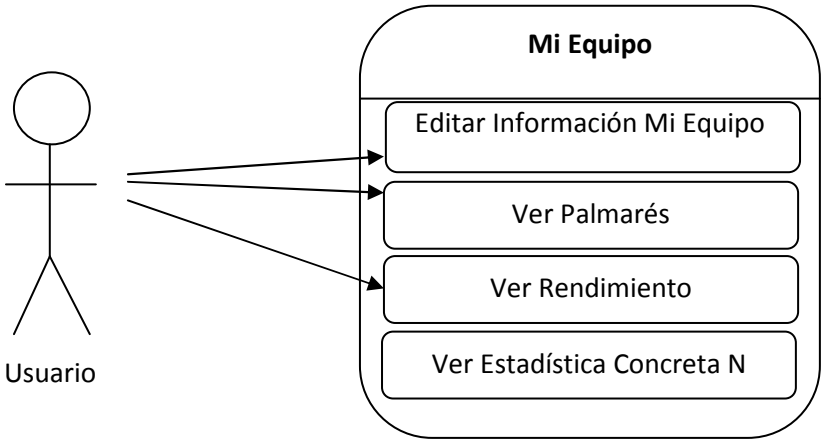
Precondiciones: El equipo debe estar dado de alta en el sistema.

Postcondiciones: Ninguna.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona un equipo de la tabla de equipos haciendo clic sobre la fila correspondiente.	
2.	El usuario pulsa el botón “Ver Detalles”	
3.		El sistema muestra el balance de partidos entre el equipo seleccionado y el equipo del usuario, y un desglose de los partidos con sus resultados

2.2.2. CASOS DE USO MI EQUIPO



El usuario gestiona toda la información relativa a su equipo.

Detalles del caso de uso: Editar Información Mi Equipo

Nombre: Editar Información Mi Equipo.

Actores: Usuario.

Precondiciones: Ninguna.

Postcondiciones: El equipo manejado por el usuario quedará registrado en el sistema con los datos que éste proporcione del mismo.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario introduce los datos del equipo.	
2.	El usuario pulsa el botón de Finalizar.	
3.		El sistema valida el formato de los datos introducidos.
4.		El sistema almacena todos los datos del equipo que ya queda registrado.

Extensiones síncronas

#1 En 1 y 2, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de mi equipo y ningún cambio queda registrado en el sistema.

#2 En 2, si el usuario no había dado de alta a su propio equipo y es la primera ver que finaliza esta acción, el equipo quedará registrado en el sistema como propio del usuario. En este caso, se da de alta al equipo del usuario.

#3 En 3, si alguno de los datos tiene un formato incorrecto (por ejemplo Teléfono: 96355ABC5), el sistema muestra un mensaje de error para que el usuario corrija el fallo. No se modifica el estado del sistema hasta que éste no sea corregido y la acción sea finalizada.

Detalles del caso de uso: Ver Palmarés

Nombre: Ver Palmarés.

Actores: Usuario.

Precondiciones: El usuario debe haber dado de alta en el sistema a su equipo.

Postcondiciones: Ninguna.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona en el menú Mi Equipo la opción Palmarés.	
2.		El sistema muestra una tabla en la cual se indican las competiciones en las cuales participó el equipo del usuario y en qué posición terminó las mismas.

Detalles del caso de uso: Ver Rendimiento Equipo

Nombre: Ver Rendimiento Equipo.

Actores: Usuario.

Precondiciones: Que el usuario haya dado de alta en el sistema a su propio equipo.

Postcondiciones: Ninguna.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona la opción "Ver Rendimiento"	
2.		El sistema muestra una tabla en la cual se indican las estadísticas elementales del equipo. (Goles a favor y en contra, tarjetas, partidos jugados, ganados, empatados y perdidos...)

Extensiones síncronas

#1 En 2, si el usuario selecciona en el listbox alguna de las estadísticas concretas disponibles, el sistema mostrará otra tabla con los resultados deseados (Caso de uso ver Estadística Concreta N).

Detalles del caso de uso: Ver Estadística Concreta 1, ..., Ver Estadística Concreta N

Nombre: Ver Estadística Concreta 1, ..., Ver Estadística Concreta N.

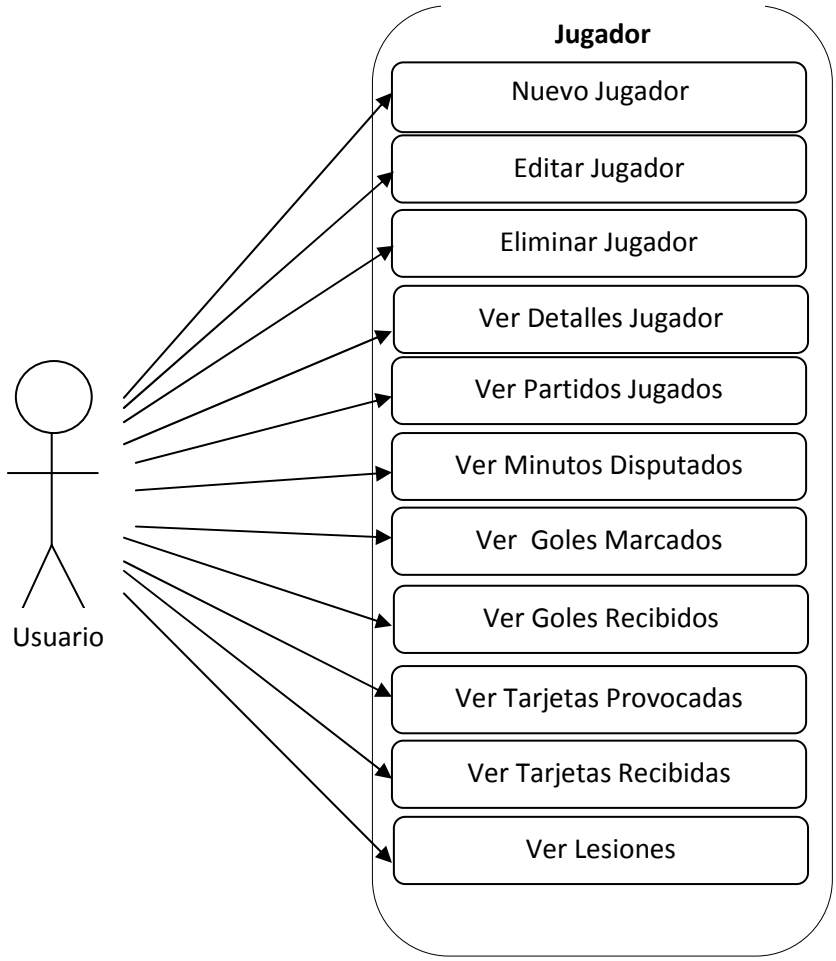
Actores: Usuario.

Precondiciones: Que el usuario haya dado de alta en el sistema a su propio equipo.

Postcondiciones: Ninguna.

Flujo de eventos		
	Intención de Usuario	**Obligaciones del Sistema**
1.	El usuario selecciona una de las estadísticas concretas del menú rendimiento en el listbox correspondiente y pulsa el botón "Ver"	
2.		El sistema muestra una tabla en la cual se indican los datos de la estadística que el usuario deseaba ver desglosados minutos y con comentarios que el usuario habría introducido.

2.2.3. CASOS DE USO JUGADORES



El usuario podrá mantener y modificar los datos referentes a los jugadores que pertenecen a su equipo.

Detalles del caso de uso: Nuevo Jugador

Nombre: Nuevo Jugador.

Actores: Usuario.

Precondiciones: El usuario debe haber dado de alta su equipo.

Postcondiciones: El nuevo jugador que ahora forma parte de la plantilla queda registrado en el sistema y se muestra en la tabla de jugadores de la interfaz junto a su dorsal y su demarcación.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario introduce los datos del jugador que va a ser dado de alta (nombre, dirección, población, teléfono, e-mail, dorsal, demarcación, código postal, año de ingreso, fecha de nacimiento y foto)	
2.	El usuario pulsa el botón de Finalizar.	
3.		El sistema valida el formato de los datos introducidos.
4.		El sistema almacena todos los datos del nuevo jugador que ya queda registrado.
5.		El sistema muestra en la tabla de jugadores el nombre del nuevo jugador, su dorsal y su demarcación.

Extensiones síncronas

#1 En 1 y 2, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de jugadores y ningún cambio queda registrado en el sistema.

#2 En 3, si alguno de los datos tiene un formato incorrecto (por ejemplo Teléfono: 96355ABC5), el sistema muestra un mensaje de error para que el usuario corrija el fallo. No se modifica el estado del sistema hasta que éste no sea corregido y la acción sea finalizada.

Detalles del caso de uso: Editar Jugador

Nombre: Editar Jugador.

Actores: Usuario.

Precondiciones: El jugador a modificar debe estar dado de alta en el sistema.

Postcondiciones: Las modificaciones quedan registradas en el sistema sobrescribiendo toda la información anterior relativa al mismo jugador.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona un jugador de la tabla de jugadores haciendo clic sobre la fila correspondiente.	El Sistema muestra la ventana correspondiente a la información vigente del jugador seleccionado.
2.	El usuario modifica los datos pertinentes (nombre, dirección, población, teléfono, e-mail, dorsal, demarcación, código postal, año de ingreso, fecha de nacimiento y/o foto)	
3.	El usuario pulsa el botón de Finalizar.	El sistema valida el formato de los datos introducidos.
4.		El sistema almacena todos los nuevos datos del equipo.

Extensiones síncronas

#1 En 2 y 3, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de jugadores y ningún cambio queda registrado en el sistema.

#2 En 4, si alguno de los datos tiene un formato incorrecto (por ejemplo Teléfono: 96355ABC5), el sistema muestra un mensaje de error para que el usuario corrija el fallo. No se modifica el estado del sistema hasta que éste no sea corregido y la acción sea finalizada.

Detalles del caso de uso: Eliminar Jugador

Nombre: Eliminar Jugador.

Actores: Usuario.

Precondiciones: El jugador a eliminar debe estar dado de alta en el sistema.

Postcondiciones: El jugador y toda la información relativa al mismo en el sistema (rendimiento individual) queda eliminada.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona un jugador de la tabla de jugadores en la interfaz haciendo clic sobre la fila correspondiente.	
2.	El usuario pulsa el botón “Eliminar Jugador”	
3.		El sistema muestra un mensaje de advertencia al usuario para que sea consciente de la operación que va a realizar y sus consecuencias en el sistema.
4.	El usuario pulsa el botón de Aceptar.	Se elimina el jugador y todos los datos relativos a él del sistema.

Extensiones síncronas

#1 En 4, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de equipos y ningún cambio queda registrado en el sistema. No se elimina el jugador seleccionado.

Detalles del caso de uso: Ver Detalles Jugador

Nombre: Ver Detalles Jugador.

Actores: Usuario.

Precondiciones: El jugador debe estar dado de alta en el sistema, y debe haber sido seleccionado haciendo clic en su fila correspondiente en la tabla de la interfaz de jugadores.

Postcondiciones: Ninguna.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona un jugador de la tabla de jugadores haciendo clic sobre la fila correspondiente.	
2.	El usuario pulsa el botón “Ver Detalles”	
3.		El sistema muestra los resultados globales de estadísticas individuales concretas (minutos jugados, partidos disputados, etc.)

Extensiones síncronas

#1 Si tras pasar por 2, el usuario pulsa alguno de los botones para desglosar alguna estadística (Ver partidos jugados, Ver Minutos Disputados, etc.), el sistema (en el que sería paso 4) mostrará otra tabla con los resultados deseados (Casos de uso a explicar a continuación. Siete opciones).

Detalles del caso de uso: Ver Partidos Jugados

Nombre: Ver Partidos Jugados.

Actores: Usuario.

Precondiciones: Que el usuario haya dado de alta en el sistema al jugador del cual se requiere esta estadística.

Postcondiciones: Ninguna.

Extiende a: Ver Detalles Jugador.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción "Ver Partidos Jugados" en la pantalla de estadísticas de un jugador.	
2.		El sistema muestra una tabla en la cual se indican los partidos disputados por ese jugador (contra qué rival) y si fue titular o suplente en el mismo.

Extensiones síncronas

#1 En 2, si el jugador seleccionado no ha disputado ningún partido, el sistema mostrará un mensaje avisando al usuario de este hecho, sin mostrar, por tanto, ninguna tabla, ya que esta sería vacía y es innecesaria.

Detalles del caso de uso: Ver Minutos Disputados

Nombre: Ver Minutos Disputados.

Actores: Usuario.

Precondiciones: Que el usuario haya dado de alta en el sistema al jugador del cual se requiere esta estadística.

Postcondiciones: Ninguna.

Extiende a: Ver Detalles Jugador.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción "Ver Minutos Disputados" en la pantalla de estadísticas de un jugador.	
2.		El sistema muestra una tabla en la cual se indican sólo los partidos disputados por ese jugador (contra qué rival) y los minutos que jugó del mismo, cuya suma se corresponderá con los minutos totales mostrados en la pantalla de estadísticas de un jugador.

Extensiones síncronas

#1 En 2, si el jugador seleccionado no ha disputado ningún minuto, el sistema mostrará un mensaje avisando al usuario de este hecho, sin mostrar, por tanto, ninguna tabla, ya que esta sería vacía y es innecesaria.

Detalles del caso de uso: Ver Goles Marcados

Nombre: Ver Goles Marcados.

Actores: Usuario.

Precondiciones: Que el usuario haya dado de alta en el sistema al jugador del cual se requiere esta estadística.

Postcondiciones: Ninguna.

Extiende a: Ver Detalles Jugador.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción “Ver Goles Marcados” en la pantalla de estadísticas de un jugador.	
2.		El sistema muestra una tabla en la cual se indican los partidos en los cuales el jugador seleccionado anotó un tanto o más, mostrando el rival, el minuto de juego en el que el tanto fue anotado, y cuál era el resultado provisional en aquel momento.

Extensiones síncronas

#1 En 2, si el jugador seleccionado no ha anotado ningún gol, el sistema mostrará un mensaje avisando al usuario de este hecho, sin mostrar, por tanto, ninguna tabla, ya que esta sería vacía y es innecesaria.

Detalles del caso de uso: Ver Goles Recibidos

Nombre: Ver Goles Recibidos.

Actores: Usuario.

Precondiciones: Que el usuario haya dado de alta en el sistema al jugador del cual se requiere esta estadística.

Postcondiciones: Ninguna.

Extiende a: Ver Detalles Jugador.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción “Ver Goles Marcados” en la pantalla de estadísticas de un jugador.	
2.		El sistema muestra una tabla en la cual se indican los partidos en los cuales el jugador seleccionado estaba en el terreno de juego cuando el equipo encajó un gol, mostrando el rival, el minuto de juego en el que el tanto fue encajado, y cuál era el resultado provisional en aquel momento.

Extensiones síncronas

#1 En 2, si el jugador seleccionado no ha estado presente en ningún momento en el que el equipo haya encajado goles, el sistema mostrará un mensaje avisando al usuario de este hecho, sin mostrar, por tanto, ninguna tabla, ya que esta sería vacía y es innecesaria.

Detalles del caso de uso: Ver Tarjetas Provocadas

Nombre: Ver Tarjetas Provocadas.

Actores: Usuario.

Precondiciones: Que el usuario haya dado de alta en el sistema al jugador del cual se requiere esta estadística.

Postcondiciones: Ninguna.

Extiende a: Ver Detalles Jugador.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción "Ver Tarjetas Provocadas" en la pantalla de estadísticas de un jugador.	
2.		El sistema muestra una tabla en la cual se indican los partidos en los cuales el jugador provocó una tarjeta a un contrario, así como el minuto de la sanción, el color de la tarjeta y un breve comentario incluido por el usuario cuando la tarjeta se almacenó en el sistema.

Extensiones síncronas

#1 En 2, si el jugador seleccionado no ha provocado ninguna amonestación a un contrario, el sistema mostrará un mensaje avisando al usuario de este hecho, sin mostrar, por tanto, ninguna tabla, ya que esta sería vacía y es innecesaria.

Detalles del caso de uso: Ver Tarjetas Recibidas

Nombre: Ver Tarjetas Recibidas.

Actores: Usuario.

Precondiciones: Que el usuario haya dado de alta en el sistema al jugador del cual se requiere esta estadística.

Postcondiciones: Ninguna.

Extiende a: Ver Detalles Jugador.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción "Ver Tarjetas Recibidas".	
2.		El sistema muestra los partidos en los cuales el jugador fue amonestado, el minuto, el color de la tarjeta y un breve comentario acerca de la sanción.

Extensiones síncronas

#1 En 2, si el jugador seleccionado no ha sido ni amonestado ni expulsado, el sistema mostrará un mensaje avisando al usuario de este hecho, sin mostrar, por tanto, ninguna tabla, ya que esta sería vacía y es innecesaria.

Detalles del caso de uso: Ver Lesiones

Nombre: Ver Lesiones.

Actores: Usuario.

Precondiciones: Que el usuario haya dado de alta en el sistema al jugador del cual se requiere esta estadística.

Postcondiciones: Ninguna.

Extiende a: Ver Detalles Jugador.

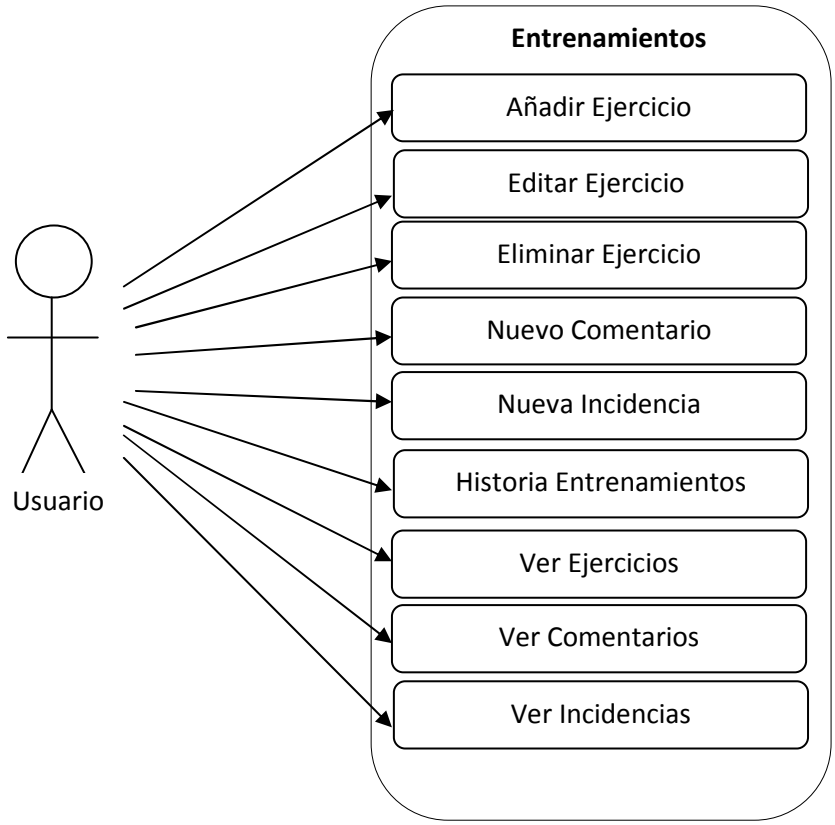
Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción "Ver Lesiones" en la pantalla de estadísticas de un jugador.	
2.		El sistema muestra una tabla en la cual se indica la fecha de cada lesión sufrida por el jugador seleccionado, el tiempo de recuperación y una descripción de la lesión sufrida en cada caso.

Extensiones síncronas

#1 En 2, si el jugador seleccionado no ha sufrido ninguna lesión, el sistema mostrará un mensaje avisando al usuario de este hecho, sin mostrar, por tanto, ninguna tabla, ya que esta sería vacía y es innecesaria.

2.2.4. CASOS DE USO ENTRENAMIENTOS



El usuario podrá realizar con los siguientes requisitos funcionales la gestión completa de los entrenamientos y sus ejercicios.

Detalles del caso de uso: Añadir Ejercicio

Nombre: Añadir Ejercicio.

Actores: Usuario.

Precondiciones: El usuario debe haber entrado en el menú de nuevo entrenamiento para añadir el ejercicio al entrenamiento de un día concreto.

Postcondiciones: El ejercicio añadido queda registrado en el sistema asociado a un entrenamiento en concreto. Se muestra el ejercicio y su duración en la tabla de ejercicios en la interfaz de entrenamientos.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario introduce los datos del ejercicio (descripción y duración del mismo)	
2.	El usuario pulsa el botón de Finalizar.	
3.		El sistema valida el formato de los datos introducidos.
4.		El sistema almacena todos los datos del ejercicio que ya queda registrado.
5.		El sistema muestra en la tabla de entrenamiento la descripción del ejercicio y la duración del mismo que se introdujo en 1

Extensiones síncronas

#1 En 1 y 2, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de nuevo entrenamiento y ningún cambio queda registrado en el sistema.

#2 En 3, si alguno de los datos tiene un formato incorrecto (por ejemplo Duración: 3n minutos), el sistema muestra un mensaje de error para que el usuario corrija el fallo. No se modifica el estado del sistema hasta que éste no sea corregido y la acción sea finalizada.

Detalles del caso de uso: Editar Ejercicio

Nombre: Editar Ejercicio.

Actores: Usuario.

Precondiciones: El ejercicio a modificar debe estar dado de alta en el sistema.

Postcondiciones: Las modificaciones quedan registradas en el sistema sobrescribiendo toda la información anterior relativa al mismo jugador.

Extiende a: Nuevo Entrenamiento.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona un ejercicio de la tabla del entrenamiento haciendo clic sobre la fila correspondiente y pulsa "Editar ejercicio".	El Sistema muestra la ventana correspondiente a la información vigente del ejercicio seleccionado.
2.	El usuario modifica los datos pertinentes (descripción y/o duración)	
3.	El usuario pulsa el botón de Finalizar.	El sistema valida el formato de los datos introducidos.
4.		El sistema almacena el ejercicio actualizado.

Extensiones síncronas

#1 En 2 y 3, si el usuario pulsa el botón "Cancelar", se vuelve a la interfaz de jugadores y ningún cambio queda registrado en el sistema.

#2 En 4, si alguno de los datos tiene un formato incorrecto (por ejemplo Duración: 3n minutos), el sistema muestra un mensaje de error para que el usuario corrija el fallo. No se modifica el estado del sistema hasta que éste no sea corregido y la acción sea finalizada.

Detalles del caso de uso: Eliminar Ejercicio

Nombre: Eliminar Ejercicio.

Actores: Usuario.

Precondiciones: El ejercicio a eliminar debe estar dado de alta en el sistema.

Postcondiciones: El ejercicio queda eliminado del entrenamiento y del sistema.

Extiende a: Nuevo Entrenamiento.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona un ejercicio de la tabla de ejercicios en la interfaz de nuevo entrenamiento haciendo clic sobre la fila correspondiente.	
2.	El usuario pulsa el botón “Eliminar ejercicio”	
3.		El sistema muestra un mensaje de advertencia al usuario para que sea consciente de la operación que va a realizar y sus consecuencias en el sistema.
4.	El usuario pulsa el botón de Aceptar.	Se elimina el ejercicio.

Extensiones síncronas

#1 En 4, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de nuevo entrenamiento y ningún cambio queda registrado en el sistema. No se elimina el ejercicio.

Detalles del caso de uso: Nueva Incidencia

Nombre: Nueva Incidencia.

Actores: Usuario.

Precondiciones: El ejercicio sobre el cual se va a anotar una incidencia debe estar dado de alta en el sistema y seleccionado antes de iniciar este caso de uso.

Postcondiciones: La incidencia añadida queda registrada en el sistema asociada un ejercicio concreto de un entrenamiento concreto.

Extiende a: Nuevo Entrenamiento.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona un ejercicio de la tabla del entrenamiento haciendo clic sobre la fila correspondiente y pulsa “Incidencias”.	El Sistema muestra la ventana correspondiente a las incidencias del ejercicio seleccionado (estará vacío el cuadro de incidencias)
2.	El usuario introduce en el cuadro de texto la incidencia a registrar.	
3.	El usuario pulsa el botón de Finalizar.	
4.		El sistema almacena la incidencia asociada al ejercicio.

Extensiones síncronas

#1 En 2 y 3, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de nuevo entrenamiento y ningún cambio queda registrado en el sistema.

Detalles del caso de uso: Nuevo Comentario

Nombre: Nuevo Comentario.

Actores: Usuario.

Precondiciones: El ejercicio sobre el cual se va a anotar un comentario debe estar dado de alta en el sistema y seleccionado antes de iniciar este caso de uso.

Postcondiciones: El nuevo comentario queda registrado en el sistema asociado un ejercicio concreto de un entrenamiento concreto.

Extiende a: Nuevo Entrenamiento.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona un ejercicio de la tabla del entrenamiento haciendo clic sobre la fila correspondiente y pulsa “Comentar Ejercicio”.	El Sistema muestra la ventana correspondiente a los comentarios del ejercicio seleccionado (estará vacío el cuadro de comentarios)
2.	El usuario introduce en el cuadro de texto comentario a registrar.	
3.	El usuario pulsa el botón de Finalizar.	
4.		El sistema almacena el comentario asociado al ejercicio.

Extensiones síncronas

#1 En 2 y 3, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de nuevo entrenamiento y ningún cambio queda registrado en el sistema.

Detalles del caso de uso: Historia Entrenamientos

Nombre: Historia Entrenamientos.

Actores: Usuario.

Precondiciones: Ninguna.

Postcondiciones: Ninguna.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción “Ver Histórico” en la pantalla de entrenamientos.	
2.		El sistema muestra una nueva ventana en la que poder seleccionar un entrenamiento (por fecha) para ver sus detalles (comentarios, ejercicios y/o incidencias acontecidas).

Extensiones síncronas

#1 En 2, si se selecciona una fecha en la cual no ha habido entrenamiento o bien porque no hay aún ningún entrenamiento registrado en el sistema, se mostrará un mensaje de advertencia al usuario para evitar que éste pierda el tiempo buscando datos inexistentes.

Detalles del caso de uso: Ver Comentarios

Nombre: Ver Comentarios.

Actores: Usuario.

Precondiciones: Que el usuario haya seleccionado la fecha correspondiente a un entrenamiento existente.

Postcondiciones: Ninguna.

Extiende a: Historia Entrenamiento.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción "Ver Comentarios" en la pantalla del entrenamiento seleccionado dentro de la historia de los entrenamientos.	
2.		El sistema muestra en el cuadro de texto correspondiente los comentarios que fueron introducidos en la aplicación en ese entrenamiento.

Extensiones síncronas

#1 En 2, si no hay comentarios que mostrar, el sistema mostrará en el cuadro de texto destinado a los comentarios un aviso de que éstos no existen.

Detalles del caso de uso: Ver Incidencias

Nombre: Ver Incidencias.

Actores: Usuario.

Precondiciones: Que el usuario haya seleccionado la fecha correspondiente a un entrenamiento existente.

Postcondiciones: Ninguna.

Extiende a: Historia Entrenamiento.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción "Ver Incidencias" en la pantalla del entrenamiento seleccionado dentro de la historia de los entrenamientos.	
2.		El sistema muestra en el cuadro de texto correspondiente las incidencias que fueron introducidos en la aplicación en ese entrenamiento.

Extensiones síncronas

#1 En 2, si no hay incidencias que mostrar, el sistema mostrará en el cuadro de texto destinado a ellas un aviso de que indique que éstas no existen para el entrenamiento seleccionado.

Detalles del caso de uso: Ver Ejercicios

Nombre: Ver Ejercicios.

Actores: Usuario.

Precondiciones: Que el usuario haya seleccionado la fecha correspondiente a un entrenamiento existente.

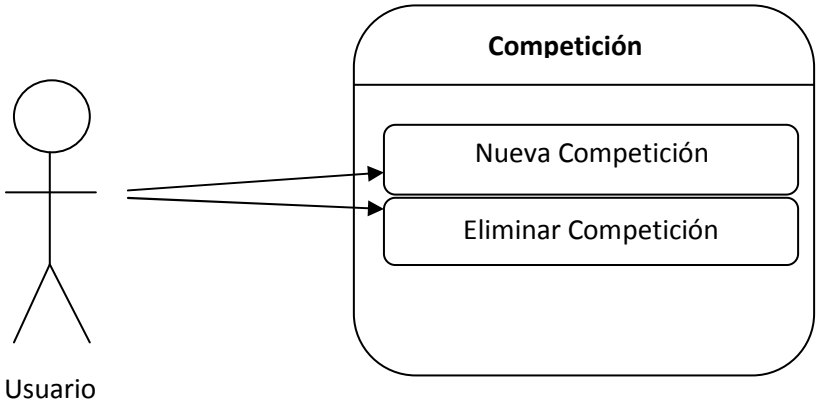
Postcondiciones: Ninguna.

Extiende a: Historia Entrenamiento.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción "Ver Ejercicios" en la pantalla del entrenamiento seleccionado dentro de la historia de los entrenamientos.	
2.		El sistema muestra en el cuadro de texto correspondiente los ejercicios que fueron realizados en ese entrenamiento.

2.2.5. CASOS DE USO COMPETICIÓN



El fin de estos casos de uso es proporcionar al usuario la posibilidad de añadir y eliminar competiciones en las cuales su equipo va a participar y en las cuales va a hacer un seguimiento del mismo.

Detalles del caso de uso: Nueva Competición

Nombre: Nueva Competición.

Actores: Usuario.

Precondiciones: El usuario debe haber dado de alta su equipo.

Postcondiciones: La competición queda registrada almacenada en el sistema.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario introduce los datos de la competición que va a ser dado de alta (nombre, temporada y normativas reseñables)	
2.	El usuario pulsa el botón de Finalizar.	
3.		El sistema valida el formato de los datos introducidos.
4.		El sistema almacena todos los datos de la nueva competición.
5.		El sistema muestra en la tabla de competiciones el nombre y la temporada de la nueva competición.

Extensiones síncronas

#1 En 1 y 2, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de competición y ningún cambio queda registrado en el sistema.

Detalles del caso de uso: Eliminar Competición

Nombre: Eliminar Competición.

Actores: Usuario.

Precondiciones: La competición a eliminar debe haber sido dada de alta previamente en el sistema.

Postcondiciones: La competición y toda la información relativa a ella (partidos, etc.) queda eliminada.

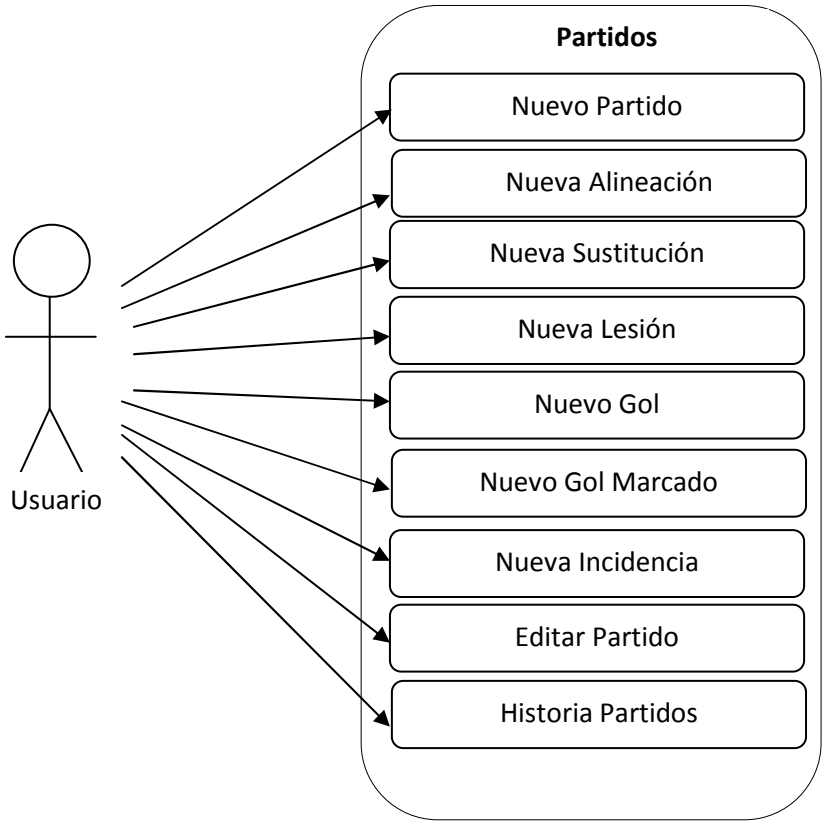
Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona una competición de la tabla de competiciones en la interfaz haciendo clic sobre la fila correspondiente.	
2.	El usuario pulsa el botón “Eliminar Competición”	
3.		El sistema muestra un mensaje de advertencia al usuario para que sea consciente de la operación que va a realizar y sus consecuencias en el sistema.
4.	El usuario pulsa el botón de Aceptar.	Se elimina la competición y todos los datos relativos a ella.

Extensiones síncronas

#1 En 4, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de competiciones y ningún cambio queda registrado en el sistema. No se elimina la competición seleccionada.

2.2.6. CASOS DE USO PARTIDOS



El usuario es capaz de gestionar la información de cada partido en el que su equipo participe, anotando incidencias y recogiendo datos que, a su vez, servirán también de cara al análisis del rendimiento de los jugadores individualmente, así como al rendimiento general del propio equipo.

Detalles del caso de uso: Nuevo Partido

Nombre: Nuevo Partido.

Actores: Usuario.

Precondiciones: Para crear un nuevo partido, el usuario debe haber almacenado previamente su equipo en el sistema.

Postcondiciones: El nuevo partido queda registrado en el sistema y las modificaciones que sobre él se hagan se irán registrando progresivamente.

Incluye a: Nueva Alineación.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario introduce los datos del partido (rival, fecha, local/visitante,)	
2.	El usuario va registrando las incidencias del partido.	
3.	El usuario pulsa el botón de Finalizar.	
4.		El sistema almacena todos los datos del partido que ya queda registrado.

Extensiones síncronas

#1 En 1, 2 o 3, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de partidos y ningún cambio queda registrado en el sistema.

Nótese que los siguientes casos de uso (desde Nueva Alineación hasta Nueva Incidencia) tendrán tablas de comportamiento idénticas para los casos de uso derivados de la “Editar Partido”, tal como se puede ver en el diagrama de 2.2. Se omite, por tanto, la inclusión de estas tablas en el documento.

Detalles del caso de uso: Nueva Alineación

Nombre: Nueva Alineación.

Actores: Usuario.

Precondiciones: Haber dado de alta un partido nuevo y no haberlo finalizado ni cancelado.

Postcondiciones: Las modificaciones quedan registradas en el sistema quedando asociada la nueva alineación inicial con el partido a la cual ésta corresponde.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona un jugador del cuadro de la plantilla.	
2.	Pulsa el botón “>>” para pasar el jugador a la lista de jugadores que conformarán el equipo inicial.	
3.	El usuario pulsa el botón de Finalizar.	
4.		El sistema almacena la alineación inicial asociada al partido.

Extensiones síncronas

#1 1 y 2 deben repetirse tantas veces como jugadores vaya a haber en la alineación inicial (5 si es fútbol sala, 7 si es fútbol 7, 11 si es fútbol “convencional”...).

#2 En 3, si el usuario pulsa “Cancelar”, ningún cambio es registrado en la alineación y por tanto esta no será dada de alta, se vuelve al menú del partido sin que éste tenga aún definida una alineación inicial para el equipo del usuario.

Detalles del caso de uso: Nueva Sustitución

Nombre: Nueva Sustitución.

Actores: Usuario.

Precondiciones: Haber dado de alta un partido nuevo y no haberlo finalizado ni cancelado, habiendo introducido en él una alineación inicial.

Postcondiciones: La sustitución queda registrada.

Extiende a: Nuevo Partido.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario rellena los datos correspondientes (jugador que entra, jugador que se retira y minuto)	
2.	El usuario pulsa el botón “Finalizar”	
3.		El sistema valida el formato de los parámetros de la sustitución introducidos.
4.		La sustitución queda dada de alta.

Extensiones síncronas

#1 En 2, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de nuevo partido y ningún cambio queda registrado en el sistema. No se almacena ninguna sustitución.

#2 En 3, si el formato de los datos introducidos no es válido (Minuto: 5v), se muestra un mensaje de error al usuario para que corrija la errata, y no se almacena la sustitución.

Detalles del caso de uso: Nueva Lesión

Nombre: Nueva Lesión.

Actores: Usuario.

Precondiciones: Haber dado de alta un partido nuevo y no haberlo finalizado ni cancelado, habiendo introducido en él una alineación inicial.

Postcondiciones: Se almacena la nueva lesión.

Extiende a: Nuevo Partido.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario rellena los datos correspondientes jugador lesionado, minuto y comentario)	
2.	El usuario pulsa el botón “Finalizar”	
3.		El sistema valida el formato de los parámetros de la lesión introducidos.
4.		La lesión queda dada de alta.

Extensiones síncronas

#1 En 2, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de nuevo partido y ningún cambio queda registrado en el sistema. No se almacena ninguna lesión.

#2 En 3, si el formato de los datos introducidos no es válido (Minuto: 5v), se muestra un mensaje de error al usuario para que corrija la errata, y no se almacena la lesión hasta que ésta no sea corregida y finalizada.

Las tablas correspondientes a Nuevo Gol y Nuevo Gol Marcado se pueden hacer extensibles a Nuevo Gol encajado con las lógicas modificaciones de significado. Las tablas de Nueva Tarjeta, Nueva Tarjeta Recibida y nueva Tarjeta Provocada son idénticas salo por el matiz de significado del caso de uso, ya que no es lo mismo un gol que una tarjeta, pero el sistema y el usuario proceden de modo similar para arrancar cada uno de estos casos de uso. Mostramos pues, a continuación las tablas de Nuevo Gol y Nuevo Gol Marcado para tener una referencia clara del comportamiento de la aplicación en cada uno de los casos mencionados anteriormente.

Detalles del caso de uso: Nuevo Gol

Nombre: Nuevo Gol.

Actores: Usuario.

Precondiciones: Haber dado de alta un partido nuevo y no haberlo finalizado ni cancelado, habiendo introducido en él una alineación inicial.

Postcondiciones: Se almacena el nuevo gol asociado al partido.

Extiende a: Nuevo Partido.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario pulsa el botón “Goles”.	
2.		El sistema muestra la opción de anotar un gol a favor o un gol en contra del equipo del usuario.

Detalles del caso de uso: Nuevo Gol Marcado

Nombre: Nuevo Gol Marcado.

Actores: Usuario.

Precondiciones: Haber dado de alta un partido nuevo y no haberlo finalizado ni cancelado, habiendo introducido en él una alineación inicial.

Postcondiciones: Se almacena el nuevo gol anotado por el equipo del usuario, teniendo repercusión en las estadísticas del equipo, así como en las de rendimiento individual del jugador que anotó el tanto.

Hereda de: Nuevo Gol.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario pulsa el botón “Goles”. (HEREDADO)	
2.		El sistema muestra la opción de anotar un gol a favor o un gol en contra del equipo del usuario. (HEREDADO)
3.	El usuario selecciona “Gol a favor”	El sistema le muestra la pantalla para registrar el gol anotado por su equipo.
4.	El usuario anota los datos correspondientes al gol (autor del gol, minuto, resultado provisional, comentarios)	
5.	El usuario pulsa el botón “Finalizar”.	
6.		El sistema valida el formato de los datos introducidos por el usuario.
7.		El sistema almacena el nuevo gol anotado.

Extensiones síncronas

#1 En 5, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de nuevo partido y ningún cambio queda registrado en el sistema. No se almacena ningún gol a favor.

#2 En 3, si el formato de los datos introducidos no es válido (Minuto: 5v), se muestra un mensaje de error al usuario para que corrija la errata, y no se almacena el tanto anotado hasta que ésta no sea corregida y finalizada.

Detalles del caso de uso: Nueva Incidencia

Nombre: Nueva Incidencia.

Actores: Usuario.

Precondiciones: Haber dado de alta un partido nuevo y no haberlo finalizado ni cancelado.

Postcondiciones: La incidencia asociada al partido queda registrada.

Extiende a: Nuevo Partido.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario rellena los datos correspondientes a la incidencia.	
2.	El usuario pulsa el botón “Finalizar”	
3.		El sistema valida el formato de los parámetros de la incidencia.
4.		La incidencia queda dada de alta.

Extensiones síncronas

#1 En 2, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de nuevo partido y ningún cambio queda registrado en el sistema. No se almacena ninguna incidencia.

#2 En 3, si el formato de los datos introducidos no es válido (Minuto: 5v), se muestra un mensaje de error al usuario para que corrija la errata, y no se almacena la incidencia.

Detalles del caso de uso: Editar Partido

Nombre: Editar Partido.

Actores: Usuario.

Precondiciones: El partido debe haber sido creado previamente.

Postcondiciones: La información del partido se verá modificada sobrescribiéndose la información que el usuario introduzca en la edición y quedando estos datos vigentes para el partido.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario introduce los nuevos datos del partido (rival, fecha, local/visitante,) si lo considera oportuno.	
2.	El usuario modifica las incidencias del partido que desee actualizar.	
3.	El usuario pulsa el botón de Finalizar.	
4.		El sistema almacena todos los datos del partido que ya quedan registrados y en vigor, siendo eliminados los datos anteriores a la actualización.

Extensiones síncronas

#1 En 1, 2 o 3, si el usuario pulsa el botón “Cancelar”, se vuelve a la interfaz de partidos y ningún cambio queda registrado en el sistema.

Detalles del caso de uso: Historia Partidos

Nombre: Historia Partidos.

Actores: Usuario.

Precondiciones: Ninguna.

Postcondiciones: Ninguna.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario hace clic en la opción “Ver Histórico” en la pantalla de partidos.	
2.		El sistema muestra una nueva ventana en la que poder seleccionar un partido (por fecha y/o rival) para ver sus detalles (alineación inicial, sustituciones, tarjetas, goles, incidencias y/o lesiones).

Extensiones síncronas

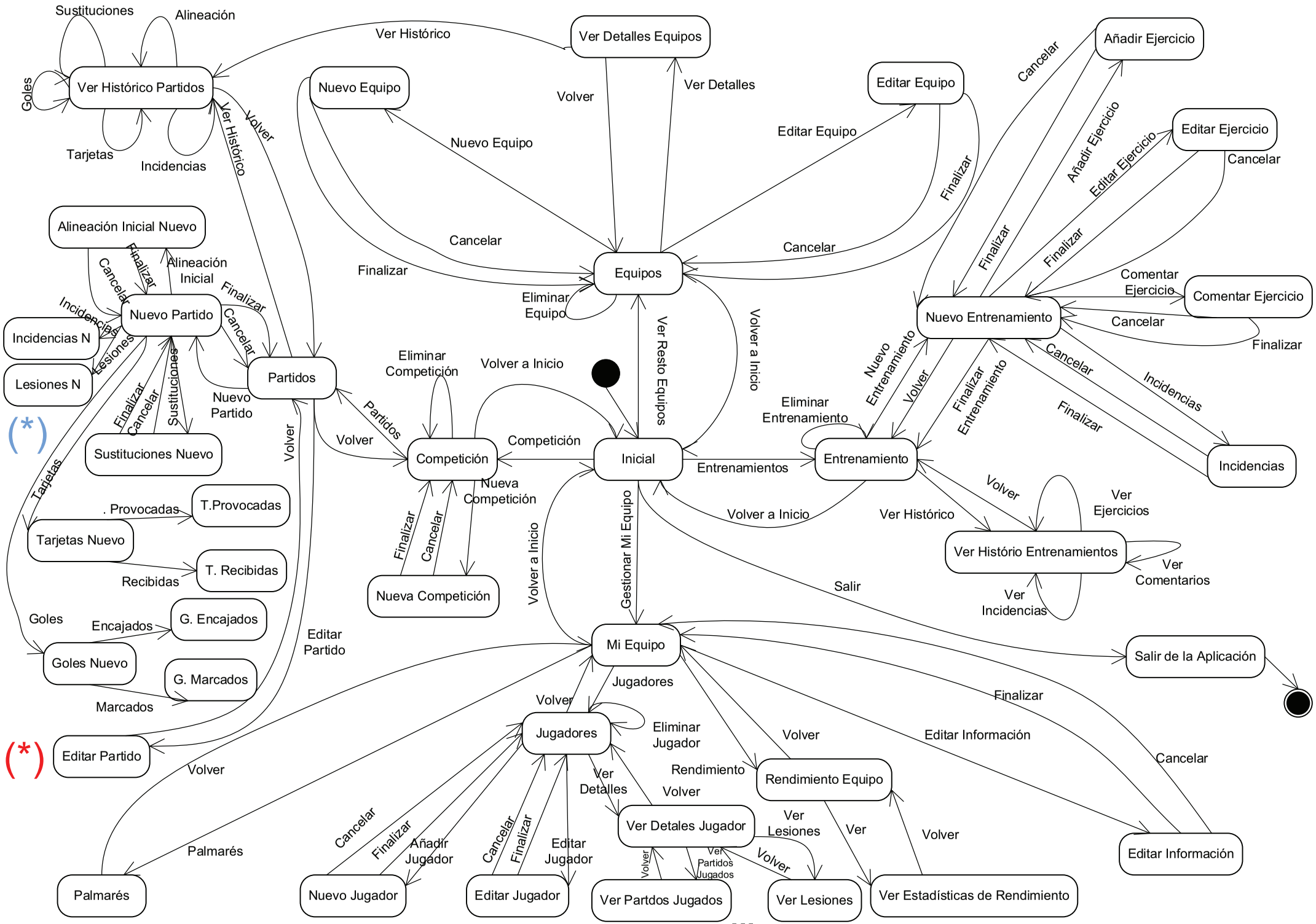
#1 En 2, si se selecciona una fecha en la cual no ha habido partido, o se selecciona un rival contra el que no se ha jugado todavía, se mostrará un mensaje de advertencia al usuario para evitar que éste pierda el tiempo buscando datos inexistentes.

El caso de uso “Historia Partidos” tiene como casos de uso que lo extienden a “Ver Alineación Inicial”, “Ver Sustituciones”, “Ver Goles”, “Ver Tarjetas”, “Ver Lesiones” y “Ver Incidencias”, cuyas tablas e descripción se omiten por su similitud a las tabla de los casos de uso “Ver X” que extienden a “Historia Entrenamientos” en el punto 2.2.4. del presente documento.

2.3. MÁQUINA DE ESTADOS

Tras esta descripción detallada de todas las pantallas que compondrán la aplicación con sus correspondientes entradas y salidas, procederemos a continuación a seguir con el diseño de la misma siendo ésta modelizada como la máquina de estados que se muestra en su totalidad en la página siguiente.

Nótese que el estado “Editar Partidos” tiene transiciones análogas a “Nuevo Partido”, es decir, Alineación Editar, Goles Editar... con transiciones entre éstos y “Editar Partido” de cancelar y finalizar. Se omiten en la máquina de estado representada por falta de espacio. Ver (*) en la página siguiente. Desde los estados Tarjetas, Goles, Incidencias y Lesiones debe haber dos transiciones desde cada uno de ellos hacia nuevo partido que responderán a los eventos “Cancelar” y “Finalizar”, omitidos también por falta de espacio . Desde “Tarjetas Nuevo” y “Goles Nuevo” se omiten sendas transiciones hacia “Nuevo Partido” con identificación “Volver” (Ver (*)).


Sustituciones
Alineación
Goles
Ver Histórico Partidos
Tarjetas
Incidencias
Ver Histórico
Volver
Ver Detalles Equipos
Volver
Ver Detalles
Nuevo Equipo
Nuevo Equipo
Editar Equipo
Editar Equipo
Finalizar
Cancelar
Cancelar
Finalizar
Equipos
Eliminar Equipo
Añadir Ejercicio
Cancelar
Añadir Ejercicio
Editar Ejercicio
Editar Ejercicio
Cancelar
Finalizar
Finalizar
Comentar Ejercicio
Comentar Ejercicio
Cancelar
Finalizar
Nuevo Entrenamiento
Alineación Inicial Nuevo
Finalizar
Cancelar
Alineación Inicial
Finalizar
Incidencias
Lesiones
Nuevo Partido
Cancelar
Incidencias N
Lesiones N
Partidos
Nuevo Partido
Finalizar
Cancelar
Sustituciones
Sustituciones Nuevo
(*)
Tarjetas
Volver
Partidos
Volver
Eliminar Competición
Volver a Inicio
Competición
Competición
Nueva Competición
Finalizar
Cancelar
Nueva Competición
Ver Resto Equipos
Volver a Inicio
Inicial
Eliminar Entrenamiento
Nuevo Entrenamiento
Volver
Finalizar Entrenamiento
Cancelar
Incidencias
Finalizar
Entrenamiento
Entrenamientos
Incidencias
Volver a Inicio
Volver a Inicio
Gestionar Mi Equipo
Volver
Ver Ejercicios
Ver Histórico
Ver Histório Entrenamientos
Ver Comentarios
Ver Incidencias
Salir
. Provocadas
T.Provocadas
Tarjetas Nuevo
Recibidas
T. Recibidas
Goles
Encajados
G. Encajados
Goles Nuevo
G. Marcados
Marcados
Editar Partido
(*)
Editar Partido
Volver
Mi Equipo
Salir de la Aplicación
Jugadores
Volver
Eliminar Jugador
Volver
Editar Información
Finalizar
Cancelar
Rendimiento
Rendimiento Equipo
Ver Detalles
Volver
Ver Detalles Jugador
Ver Lesiones
Ver
Volver
Editar Información
Palmarés
Palmarés
Cancelar
Finalizar
Añadir Jugador
Nuevo Jugador
Cancelar
Finalizar
Editar Jugador
Editar Jugador
Volver
Ver Partidos Jugados
Ver Partdos Jugados
Volver
Ver Lesiones
Ver Estadísticas de Rendimiento
...

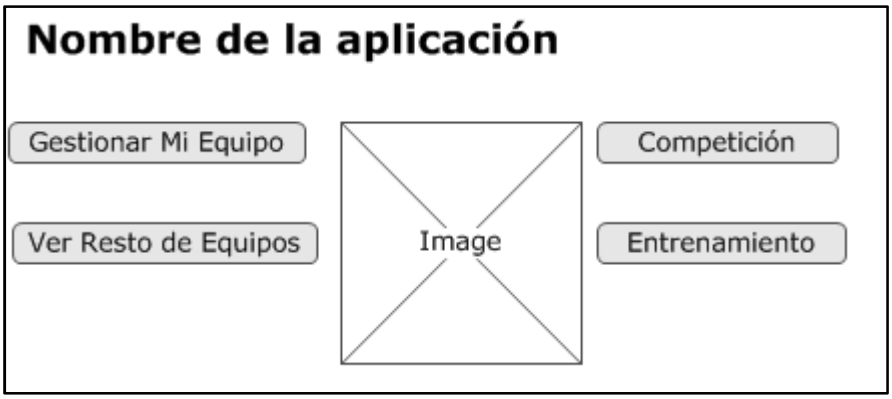
2.4. PANTALLAS DE LA APLICACIÓN

En lo sucesivo, y a partir del estudio previo elaborado, se procederá a enumerar las distintas pantallas que, a priori, constituirán la misma, siendo éstas susceptibles de ser modificadas a lo largo del proceso de implementación, y para las cuales nos basamos en cada uno de los estados que componen la máquina de estados del punto 2.3.

Destacar que se ha realizado un proceso iterativo e incremental en el diseño de la aplicación para llegar a este punto, tal y como se puede seguir desde el principio de este documento.

Asimismo, se hace notar al lector que lo que en lo sucesivo se expone no es más que una idea inicial de cómo serán el aspecto visual de la aplicación para poder hacernos una idea lo suficientemente clara de cómo será este cuando esté implementado y testeado, y por tanto, como prueba evolutiva del crecimiento del proyecto y de su toma de forma, en las sucesivas fases, se mostrarán, sin ser explicadas para evitar redundancias con las explicaciones de las pantallas en este punto, las ventanas y el aspecto real que ha cobrado la aplicación, tanto en la implementación con las librerías Qt como una vez migrada la aplicación al dispositivo Maemo.

2.4.1 INICIAL

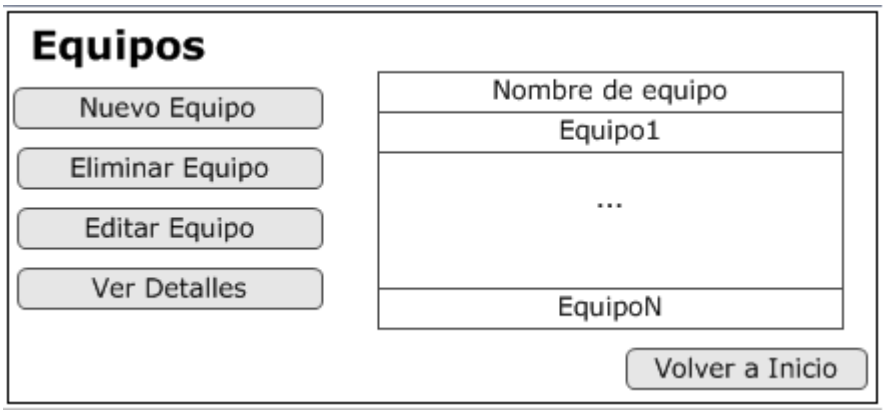


La pantalla inicial será la primera que el usuario verá al entrar a la aplicación.

Entradas: ninguna.

Salidas: Mi Equipo, Equipos, Competición y Entrenamiento.

2.4.2. EQUIPOS



Desde esta ventana se podrá gestionar el elenco de equipos al que su propio equipo se ha enfrentado o se enfrentará en partidos venideros.

Entradas: Inicial, Nuevo Equipo, Editar Equipo, Ver detalles.

Salidas: Inicial, Nuevo Equipo, Editar Equipo, Ver detalles.

Destacar que la eliminación de equipos se llevará a cabo mediante la selección de uno de los equipos de la tabla, y al pulsar el botón “Eliminar equipo” éste desaparecerá de la misma y será eliminado del sistema.

2.4.3. NUEVO EQUIPO

Desde esta ventana se añadirán al sistema los equipos que se medirán al equipo del usuario.

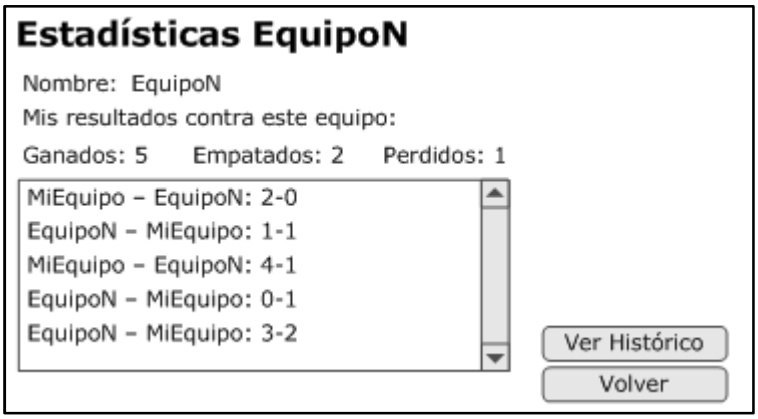
Entrada y Salida: Equipos.

2.4.4. EDITAR EQUIPO

Desde esta pantalla el usuario podrá editar la información que el sistema contenía acerca de cualquiera de sus equipos rivales, que debe estar registrado en el sistema.

Entrada y Salida: Equipos.

2.4.5. VER DETALLES EQUIPOS

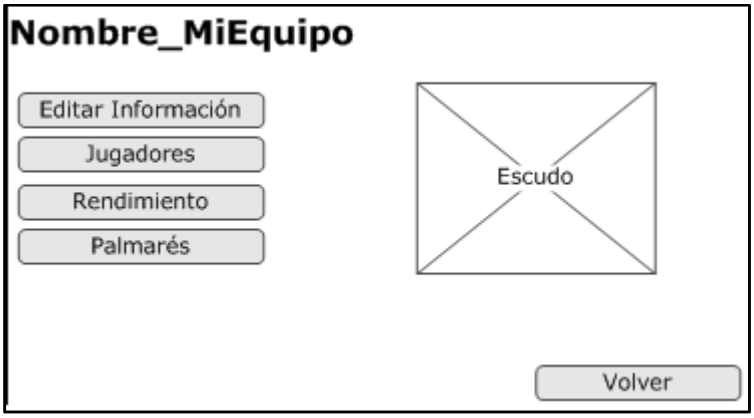


Esta ventana facilitará al usuario el análisis de la trayectoria de su equipo contra un equipo seleccionado en particular. Podrá ver todos los resultados de los partidos disputados contra ese equipo, y asimismo, podrá acceder a información detallada de cada uno de los partidos seleccionando uno de los de la lista y pulsando “Ver Histórico”

Entradas: Equipos.

Salidas: Equipos, Ver histórico partidos.

2.4.6. MI EQUIPO

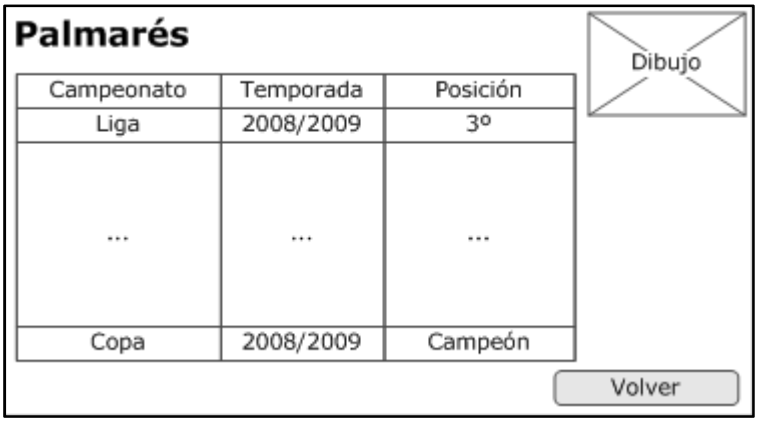


Esta interfaz proporcionará al usuario los mecanismos necesarios para poder gestionar la información de su propio equipo. Si al inicio el jugador no tienen ningún equipo creado, lo creará por medio del botón “Editar Información”. Se determinará más adelante la posibilidad de que el usuario maneje más de un equipo en el sistema, de momento, en este diseño inicial contemplamos sólo la posibilidad de que dirija sólo a un equipo.

Entradas: Inicial, Editar Información, Jugadores, Rendimiento, Palmarés.

Salidas: Editar Información, Jugadores, Rendimiento, Palmarés, Inicial.

2.4.7. PALMARÉS



En esta parte de la aplicación, el usuario podrá ver información presentada de modo tabular en la cual se especificará la clasificación en la que terminó cada una de las competiciones en las cuales participó.

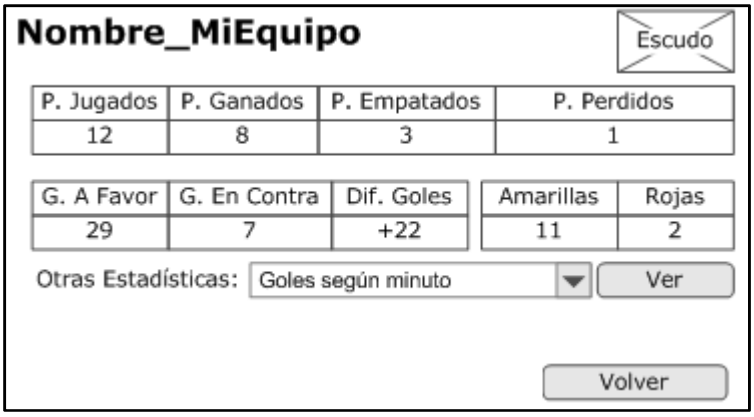
Entrada y Salida: Mi Equipo.

2.4.8. EDITAR INFORMACIÓN

En este apartado de la aplicación, al usuario se le permite modificar o introducir los datos pertinentes acerca del equipo que va a ser dirigido por él.

Entrada y Salida: Mi Equipo.

2.4.9. RENDIMIENTO EQUIPO

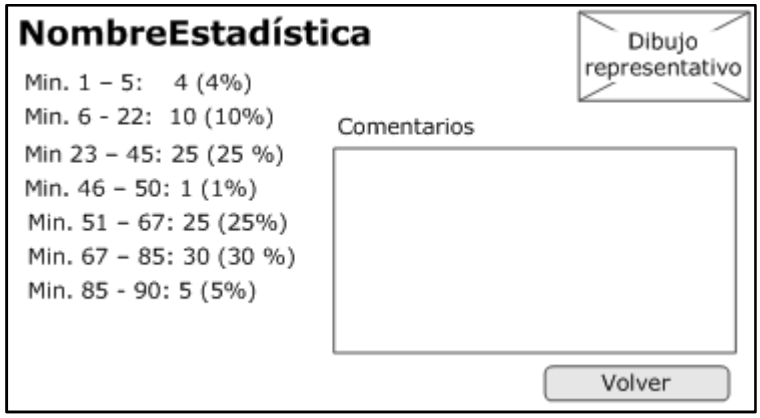


Esta ventana proporciona al usuario la comodidad de poder ver las estadísticas de su equipo. Se muestran las estadísticas básicas en la presentación, pero el usuario dispone de une funcionalidad que le permitirá acceder a una serie de estadísticas que desee.

Entradas: Mi Equipo, Ver Estadísticas de Rendimiento.

Salidas: Mi Equipo, Ver Estadísticas de Rendimiento.

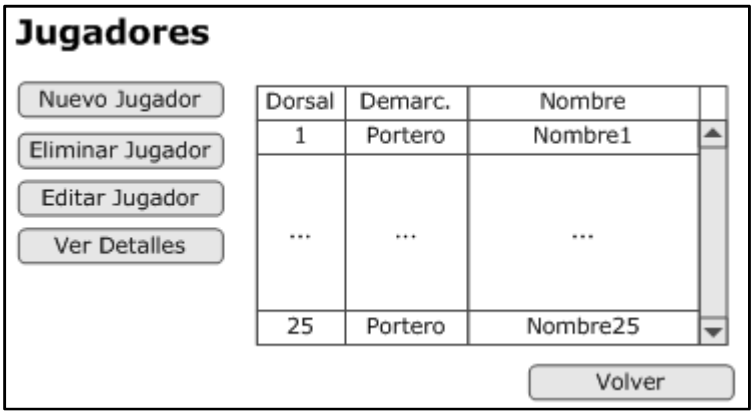
2.4.10. VER ESTADÍSTICAS DE RENDIMIENTO



En este módulo de la aplicación el usuario verá detalladamente una estadística concreta previamente seleccionada por él mismo. En la pantalla de ejemplo vemos como se desglosan los acontecimientos relativos a dicha estadística en función del periodo de partido en los que éstos acontecen, pudiendo ver a su vez una serie comentarios que él mismo haya introducido.

Entrada y Salida: Rendimiento Equipo.

2.4.11. JUGADORES



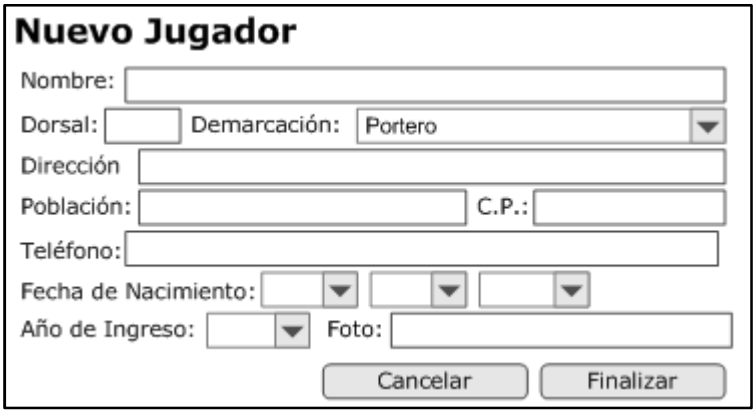
En esta pantalla el usuario dispondrá de la interfaz y la funcionalidad necesaria para poder analizar y gestionar los jugadores de su plantilla.

Entradas: Mi Equipo, Nuevo Jugador, Editar Jugador, Ver Detalles Jugador.

Salidas: Mi Equipo, Nuevo Jugador, Editar Jugador, Ver Detalles Jugador.

La eliminación de jugadores no requerirá de una pantalla nueva. Se llevará a cabo seleccionando la fila correspondiente al jugador en la tabla de la derecha y pulsando “Eliminar Jugador”. Los detalles del jugador serán accedidos también mediante el mismo procedimiento.

2.4.12. NUEVO JUGADOR



El usuario podrá dar de alta a un jugador en el sistema rellenando todos los datos en los campos correspondientes y pulsando “Finalizar”. Si antes de finalizar pulsa “Cancelar”, volverá a la pantalla anterior sin haber dado de alta a ningún jugador.

Entrada y Salida: Jugadores.

2.4.13. EDITAR JUGADOR

El usuario podrá editar la información relativa a cualquier jugador que previamente hubiese sido registrado en el sistema modificando los datos en los campos correspondientes y pulsando “Finalizar”. Si antes de finalizar pulsa “Cancelar”, volverá a la pantalla anterior sin haber modificado la información perteneciente al jugador en cuestión.

Entrada y Salida: Jugadores.

2.4.14. VER DETALLES JUGADOR

En esta ventana el usuario podrá analizar una a una las estadísticas más relevantes de todos y cada uno de sus jugadores. Cada jugador tendrá su propia pantalla de detalles donde se presentará su rendimiento de forma numérica.

Entrada: Jugadores, , Ver Partidos Jugados, Ver Minutos Disputados, Ver Goles Marcados, Ver Goles Recibidos, Ver Tarjetas, Ver Tarjetas Provocadas, Ver Lesiones.

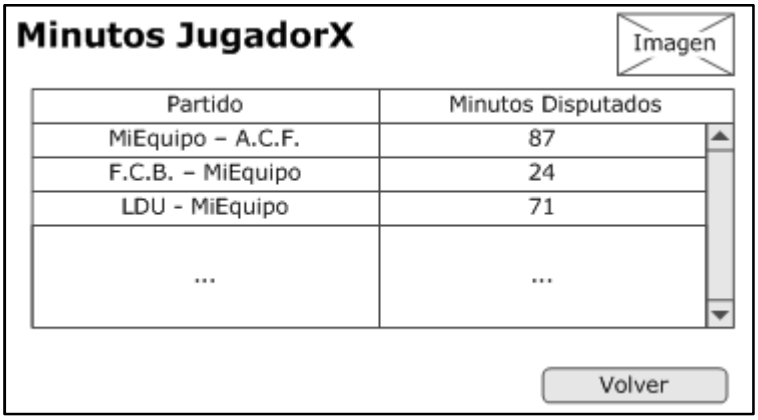
Salidas: Jugadores, Ver Partidos Jugados, Ver Minutos Disputados, Ver Goles Marcados, Ver Goles Recibidos, Ver Tarjetas, Ver Tarjetas Provocadas, Ver Lesiones.

2.4.15. VER PARTIDOS JUGADOS

Cada jugador tendrá una ventana de este estilo en la cual se presentará de forma tabular la información relativa a qué partidos ha disputado y en cuáles ha sido titular y en cuáles suplente. El número total de partidos jugados ya se muestra en la pantalla anterior y por tanto aquí no los mostraremos para no proporcionar información redundante.

Entrada y Salida: Ver Detalles Jugador.

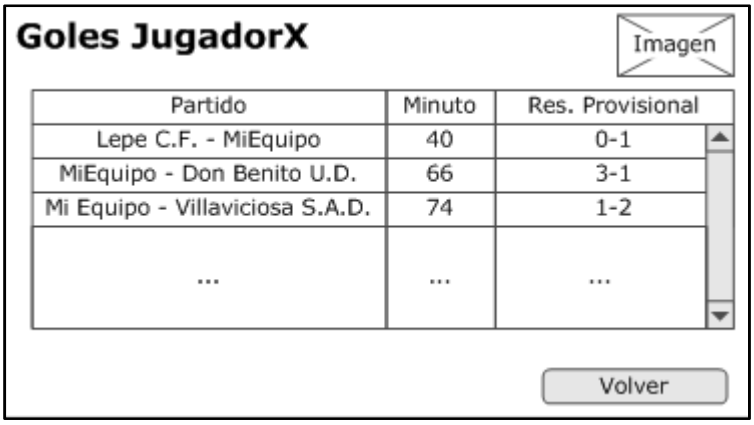
2.4.16. VER MINUTOS DISPUTADOS



Cada jugador tendrá una ventana de este estilo en la cual se presentará de forma tabular la información relativa a qué partidos ha disputado y cuántos minutos ha disputado en cada uno de ellos, siendo los valores posibles desde 1 hasta 90 minutos.

Entrada y Salida: Ver Detalles Jugador.

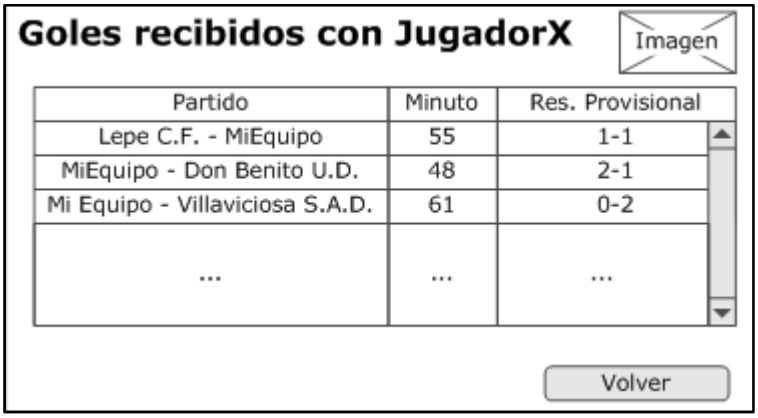
2.4.17. VER GOLES MARCADOS



En estas ventanas se informará al usuario de los goles marcados por un determinado jugador, siendo proporcionado tanto el partido en el cual marcó, como el minuto y el resultado provisional del partido en el momento en el cual se produjo el gol.

Entrada y Salida: Ver Detalles Jugador.

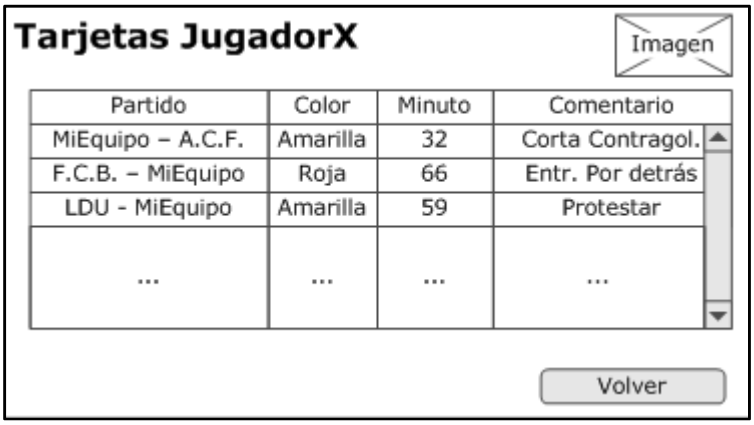
2.4.18. VER GOLES RECIBIDOS



En estas ventanas se informará al usuario de los goles que su equipo ha recibido estando un determinado jugador en el campo, siendo proporcionado tanto los partidos en los que el equipo recibió goles mientras el jugador era partícipe dl juego, como el minuto y el resultado provisional del partido en el momento en el cual se encajó el gol.

Entrada y Salida: Ver Detalles Jugador.

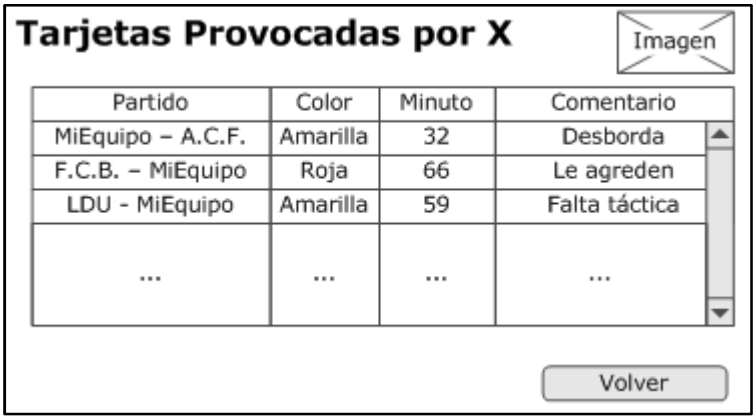
2.4.19. VER TARJETAS



Este apartado, igual que los más recientes comentados, proporcionará al usuario la información relativa a las tarjetas que le han sido mostradas a cada uno de sus jugadores, incluyéndose el partido, el color de la tarjeta, el minuto y un comentario que el usuario hubiese escrito en el momento en que dicha amonestación se produjo.

Entrada y Salida: Ver Detalles Jugador.

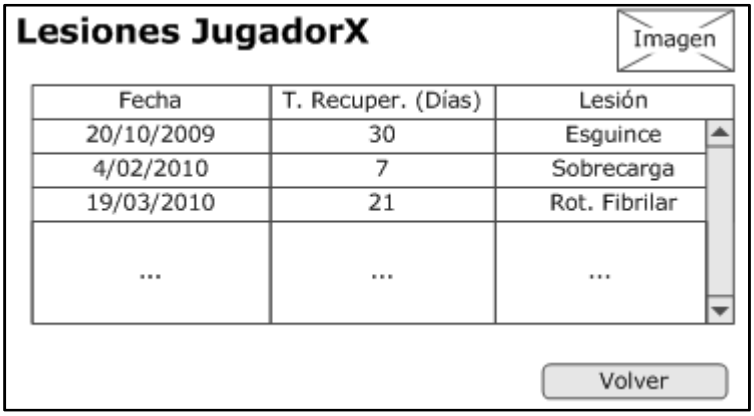
2.4.20. VER TARJETAS PROVOCADAS



De modo análogo a lo explicado en el apartado 2.1.19, en estas ventanas se obtendrá información de las tarjetas que cada uno de los jugadores del equipo gestionado por el usuario ha provocado a un rival, incluyéndose igual que en caso anterior, el partido, el color de la tarjeta, el minuto y un comentario que el usuario hubiese escrito en el momento en que dicha amonestación fue provocada.

Entrada y Salida: Ver Detalles Jugador.

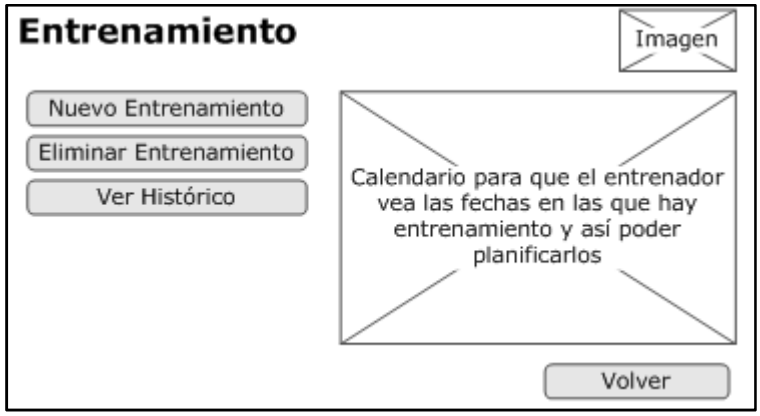
2.4.21. VER LESIONES



El usuario podrá obtener información del historial médico de un jugador, teniendo registrada para cada lesión que el mismo haya tenido la fecha en la cual esta se produjo, en qué consistía dicha lesión y el tiempo de recuperación que fue necesario para que el jugador volviera a ejercer la práctica deportiva con total normalidad.

Entrada y Salida: Ver Detalles Jugador.

2.4.22. ENTRENAMIENTO



Este menú proporciona acceso a la gestión de los entrenamientos realizados, que se realizarán o que están teniendo lugar.

Entrada: Inicial, Ver Histórico Entrenamientos, Nuevo Entrenamiento.

Salidas: Ver Histórico Entrenamientos, Nuevo Entrenamiento.

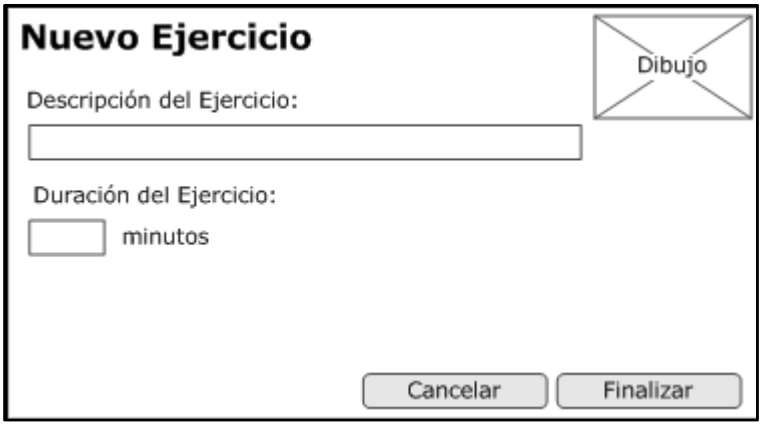
La eliminación de entrenamientos tendrá lugar, a priori seleccionando una fecha del calendario y pulsando “Eliminar Entrenamiento”. Se considera que sólo habrá un entrenamiento por día, y si hay varias sesiones de entrenamiento, en este modelo inicial se considerará todo como parte de un mismo entrenamiento.

2.4.23. NUEVO ENTRENAMIENTO

Desde esta ventana se proporciona la interfaz necesaria para crear un nuevo entrenamiento en un día concreto. Se podrán eliminar ejercicios, editarlos, comentarlos o añadir incidencias previa selección de la fila de la tabla correspondiente al ejercicio en cuestión.

Entradas y Salidas: Entrenamiento, Añadir Ejercicio, Editar Ejercicio, Comentar Ejercicio, Incidencias.

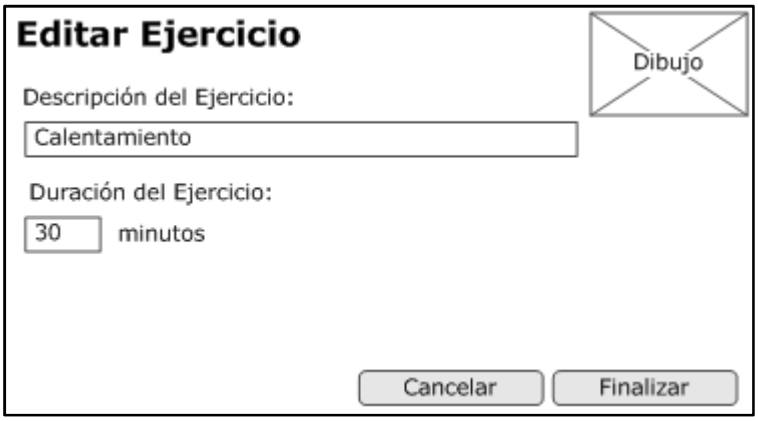
2.4.24 AÑADIR EJERCICIO



Desde esta pantalla el usuario podrá introducir un nuevo ejercicio que se realizará en el entrenamiento del día programado. Al finalizar, el usuario volverá a la pantalla de "Nuevo Entrenamiento" y en la tabla de ejercicios se habrá incluido la nueva actividad añadida. Si el usuario cancela no se almacena ningún cambio en el sistema.

Entrada y Salida: Nuevo Entrenamiento.

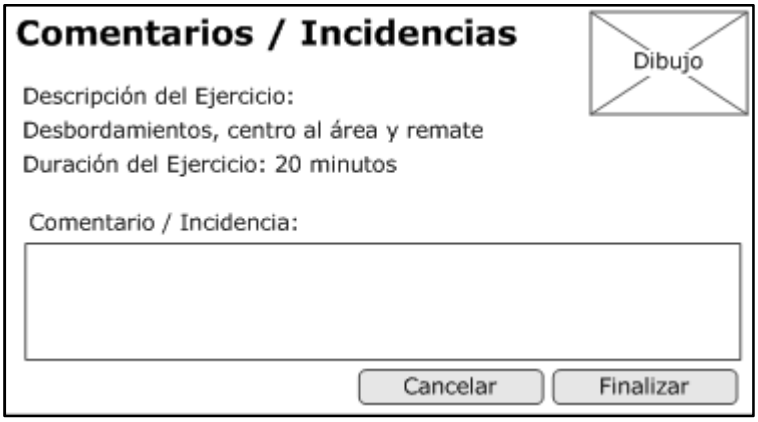
2.4.25. EDITAR EJERCICIO



El usuario podrá editar un ejercicio que ya había sido registrado y añadido a un determinado entrenamiento. Si finaliza, los cambios realizados se manifestarán en la tabla de la pantalla "Nuevo Entrenamiento" donde se muestran todos los ejercicios, pero si, por el contrario, cancela, el ejercicio no será modificado y no se apreciará cambio alguno en la tabla mencionada.

Entrada y Salida: Nuevo Entrenamiento.

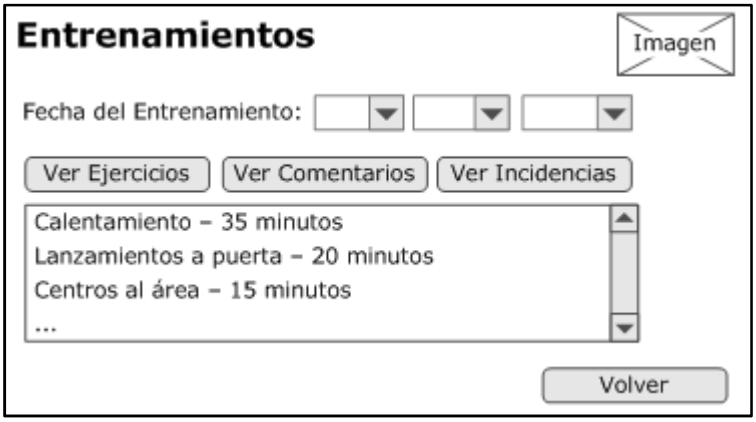
2.4.26. COMENTAR EJERCICIO E INCIDENCIAS



Estas dos ventanas de aspecto casi idéntico, proporcionan al usuario la posibilidad de añadir comentarios o comentar cualquier hecho reseñable durante el transcurso de cada entrenamiento o cada ejercicio del mismo. Si el usuario finaliza la acción, el comentario o incidencia será añadida al sistema y podrá ser visualizada desde otros módulos del sistema, como, por ejemplo, el 2.1.27 que se detallará a continuación. Si por el contrario canela la acción antes de finalizar, el sistema no registrará ningún cambio.

Entrada y Salida: Nuevo Entrenamiento.

2.4.27.VER HISTÓRICO ENTRENAMIENTOS



En este apartado de la aplicación, el entrenador podrá ver las incidencias, comentarios y ejercicios realizados durante un determinado entrenamiento para así tomar ideas para programar nuevos entrenamientos, para no repetir ejercicios en sesiones consecutivas o bien para analiza el rendimiento del equipo durante la semana para decidir cuál será el once inicial en el partido del fin de semana.

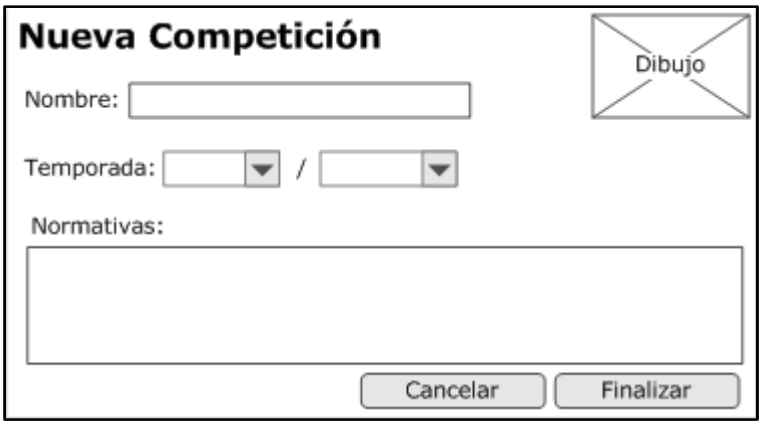
Entrada y Salida: Entrenamiento.

2.4.28. COMPETICIÓN

En este menú el usuario encontrará la interfaz y las funcionalidades que le permitirán gestionar las competiciones en las cuáles está inmerso su equipo. Para poder tener acceso a los partidos deberá estar seleccionada una fila de la tabla competiciones y se mostrarán los datos históricos relativos a esa competición, se añadirá un nuevo partido a esa competición, o se eliminará la misma y toda la información asociada a ésta mediante el pulsado del botón “Eliminar Competición”.

Entradas y Salidas: Inicial, Nueva Competición, Partidos.

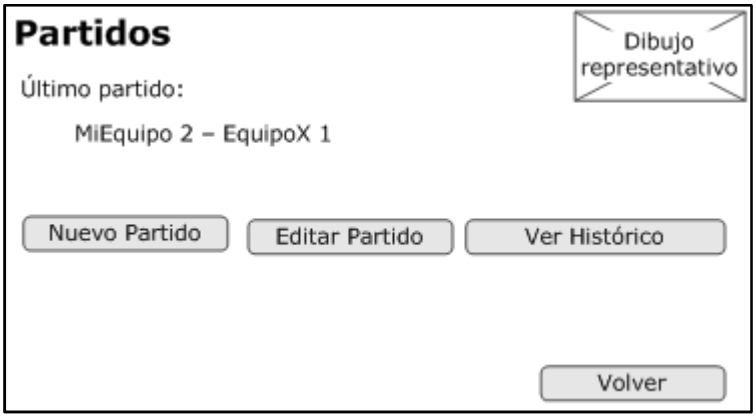
2.4.29. NUEVA COMPETICIÓN



Para dar de alta una competición y poder así iniciar la gestión de partidos, de datos de los jugadores, estadísticas, etc. el usuario deberá cumplimentar debidamente la información requerida en esta pantalla.

Entrada y Salida: Competición.

2.4.30. PARTIDOS



Desde este formulario el usuario podrá gestionar los partidos que disputa su equipo, tanto antes de que éstos comiencen como durante y después del mismo. Toda la información de cada partido será relativa a una competición en particular previamente seleccionada. Podrá obtenerse mecanismos a partir de aquí tanto para anotar eventos a pie de campo, como para analizar la evolución del equipo a lo largo de sus múltiples partidos en el histórico.

Entrada y Salida: Inicial, Nuevo Partido, Editar Partido, Ver Histórico Partidos.

2.4.31. NUEVO PARTIDO

Este formulario cobra un interés especial los días de partidos, ya que desde aquí el usuario podrá introducir y manejar todos los datos que crea necesarios mientras el partido avanza su curso. Se determinará el rival, el campo donde se juega, la fecha, y a partir de ahí se le ofrecerá al usuario un elenco de opciones para registrar los datos del partido, desde goles hasta lesiones y alineación, etc. Cualquier acción finalizada será registrada en el sistema, si se cancela los cambios no surtirán efecto.

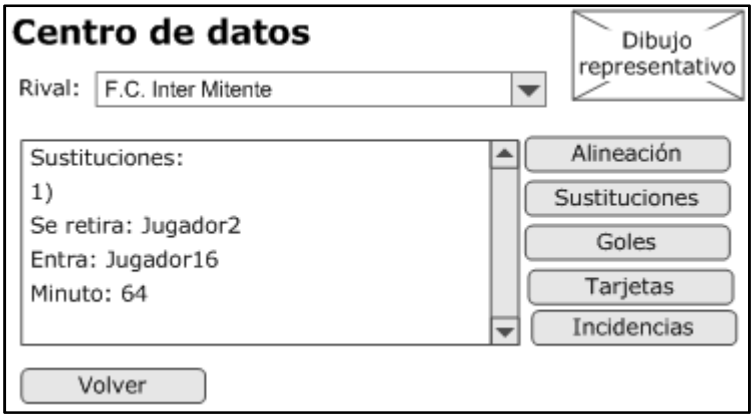
Entradas y Salidas: Partidos, Alineación Inicial, Sustituciones, Goles, Tarjetas, Lesiones, Incidencias.

2.4.32. EDITAR PARTIDO

Esta pantalla permitirá al usuario editar cualquier dato referente a un partido que, por supuesto, haya sido creado o dado de alta en el sistema previamente. Cualquier cambio realizado tendrá efecto al finalizar la acción, si se cancela, todo cambio se perderá.

Entradas y Salidas: Partidos, Alineación Inicial, Sustituciones, Goles, Tarjetas, Lesiones, Incidencias.

2.4.33. VER HISTÓRICO PARTIDOS

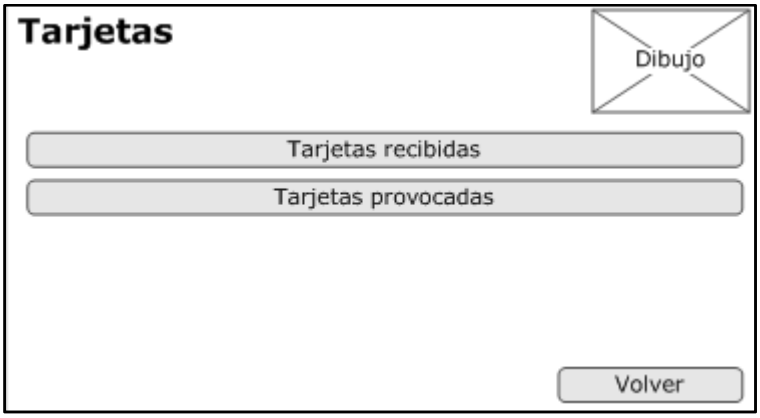


En este apartado de la aplicación, el entrenador podrá ver las incidencias, amonestaciones, goles alineación y sustituciones realizados durante un determinado partido para así tomar decisiones de cara a programar nuevos partidos, o para analizar el partido del equipo de cara a optimizar el rendimiento de grupo de cara a próximos partidos.

Las estadísticas se mostrarán el cuadro de texto central mediante la pulsación del botón correspondiente a la estadística del partido que el usuario quisiera analizar. En el ejemplo se han seleccionado “Sustituciones”.

Entradas: Partidos, Ver Detalles Equipos. Salidas: Partidos.

2.4.34. GOLES Y TARJETAS

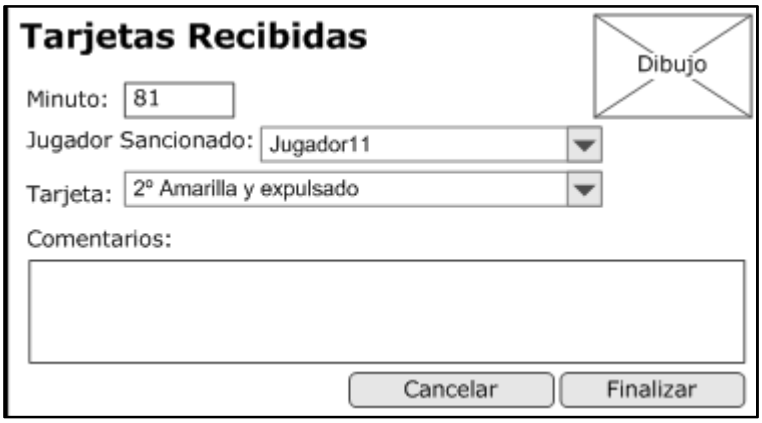


Unimos estos dos apartados porque, a pesar de ser pantallas diferentes su explicación es análoga. En estos formularios se proporciona la interfaz al usuario para que anote las tarjetas que les muestran a sus jugadores, o bien las que éstos provocan. En el caso de los goles, se accederá a poder anotar los goles marcados por el equipo, así como los que el equipo encaja.

Entradas y Salidas Tarjetas: Partido, Tarjetas en Contra, Tarjetas Provocadas.

Entradas y Salidas Goles: Partido, Goles Marcados, Goles Encajados.

2.4.35. GOLES Y TARJETAS DEL EQUIPO PROPIO

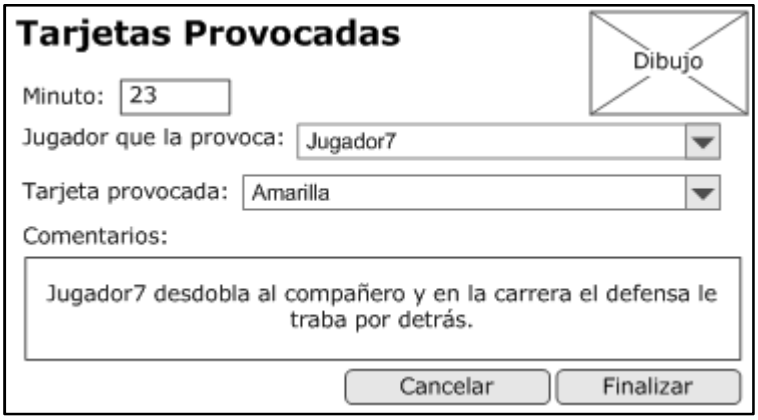


En estos formularios el usuario podrá introducir los datos referentes a goles que sus jugadores anotan, así como a tarjetas que sus jugadores reciben por conductas extra reglamentarias.

Entradas y Salidas Tarjetas Recibidas: Tarjetas.

Entradas y Salidas Goles Marcados: Goles.

2.4.36. GOLES RECIBIDOS Y TARJETAS PROVOCADAS



En estas pantallas el usuario podrá introducir los datos referentes a goles que su equipo recibe, así como a tarjetas que sus jugadores provocan al adversario. Esta parte se centra en datos referentes a las facilidades o conductas que los jugadores del equipo dirigido por el usuario tienen para con sus adversarios.

Entradas y Salidas Tarjetas Provocadas: Tarjetas.

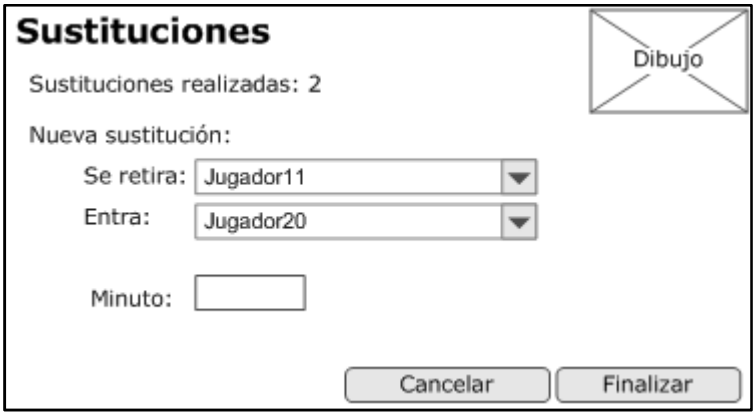
Entradas y Salidas Goles Recibidos: Goles.

2.4.37. ALINEACIÓN INICIAL

Este formulario proporcionará una interfaz sencilla para que él usuario-entrenador determine qué jugadores formarán parte de la alineación inicial de un determinado partido. Podrá pasar un jugador de la plantilla a la alineación inicial pulsando el botón “>>”, y viceversa pulsando “<<”. Los cambios que se cancelen no serán registrados, y la alineación inicial que se almacenará en el sistema para el partido completo será la compuesta por los jugadores del cuadro de la derecha en el momento de pulsar “Finalizar”.

Entradas y Salidas: Nuevo Partido, Editar Partido.

2.4.38. SUSTITUCIONES



En esta pantalla el usuario encontrará la funcionalidad que le permitirá registrar en el sistema las sustituciones que se realizan en un encuentro. Obtendrá la información de cuantas sustituciones ha realizado ya, y podrá establecer que jugador sustituye a qué otro, siendo requisito que uno de ellos esté jugando y el otro no esté alineado, y que el jugador que entra al formar parte de la alineación no ha sido previamente sustituido en el mismo partido.

La información que se almacenará será la que contenga el formulario al pulsar el botón “Finalizar” (siempre y cuando estén todos los campos rellenos de manera correcta), y los cambios que se cancelen no serán almacenados y, por tanto, dicha sustitución no será almacenada en el sistema, obteniendo el usuario al volver a acceder a este formulario el mismo número de sustituciones realizadas que la vez anterior cuando se canceló el cambio.

Entradas y Salidas: Nuevo Partido, Editar Partido.

3. FASE DE IMPLEMENTACIÓN

3.1. INTERFAZ GRÁFICA DE USUARIO

El trabajo de la fase de diseño explicado y justificado anteriormente toma forma en la fase de implementación que en este punto se procede a explicar con detalle.

El desarrollo y programación del proyecto empieza con la implementación de la interfaz gráfica de usuario sin funcionalidad alguna, es decir, sin implementar las acciones correspondientes a cada uno de los casos de uso descritos como requerimientos del sistema.

Para llevar a cabo esta labor, nos hemos ayudado de la herramienta Qt Designer que incorpora el entorno de programación Qt Creator bajo el sistema operativo Windows, así como de las librerías QtGui y QtCore incorporadas en el propio entorno. Destacar que la herramienta Qt Designer genera código automáticamente en una serie de ficheros cuyos nombre encajan con el patrón "ui_X.h", en esta caso, ui_mainwindow.h, que en todo momento ha sido supervisado para verificar lanecesidad de correcciones en caso de haberse dado.

La interfaz gráfica de usuario implementada en este apartado reproduce fielmente el diseño de las pantallas de los puntos anteriores, y las interconexiones y transiciones que entre ellas se definen en la máquina de estados que define el funcionamiento de la aplicación. Se omite en este documento el aspecto de cada una de las pantallas debido a que junto a esta memoria se adjunta un vídeo llamado "GUI _SinFuncionalidad.avi" en el cual se hace un recorrido detallado por todas y cada una de las pantallas y ventanas que en conjunto definen el total de la aplicación, pudiéndose apreciar en él las transiciones entre las partes de la interfaz gráfica del sistema.

A continuación se procederá a explicar las directrices y el método de implementación seguido para llevar a cabo la tarea que en este apartado se describe, ya que es requisito indispensable de este proyecto que la aplicación funcione como una máquina de estados, y así sucede.

Para la comprensión de la metodología, en primer lugar facilitaremos un elenco de métodos que han sido necesarios para el total desarrollo de la interfaz.

En primer lugar, definimos un atributo "state" que sirve para indicar en que estado se encuentra la ventana principal de la aplicación, y dependiendo de ese estado, mostrará una pantalla u otra. Se definen también una serie de métodos y de slots que encuentran definidos en el fichero MainWindow.h e implementados en MainWindow.cpp, que son los siguientes que se muestran en la Tabla 3.

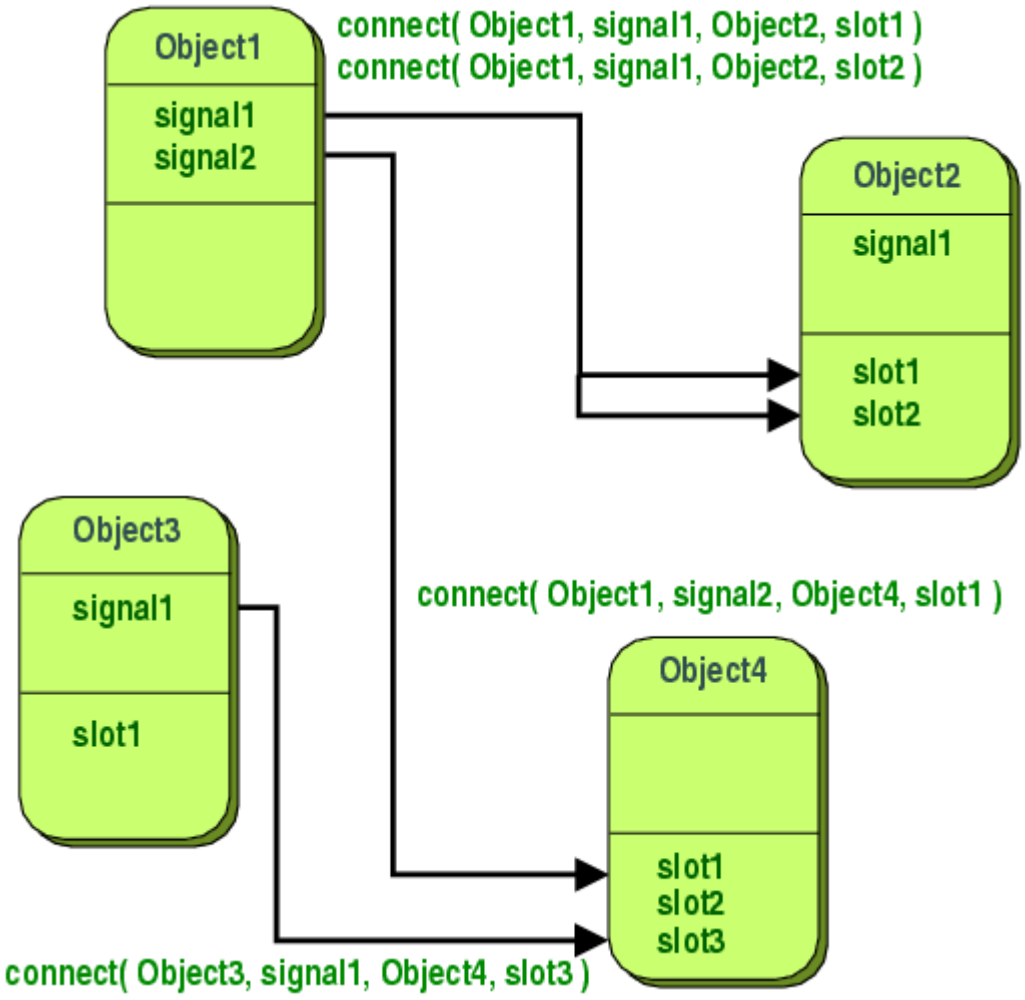
```
public:
   // …
   void construirMaquinaEstados();

public slots:
   // Slots de la Máquina de Estados
    void stateChanged(int state);

    void startInicio();

    void startEquipos();
    void startNuevoEquipo();
    void startEditarEquipo();
    void startVerDetallesEquipo();
```

```
void startMiEquipo();
void startJugadores();
void startNuevoJugador();
void startEditarJugador();
void startVerDetallesJugador();
void startVerPartidosJugados();
void startVerMinutosDisputados();
void startVerGolesMarcados();
void startVerGolesRecibidos();
void startVerTarjetasRecibidas();
void startVerTarjetasProvocadas();
void startVerLesiones();
void startPalmares();
void startNuevoTitulo();
void startEditarTitulo();
void startInfoMiEquipo();
void startRendimientoMiEquipo();
void startGolesMarcadosPorMinuto();
void startGolesRecibidosPorMinuto();
void startAmonestacionesPorMinuto();

void startEntrenamiento();
void startNuevoEntrenamiento();
void startNuevoEjercicio();
void startEditarEjercicio();
void startComentarEjercicio();
void startIncidenciasEjercicio();
void startHistoricoEntrenamientos();

void startCompeticion();
void startNuevaCompeticion();
void startPartidos();
void startVerHistoricoPartidos();
void startNuevoPartido();
void startNuevaAlineacionInicial();
void startNuevaSustitucion();
void startNuevoGol();
void startNuevoGolMarcado();
void startNuevoGolRecibido();
void startNuevaTarjeta();
void startNuevaTarjetaRecibida();
void startNuevaTarjetaProvocada();
void startNuevaLesion();
void startNuevaIncidencia();
void startEditarPartido();
void startEditarAlineacionInicial();
void startEditarSustitucion();
void startEditarGol();
void startEditarGolMarcado();
void startEditarGolRecibido();
void startEditarTarjeta();
void startEditarTarjetaRecibida();
void startEditarTarjetaProvocada();
void startEditarLesion();
void startEditarIncidencia();
```

- Tabla 3 -

Antes de continuar, definimos los slots en Qt como mecanismos de comunicación entre objetos, en el cual un objeto reacciona a una determinada señal que recibe, como puede ser, por ejemplo un evento

de pulsación de un botón, con un determinado slot, o acción de respuesta, unido todo mediante la instrucción connect. Ver Figura 3.



- Figura 3 -

Cada uno de los slots define la acción a realizar cuando la ventana principal recibe el evento correspondiente para que cada pantalla en cuestión sea activada.

La aplicación consta de una ventana principal que, tras numerosos problemas a la hora de implementar las pantallas y unirlas a ésta como widgets, tiene en sí un QFrame por cada estado de la máquina que representa cada una de las pantallas ya definidas esquemáticamente en la fase de diseño que se hace visible cuando las transiciones de la máquina hacen llegar a la misma al estado correspondiente. La gestión de la visibilidad de los QFrames se lleva a cabo mediante el método MainWindow::stateChanged(int state), en el cual se ponen todos los QFrames como no visibles y en función del estado que se le pase al método, se hará visible uno en concreto, de este modo, al pasar a otro estado, se volverá a invocar a stateChanged con otro estado como parámetro, y el QFrame anterior se hará invisible y entonces activará el correspondiente. Se especifica el código de este método en la Tabla 4.

```
void MainWindow::stateChanged(int state)
{
   // Todos los frames están ocultos
   this->ui->inicioFrame->setVisible(false);

   this->ui->equiposFrame->setVisible(false);
   this->ui->nuevoEquipoFrame->setVisible(false);
   this->ui->editarEquipoFrame->setVisible(false);
   this->ui->verDetallesEquipoFrame->setVisible(false);

   this->ui->miEquipoFrame->setVisible(false);
   this->ui->jugadoresFrame->setVisible(false);
   this->ui->nuevoJugadorFrame->setVisible(false);
   this->ui->editarJugadorFrame->setVisible(false);
   this->ui->verDetallesJugadorFrame->setVisible(false);
   this->ui->partidosJugadosFrame->setVisible(false);
   this->ui->minutosDisputadosFrame->setVisible(false);
   this->ui->golesMarcadosFrame->setVisible(false);
   this->ui->golesRecibidosFrame->setVisible(false);
   this->ui->tarjetasRecibidasFrame->setVisible(false);
   this->ui->tarjetasProvocadasFrame->setVisible(false);
   this->ui->lesionesFrame->setVisible(false);
   this->ui->palmaresFrame->setVisible(false);
   this->ui->nuevoTituloFrame->setVisible(false);
   this->ui->editarTituloFrame->setVisible(false);
   this->ui->infoMiEquipoFrame->setVisible(false);
   this->ui->rendimientoMiEquipoFrame->setVisible(false);
   this->ui->golesMarcadosPorMinutoFrame->setVisible(false);
   this->ui->golesRecibidosPorMinutoFrame->setVisible(false);
   this->ui->amonestacionesPorMinutoFrame->setVisible(false);

   this->ui->entrenamientosFrame->setVisible(false);
   this->ui->nuevoEntrenamientoFrame->setVisible(false);
   this->ui->nuevoEjercicioFrame->setVisible(false);
   this->ui->editarEjercicioFrame->setVisible(false);
   this->ui->comentarEjercicioFrame->setVisible(false);
   this->ui->incidenciasEjercicioFrame->setVisible(false);
   this->ui->historicoEntrenamientoFrame->setVisible(false);

   this->ui->competicionFrame->setVisible(false);
   this->ui->nuevaCompeticionFrame->setVisible(false);
   this->ui->partidosFrame->setVisible(false);
   this->ui->verHistoricoPartidosFrame->setVisible(false);
   this->ui->nuevoPartidoFrame->setVisible(false);
   this->ui->nuevaAlineacionInicialFrame->setVisible(false);
   this->ui->nuevaSustitucionFrame->setVisible(false);
   this->ui->nuevoGolFrame->setVisible(false);
   this->ui->nuevoGolMarcadoFrame->setVisible(false);
   this->ui->nuevoGolRecibidoFrame->setVisible(false);
   this->ui->nuevaTarjetaFrame->setVisible(false);
   this->ui->nuevaTarjetaRecibidaFrame->setVisible(false);
   this->ui->nuevaTarjetaProvocadaFrame->setVisible(false);
   this->ui->nuevaLesionFrame->setVisible(false);
   this->ui->nuevaIncidenciaFrame->setVisible(false);
   this->ui->editarPartidoFrame->setVisible(false);
   this->ui->editarAlineacionInicialFrame->setVisible(false);
   this->ui->editarSustitucionFrame->setVisible(false);
   this->ui->editarGolFrame->setVisible(false);
   this->ui->editarGolMarcadoFrame->setVisible(false);
   this->ui->editarGolRecibidoFrame->setVisible(false);
   this->ui->editarTarjetaFrame->setVisible(false);
   this->ui->editarTarjetaRecibidaFrame->setVisible(false);
   this->ui->editarTarjetaProvocadaFrame->setVisible(false);
```

```
this->ui->editarLesionFrame->setVisible(false);
this->ui->editarIncidenciaFrame->setVisible(false);

// Se muestra el frame correspondiente al estado activo
if (state == 0){
   this->ui->inicioFrame->setVisible(true);
}

// Equipos
else if (state == 1){
   this->ui->equiposFrame->setVisible(true);
}
else if (state == 11){
   this->ui->nuevoEquipoFrame->setVisible(true);
}
else if (state == 12){
   this->ui->editarEquipoFrame->setVisible(true);
}
else if (state == 13){
   this->ui->verDetallesEquipoFrame->setVisible(true);
}

// Mi Equipo
else if (state == 2){
   this->ui->miEquipoFrame->setVisible(true);
}
else if (state == 21){
   this->ui->jugadoresFrame->setVisible(true);
}
else if (state == 211){
   this->ui->nuevoJugadorFrame->setVisible(true);
}
else if (state == 212){
   this->ui->editarJugadorFrame->setVisible(true);
}
else if (state == 213){
   this->ui->verDetallesJugadorFrame->setVisible(true);
}
else if (state == 2131){
   this->ui->partidosJugadosFrame->setVisible(true);
}
else if (state == 2132){
   this->ui->minutosDisputadosFrame->setVisible(true);
}
else if (state == 2133){
   this->ui->golesMarcadosFrame->setVisible(true);
}
else if (state == 2134){
   this->ui->golesRecibidosFrame->setVisible(true);
}
else if (state == 2135){
   this->ui->tarjetasRecibidasFrame->setVisible(true);
}
else if (state == 2136){
   this->ui->tarjetasProvocadasFrame->setVisible(true);
}
else if (state == 2137){
   this->ui->lesionesFrame->setVisible(true);
}
else if (state == 22){
   this->ui->palmaresFrame->setVisible(true);
}
else if (state == 221){
```

```
    this->ui->nuevoTituloFrame->setVisible(true);
}
else if (state == 222){
    this->ui->editarTituloFrame->setVisible(true);
}
else if (state == 23){
    this->ui->infoMiEquipoFrame->setVisible(true);
}
else if (state == 24){
    this->ui->rendimientoMiEquipoFrame->setVisible(true);
}
else if (state == 241){
    this->ui->golesMarcadosPorMinutoFrame->setVisible(true);
}
else if (state == 242){
    this->ui->golesRecibidosPorMinutoFrame->setVisible(true);
}
else if (state == 243){
    this->ui->amonestacionesPorMinutoFrame->setVisible(true);
}

// Entrenamiento
else if (state == 3){
    this->ui->entrenamientosFrame->setVisible(true);
}
else if (state == 31){
    this->ui->nuevoEntrenamientoFrame->setVisible(true);
}
else if (state == 311){
    this->ui->nuevoEjercicioFrame->setVisible(true);
}
else if (state == 312){
    this->ui->editarEjercicioFrame->setVisible(true);
}
else if (state == 313){
    this->ui->comentarEjercicioFrame->setVisible(true);
}
else if (state == 314){
    this->ui->incidenciasEjercicioFrame->setVisible(true);
}
else if (state == 32){
    this->ui->historicoEntrenamientoFrame->setVisible(true);
}

// Competición
else if (state == 4){
    this->ui->competicionFrame->setVisible(true);
}
else if (state == 41){
    this->ui->nuevaCompeticionFrame->setVisible(true);
}
else if (state == 42){
    this->ui->partidosFrame->setVisible(true);
}
else if (state == 421){
    this->ui->verHistoricoPartidosFrame->setVisible(true);
}
else if (state == 422){
    this->ui->nuevoPartidoFrame->setVisible(true);
}
else if (state == 4221){
    this->ui->nuevaAlineacionInicialFrame->setVisible(true);
}
```

```
    else if (state == 4222){
        this->ui->nuevaSustitucionFrame->setVisible(true);
    }
    else if (state == 4223){
        this->ui->nuevoGolFrame->setVisible(true);
    }
    else if (state == 42231){
        this->ui->nuevoGolMarcadoFrame->setVisible(true);
    }
    else if (state == 42232){
        this->ui->nuevoGolRecibidoFrame->setVisible(true);
    }
    else if (state == 4224){
        this->ui->nuevaTarjetaFrame->setVisible(true);
    }
    else if (state == 42241){
        this->ui->nuevaTarjetaRecibidaFrame->setVisible(true);
    }
    else if (state == 42242){
        this->ui->nuevaTarjetaProvocadaFrame->setVisible(true);
    }
    else if (state == 4225){
        this->ui->nuevaLesionFrame->setVisible(true);
    }
    else if (state == 4226){
        this->ui->nuevaIncidenciaFrame->setVisible(true);
    }
    else if (state == 423){
        this->ui->editarPartidoFrame->setVisible(true);
    }
    else if (state == 4231){
        this->ui->editarAlineacionInicialFrame->setVisible(true);
    }
    else if (state == 4232){
         this->ui->editarSustitucionFrame->setVisible(true);
    }
    else if (state == 4233){
        this->ui->editarGolFrame->setVisible(true);
    }
    else if (state == 42331){
         this->ui->editarGolMarcadoFrame->setVisible(true);
    }
    else if (state == 42332){
         this->ui->editarGolRecibidoFrame->setVisible(true);
    }
    else if (state == 4234){
         this->ui->editarTarjetaFrame->setVisible(true);
    }
    else if (state == 42341){
         this->ui->editarTarjetaRecibidaFrame->setVisible(true);
    }
    else if (state == 42342){
         this->ui->editarTarjetaProvocadaFrame->setVisible(true);
    }
    else if (state == 4235){
         this->ui->editarLesionFrame->setVisible(true);
    }
    else if (state == 4236){
         this->ui->editarIncidenciaFrame->setVisible(true);
    }
}
```

- Tabla 4 -

Así pues, es sencillo ahora entender, que cada uno de los slots da un valor concreto al atributo “state”, y una vez proporcionado ese valor, el slot invoca a al método stateChanged que se encargará de la visualización del QFrame pertinente en la ventana principal. Veamos en la tabla 5 un par de ejemplos del contenido de los slots, ya que el resto tienen comportamiento análogo.

// Slot que se invoca en el estado inicial (Estado 0)
void MainWindow::startInicio()
{
  state = 0;
  stateChanged(state);
}
	// Slot que se invoca en el estado nuevoGolMarcado (Estado 42231_4 por competición, 2 por Partidos, 2 por Nuevo Partido, 3 por Goles, 1 por Goles Marcados. El número del estado atiende a su nivel de profundidad a medida que se ha implementado.
	)
	void MainWindow::startNuevoGolMarcado()
	{
	  state = 42231;
	  stateChanged(state);
	}

- Tabla 5 -

Por último, detallamos a continuación a la “madre” de todos estos métodos y slots, la función necesaria para que se cree la máquina de estados, y toda la infraestructura de llamadas anidadas cobre sentido y dé una semántica funcional a la interfaz gráfica del usuario. El método MainWindow::construirMaquinaEstados() declarado en MainWindow.h e implementado en MainWindow.cpp. Antes de introducir el código completo de este método, explicaremos brevemente funcionamiento del Qt State Machine Framework adaptado a nuestro proyecto.

En primer lugar se crea la máquina y los estados de la siguiente manera.

```
// Declaración de la Máquina
QStateMachine *machine = new QStateMachine(this);

// Declaración de Estados
 QState *estadoInicio = new QState();
 connect(estadoInicio, SIGNAL(entered()), this, SLOT (startInicio()));

QState *estadoEquipos = new QState();
connect(estadoEquipos, SIGNAL(entered()), this, SLOT(startEquipos()));
```

La función “connect” permite que cuando se entra en cada uno de los estados, el evento produzca como respuesta la acción que el slot correspondiente determine, comportamientos éstos ya descritos con anterioridad.
En segundo lugar, se crean las transiciones entre los estados.

```
// Declaración de Transiciones entre Inicio y Equipos
estadoInicio->addTransition(this->ui->botonInicio_Equipos, SIGNAL(clicked()), estadoEquipos);
```

En este caso, estadoInicio tiene una transición en la cual, cuando el botón Equipos es “clickado”, se transita al estadoEquipos.
Tras definir todos los estados y todas las transiciones, los estados son añadidos a la máquina, a ésta se le dice cuál es el estado inicial, y se arranca la misma de la siguiente manera.

```
// Añadimos los estados a la máquina
machine->addState(estadoInicio);
machine->addState(estadoEquipos);
```

```
// Ajustamos el estado Inicial de la Máquina de Estados "machine" y la iniciamos
machine->setInitialState(estadoInicio);
machine->start();
```

Tras esta breve explicación, estamos en posición ya de mostrar el código fuente íntegro del método encargado de construir y gestionar la máquina de estados. Ver Tabla 6.

```
// Construcción de la Máquina de Estados
void MainWindow::construirMaquinaEstados()
{
  // Declaración de la Máquina
  QStateMachine *machine = new QStateMachine(this);

  // Declaración de los Estados

  // Inicio ////////////////////////////////////////////////////////
  QState *estadoInicio = new QState();
  connect(estadoInicio, SIGNAL(entered()), this, SLOT (startInicio()));

  // Menú de Equipos y miveles inferiores ////////////////////////////
  QState *estadoEquipos = new QState();
  connect(estadoEquipos, SIGNAL(entered()), this, SLOT(startEquipos()));

  QState *nuevoEquipo = new QState();
  connect(nuevoEquipo, SIGNAL(entered()), this, SLOT(startNuevoEquipo()));

  QState *editarEquipo = new QState();
  connect(editarEquipo, SIGNAL(entered()), this, SLOT(startEditarEquipo()));

  QState *verDetallesEquipo = new QState();
  connect(verDetallesEquipo, SIGNAL(entered()), this, SLOT(startVerDetallesEquipo()));


  // Menú Mi Equipo y niveles inferiores ////////////////////////////
  QState *estadoMiEquipo = new QState();
  connect(estadoMiEquipo, SIGNAL(entered()), this, SLOT(startMiEquipo()));

  // Jugadores
  QState *estadoJugadores = new QState();
  connect(estadoJugadores, SIGNAL(entered()), this, SLOT(startJugadores()));

  QState *nuevoJugador = new QState();
  connect(nuevoJugador, SIGNAL(entered()), this, SLOT(startNuevoJugador()));

  QState *editarJugador = new QState();
  connect(editarJugador, SIGNAL(entered()), this, SLOT(startEditarJugador()));

  QState *verDetallesJugador = new QState();
  connect(verDetallesJugador, SIGNAL(entered()), this, SLOT(startVerDetallesJugador()));

  QState *verPartidosJugados = new QState();
  connect(verPartidosJugados, SIGNAL(entered()), this, SLOT(startVerPartidosJugados()));

  QState *verMinutosDisputados = new QState();
  connect(verMinutosDisputados, SIGNAL(entered()), this, SLOT(startVerMinutosDisputados()));

  QState *verGolesMarcados = new QState();
  connect(verGolesMarcados, SIGNAL(entered()), this, SLOT(startVerGolesMarcados()));

  QState *verGolesRecibidos = new QState();
  connect(verGolesRecibidos, SIGNAL(entered()), this, SLOT(startVerGolesRecibidos()));
```

```
QState *verTarjetasRecibidas = new QState();
connect(verTarjetasRecibidas, SIGNAL(entered()), this, SLOT(startVerTarjetasRecibidas()));

QState *verTarjetasProvocadas = new QState();
connect(verTarjetasProvocadas, SIGNAL(entered()), this, SLOT(startVerTarjetasProvocadas()));

QState *verLesiones = new QState();
connect(verLesiones, SIGNAL(entered()), this, SLOT(startVerLesiones()));

// Palmarés
QState *estadoPalmares = new QState();
connect(estadoPalmares, SIGNAL(entered()), this, SLOT(startPalmares()));

QState *nuevoTitulo = new QState();
connect(nuevoTitulo, SIGNAL(entered()), this, SLOT(startNuevoTitulo()));

QState *editarTitulo = new QState();
connect(editarTitulo, SIGNAL(entered()), this, SLOT(startEditarTitulo()));

// Información Mi Equipo
QState *infoMiEquipo = new QState();
connect(infoMiEquipo, SIGNAL(entered()), this, SLOT(startInfoMiEquipo()));

// Rendimiento Mi Equipo
QState *rendimientoMiEquipo = new QState();
connect(rendimientoMiEquipo, SIGNAL(entered()), this, SLOT(startRendimientoMiEquipo()));

QState *golesMarcadosPorMinuto = new QState();
connect(golesMarcadosPorMinuto, SIGNAL(entered()), this, SLOT(startGolesMarcadosPorMinuto()));

QState *golesRecibidosPorMinuto = new QState();
connect(golesRecibidosPorMinuto, SIGNAL(entered()), this, SLOT(startGolesRecibidosPorMinuto()));

QState *amonestacionesPorMinuto = new QState();
connect(amonestacionesPorMinuto, SIGNAL(entered()), this, SLOT(startAmonestacionesPorMinuto()));

// Menú Entrenamiento y niveles inferiores ////////////////////////////
QState *estadoEntrenamiento = new QState();
connect(estadoEntrenamiento, SIGNAL(entered()), this, SLOT(startEntrenamiento()));

// Nuevo Entrenamiento
QState *nuevoEntrenamiento = new QState();
connect(nuevoEntrenamiento, SIGNAL(entered()), this, SLOT(startNuevoEntrenamiento()));

QState *nuevoEjercicio = new QState();
connect(nuevoEjercicio, SIGNAL(entered()), this, SLOT(startNuevoEjercicio()));

QState *editarEjercicio = new QState();
connect(editarEjercicio, SIGNAL(entered()), this, SLOT(startEditarEjercicio()));

QState *comentarEjercicio = new QState();
connect(comentarEjercicio, SIGNAL(entered()), this, SLOT(startComentarEjercicio()));

QState *incidenciasEjercicio = new QState();
connect(incidenciasEjercicio, SIGNAL(entered()), this, SLOT(startIncidenciasEjercicio()));

// Histórico entrenamientos
QState *historicoEntrenamientos = new QState();
connect(historicoEntrenamientos, SIGNAL(entered()), this, SLOT(startHistoricoEntrenamientos()));

// Menú Competición y niveles inferiores ////////////////////////////////////////
QState *estadoCompeticion = new QState();
connect(estadoCompeticion, SIGNAL(entered()), this, SLOT(startCompeticion()));
```

```
// Nueva Competicion
QState *nuevaCompeticion = new QState();
connect(nuevaCompeticion, SIGNAL(entered()), this, SLOT(startNuevaCompeticion()));

// Partidos
QState *estadoPartidos = new QState();
connect(estadoPartidos, SIGNAL(entered()), this, SLOT(startPartidos()));

QState *historicoPartidos = new QState();
connect(historicoPartidos, SIGNAL(entered()), this, SLOT(startVerHistoricoPartidos()));

// Nuevo Partido
QState *nuevoPartido = new QState();
connect(nuevoPartido, SIGNAL(entered()), this, SLOT(startNuevoPartido()));

QState *nuevaAlineacionInicial = new QState();
connect(nuevaAlineacionInicial, SIGNAL(entered()), this, SLOT(startNuevaAlineacionInicial()));

QState *nuevaSustitucion = new QState();
connect(nuevaSustitucion, SIGNAL(entered()), this, SLOT(startNuevaSustitucion()));

QState *nuevoGol = new QState();
connect(nuevoGol, SIGNAL(entered()), this, SLOT(startNuevoGol()));

QState *nuevoGolMarcado = new QState();
connect(nuevoGolMarcado, SIGNAL(entered()), this, SLOT(startNuevoGolMarcado()));

QState *nuevoGolRecibido = new QState();
connect(nuevoGolRecibido, SIGNAL(entered()), this, SLOT(startNuevoGolRecibido()));

QState *nuevaTarjeta = new QState();
connect(nuevaTarjeta, SIGNAL(entered()), this, SLOT(startNuevaTarjeta()));

QState *nuevaTarjetaRecibida = new QState();
connect(nuevaTarjetaRecibida, SIGNAL(entered()), this, SLOT(startNuevaTarjetaRecibida()));

QState *nuevaTarjetaProvocada = new QState();
connect(nuevaTarjetaProvocada, SIGNAL(entered()), this, SLOT(startNuevaTarjetaProvocada()));

QState *nuevaLesion = new QState();
connect(nuevaLesion, SIGNAL(entered()), this, SLOT(startNuevaLesion()));

QState *nuevaIncidencia = new QState();
connect(nuevaIncidencia, SIGNAL(entered()), this, SLOT(startNuevaIncidencia()));

// Editar Partidos
QState *editarPartido = new QState();
connect(editarPartido, SIGNAL(entered()), this, SLOT(startEditarPartido()));

QState *editarAlineacionInicial = new QState();
connect(editarAlineacionInicial, SIGNAL(entered()), this, SLOT(startEditarAlineacionInicial()));

QState *editarSustitucion = new QState();
connect(editarSustitucion, SIGNAL(entered()), this, SLOT(startEditarSustitucion()));

QState *editarGol = new QState();
connect(editarGol, SIGNAL(entered()), this, SLOT(startEditarGol()));

QState *editarGolMarcado = new QState();
connect(editarGolMarcado, SIGNAL(entered()), this, SLOT(startEditarGolMarcado()));

QState *editarGolRecibido = new QState();
connect(editarGolRecibido, SIGNAL(entered()), this, SLOT(startEditarGolRecibido()));
```

```
    QState *editarTarjeta = new QState();
    connect(editarTarjeta, SIGNAL(entered()), this, SLOT(startEditarTarjeta()));

    QState *editarTarjetaRecibida = new QState();
    connect(editarTarjetaRecibida, SIGNAL(entered()), this, SLOT(startEditarTarjetaRecibida()));

    QState *editarTarjetaProvocada = new QState();
    connect(editarTarjetaProvocada, SIGNAL(entered()), this, SLOT(startEditarTarjetaProvocada()));

    QState *editarLesion = new QState();
    connect(editarLesion, SIGNAL(entered()), this, SLOT(startEditarLesion()));

    QState *editarIncidencia = new QState();
    connect(editarIncidencia, SIGNAL(entered()), this, SLOT(startEditarIncidencia()));


    // Declaración de Transiciones

    ////////////////////////////
    // Desde pantalla Inicial //
    ////////////////////////////

    estadoInicio->addTransition(this->ui->botonInicio_Equipos, SIGNAL(clicked()), estadoEquipos);
    estadoInicio->addTransition(this->ui->botonInicio_MiEquipo, SIGNAL(clicked()), estadoMiEquipo);
    estadoInicio->addTransition(this->ui->botonInicio_Entrenamiento, SIGNAL(clicked()), estadoEntrenamiento);
    estadoInicio->addTransition(this->ui->botonInicio_Competicion, SIGNAL(clicked()), estadoCompeticion);

    ////////////////////////
    // Equipos ////////////
    ////////////////////////
    estadoEquipos->addTransition(this->ui->botonEquipos_Volver, SIGNAL(clicked()), estadoInicio);
    estadoEquipos->addTransition(this->ui->botonEquipos_NuevoEquipo, SIGNAL(clicked()),nuevoEquipo);
    estadoEquipos->addTransition(this->ui->botonEquipos_EditarEquipo, SIGNAL(clicked()), editarEquipo);
    estadoEquipos->addTransition(this->ui->botonEquipos_VerDetallesEquipo, SIGNAL(clicked()),
verDetallesEquipo);

    // Nuevo Equipo
    nuevoEquipo->addTransition(this->ui->botonNuevoEquipo_Cancelar,SIGNAL(clicked()), estadoEquipos);
    nuevoEquipo->addTransition(this->ui->botonNuevoEquipo_Finalizar,SIGNAL(clicked()), estadoEquipos);

    // Editar Equipo
    editarEquipo->addTransition(this->ui->botonEditarEquipo_Cancelar,SIGNAL(clicked()), estadoEquipos);
    editarEquipo->addTransition(this->ui->botonEditarEquipo_Finalizar,SIGNAL(clicked()), estadoEquipos);

    // Ver Detalles Equipo
    verDetallesEquipo->addTransition(this->ui->botonVerDetallesEquipo_Volver,SIGNAL(clicked()), estadoEquipos);
    verDetallesEquipo->addTransition(this->ui->botonVerDetallesEquipo_VerHistorico,SIGNAL(clicked()),
historicoPartidos);

    ////////////////////////
    // Mi Equipo //////////
    ////////////////////////
    estadoMiEquipo->addTransition(this->ui->botonMiEquipo_Volver, SIGNAL(clicked()), estadoInicio);
    estadoMiEquipo->addTransition(this->ui->botonMiEquipo_Jugadores, SIGNAL(clicked()), estadoJugadores);
    estadoMiEquipo->addTransition(this->ui->botonMiEquipo_Palmares, SIGNAL(clicked()), estadoPalmares);
    estadoMiEquipo->addTransition(this->ui->botonMiEquipo_EditarInformacion, SIGNAL(clicked()), infoMiEquipo);
    estadoMiEquipo->addTransition(this->ui->botonMiEquipo_Rendimiento, SIGNAL(clicked()),
rendimientoMiEquipo);

    // Jugadores
    estadoJugadores->addTransition(this->ui->botonJugadores_Volver, SIGNAL(clicked()), estadoMiEquipo);
    estadoJugadores->addTransition(this->ui->botonJugadores_NuevoJugador, SIGNAL(clicked()), nuevoJugador);
    estadoJugadores->addTransition(this->ui->botonJugadores_EditarJugador, SIGNAL(clicked()), editarJugador);
```

```
    estadoJugadores->addTransition(this->ui->botonJugadores_VerDetalles, SIGNAL(clicked()), verDetallesJugador);

    // Nuevo Jugador
    nuevoJugador->addTransition(this->ui->botonNuevoJugador_Cancelar, SIGNAL(clicked()), estadoJugadores);
    nuevoJugador->addTransition(this->ui->botonNuevoJugador_Finalizar, SIGNAL(clicked()), estadoJugadores);

    // Editar Jugador
    editarJugador->addTransition(this->ui->botonEditarJugador_Cancelar, SIGNAL(clicked()), estadoJugadores);
    editarJugador->addTransition(this->ui->botonEditarJugador_Finalizar, SIGNAL(clicked()), estadoJugadores);

    // Ver Detalles Jugador
    verDetallesJugador->addTransition(this->ui->botonVerDetallesJugador_Volver, SIGNAL(clicked()),
estadoJugadores);
    verDetallesJugador->addTransition(this->ui->botonVerDetallesJugador_VerPartidos, SIGNAL(clicked()),
verPartidosJugados);
    verDetallesJugador->addTransition(this->ui->botonVerDetallesJugador_VerMinutos, SIGNAL(clicked()),
verMinutosDisputados);
    verDetallesJugador->addTransition(this->ui->botonVerDetallesJugador_VerGolesMarcados, SIGNAL(clicked()),
verGolesMarcados);
    verDetallesJugador->addTransition(this->ui->botonVerDetallesJugador_VerGolesRecibidos, SIGNAL(clicked()),
verGolesRecibidos);
    verDetallesJugador->addTransition(this->ui->botonVerDetallesJugador_VerTarjetasRecibidas, SIGNAL(clicked()),
verTarjetasRecibidas);
    verDetallesJugador->addTransition(this->ui->botonVerDetallesJugador_VerTarjetasProvocadas,
SIGNAL(clicked()), verTarjetasProvocadas);
    verDetallesJugador->addTransition(this->ui->botonVerDetallesJugador_VerLesiones, SIGNAL(clicked()),
verLesiones);

    // Ver Partidos Jugados
    verPartidosJugados->addTransition(this->ui->botonPartidosJugados_Volver, SIGNAL(clicked()),
verDetallesJugador);

    // Ver Minutos Disputados
    verMinutosDisputados->addTransition(this->ui->botonVerMinutosDisputados_Volver, SIGNAL(clicked()),
verDetallesJugador);

    // Ver Goles Marcados
    verGolesMarcados->addTransition(this->ui->botonGolesMarcados_Volver, SIGNAL(clicked()),
verDetallesJugador);

    // Ver Goles Recibidos
    verGolesRecibidos->addTransition(this->ui->botonGolesRecibidos_Volver, SIGNAL(clicked()),
verDetallesJugador);

    // Ver Tarjetas Recibidas
    verTarjetasRecibidas->addTransition(this->ui->botonTarjetasRecibidas_Volver, SIGNAL(clicked()),
verDetallesJugador);

    // Ver Tarjetas Provocadas
    verTarjetasProvocadas->addTransition(this->ui->botonTarjetasProvocadas_Volver, SIGNAL(clicked()),
verDetallesJugador);

    // Ver Lesiones
    verLesiones->addTransition(this->ui->botonLesiones_Volver, SIGNAL(clicked()), verDetallesJugador);

    // Palmarés
    estadoPalmares->addTransition(this->ui->botonPalmares_Volver, SIGNAL(clicked()), estadoJugadores);
    estadoPalmares->addTransition(this->ui->botonPalmares_NuevoTitulo, SIGNAL(clicked()), nuevoTitulo);
    estadoPalmares->addTransition(this->ui->botonPalmares_EditarTtulo, SIGNAL(clicked()), editarTitulo);

    // Nuevo Título
    nuevoTitulo->addTransition(this->ui->botonNuevoTitulo_Volver, SIGNAL(clicked()), estadoPalmares);
```

```
    // Editar Titulo
    editarTitulo->addTransition(this->ui->botonEditarTitulo_Volver, SIGNAL(clicked()), estadoPalmares);

    // Información Mi Equipo
    infoMiEquipo->addTransition(this->ui->botonInfoMiEquipo_Cancelar, SIGNAL(clicked()), estadoMiEquipo);
    infoMiEquipo->addTransition(this->ui->botonInfoMiEquipo_Finalizar, SIGNAL(clicked()), estadoMiEquipo);

    // Rendimiento Mi Equipo
    rendimientoMiEquipo->addTransition(this->ui->botonRendimientoMiEquipo_Volver, SIGNAL(clicked()),
estadoMiEquipo);
    rendimientoMiEquipo->addTransition(this->ui->botonRendimientoMiEquipo_GolesMarcados, SIGNAL(clicked()),
golesMarcadosPorMinuto);
    rendimientoMiEquipo->addTransition(this->ui->botonRendimientoMiEquipo_GolesEncajados, SIGNAL(clicked()),
golesRecibidosPorMinuto);
    rendimientoMiEquipo->addTransition(this->ui->botonRendimientoMiEquipo_Amonestaciones,
SIGNAL(clicked()), amonestacionesPorMinuto);

    // Goles Marcador según Minuto
    golesMarcadosPorMinuto->addTransition(this->ui->botonGolesMarcadosPorMinuto_Volver, SIGNAL(clicked()),
rendimientoMiEquipo);

    // Goles Recibidos según Minuto
    golesRecibidosPorMinuto->addTransition(this->ui->botonGolesRecibidosPorMinuto_Volver, SIGNAL(clicked()),
rendimientoMiEquipo);

    // Amonestaciones según Minuto
    amonestacionesPorMinuto->addTransition(this->ui->botonAmonestacionesPorMinuto_Volver,
SIGNAL(clicked()), rendimientoMiEquipo);

    //////////////////////////
    // Entrenamiento //////////
    //////////////////////////
    estadoEntrenamiento->addTransition(this->ui->botonEntrenamiento_Volver, SIGNAL(clicked()), estadoInicio);
    estadoEntrenamiento->addTransition(this->ui->botonEntrenamiento_NuevoEntrenamiento, SIGNAL(clicked()),
nuevoEntrenamiento);
    estadoEntrenamiento->addTransition(this->ui->botonEntrenamiento_VerHisotrico, SIGNAL(clicked()),
historicoEntrenamientos);

    // Nuevo Entrenamiento
    nuevoEntrenamiento->addTransition(this->ui->botonNuevoEntrenamiento_Volver, SIGNAL(clicked()),
estadoEntrenamiento);
    nuevoEntrenamiento->addTransition(this->ui->botonNuevoEntrenamiento_Finalizar, SIGNAL(clicked()),
estadoEntrenamiento);
    nuevoEntrenamiento->addTransition(this->ui->botonNuevoEntrenamiento_NuevoEjercicio, SIGNAL(clicked()),
nuevoEjercicio);
    nuevoEntrenamiento->addTransition(this->ui->botonNuevoEntrenamiento_EditarEjercicio, SIGNAL(clicked()),
editarEjercicio);
    nuevoEntrenamiento->addTransition(this->ui->botonNuevoEntrenamiento_ComentarEjercicio,
SIGNAL(clicked()), comentarEjercicio);
    nuevoEntrenamiento->addTransition(this->ui->botonNuevoEntrenamiento_Incidencias, SIGNAL(clicked()),
incidenciasEjercicio);

    // Nuevo Ejercicio
    nuevoEjercicio->addTransition(this->ui->botonNuevoEjercicio_Cancelar, SIGNAL(clicked()),
nuevoEntrenamiento);
    nuevoEjercicio->addTransition(this->ui->botonNuevoEjercicio_Finalizar, SIGNAL(clicked()),
nuevoEntrenamiento);

    // Editar Ejercicio
    editarEjercicio->addTransition(this->ui->botonEditarEjercicio_Cancelar, SIGNAL(clicked()), nuevoEntrenamiento);
    editarEjercicio->addTransition(this->ui->botonEditarEjercicio_Finalizar, SIGNAL(clicked()), nuevoEntrenamiento);
```

```
 // Comentar Ejercicio
   comentarEjercicio->addTransition(this->ui->botonComentarEjercicio_Cancelar, SIGNAL(clicked()),
nuevoEntrenamiento);
   comentarEjercicio->addTransition(this->ui->botonComentarEjercicio_Finalizar, SIGNAL(clicked()),
nuevoEntrenamiento);

   // Incidencias Ejercicio
   incidenciasEjercicio->addTransition(this->ui->botonIncidenciasEjercicio_Cancelar, SIGNAL(clicked()),
nuevoEntrenamiento);
   incidenciasEjercicio->addTransition(this->ui->botonIncidenciasEjercicio_Finalizar, SIGNAL(clicked()),
nuevoEntrenamiento);

   // Histórico Entrenamientos
   historicoEntrenamientos->addTransition(this->ui->botonHistoricoEntrenamientos_Volver, SIGNAL(clicked()),
estadoEntrenamiento);

   ////////////////////////
   // Competición ///////////
   ////////////////////////
   estadoCompeticion->addTransition(this->ui->botonCompeticion_Volver, SIGNAL(clicked()), estadoInicio);
   estadoCompeticion->addTransition(this->ui->botonCompeticion_NuevaCompeticion, SIGNAL(clicked()),
nuevaCompeticion);
   estadoCompeticion->addTransition(this->ui->botonCompeticion_Partidos, SIGNAL(clicked()), estadoPartidos);

   //Nueva Competicion
   nuevaCompeticion->addTransition(this->ui->botonNuevaCompeticion_Cancelar, SIGNAL(clicked()),
estadoCompeticion);
   nuevaCompeticion->addTransition(this->ui->botonNuevaCompeticion_Finalizar, SIGNAL(clicked()),
estadoCompeticion);

   // Partidos
   estadoPartidos->addTransition(this->ui->botonPartidos_Volver, SIGNAL(clicked()), estadoCompeticion);
   estadoPartidos->addTransition(this->ui->botonPartidos_VerHistorico, SIGNAL(clicked()), historicoPartidos);
   estadoPartidos->addTransition(this->ui->botonPartidos_NuevoPartido, SIGNAL(clicked()), nuevoPartido);
   estadoPartidos->addTransition(this->ui->botonPartidos_EditarPartido, SIGNAL(clicked()), editarPartido);

   // Ver Historico Partidos
   historicoPartidos->addTransition(this->ui->botonVerHistoricoPartidos_Volver, SIGNAL(clicked()),
estadoPartidos);

   // Nuevo Partido
   nuevoPartido->addTransition(this->ui->botonNuevoPartido_Cancelar, SIGNAL(clicked()), estadoPartidos);
   nuevoPartido->addTransition(this->ui->botonNuevoPartido_Finalizar, SIGNAL(clicked()), estadoPartidos);
   nuevoPartido->addTransition(this->ui->botonNuevoPartido_Alineacion, SIGNAL(clicked()),
nuevaAlineacionInicial);
   nuevoPartido->addTransition(this->ui->botonNuevoPartido_Sustituciones, SIGNAL(clicked()), nuevaSustitucion);
   nuevoPartido->addTransition(this->ui->botonNuevoPartido_Goles, SIGNAL(clicked()), nuevoGol);
   nuevoPartido->addTransition(this->ui->botonNuevoPartido_Tarjetas, SIGNAL(clicked()), nuevaTarjeta);
   nuevoPartido->addTransition(this->ui->botonNuevoPartido_Lesiones, SIGNAL(clicked()), nuevaLesion);
   nuevoPartido->addTransition(this->ui->botonNuevoPartido_Incidencias, SIGNAL(clicked()), nuevaIncidencia);

   // Nueva Alineación Inicial
   nuevaAlineacionInicial->addTransition(this->ui->botonAlineacionInicial_Cancelar, SIGNAL(clicked()),
nuevoPartido);
   nuevaAlineacionInicial->addTransition(this->ui->botonAlineacionInicial_Finalizar, SIGNAL(clicked()),
nuevoPartido);

   // Nueva Sustitucion
   nuevaSustitucion->addTransition(this->ui->botonNuevaSustitucion_Cancelar, SIGNAL(clicked()), nuevoPartido);
   nuevaSustitucion->addTransition(this->ui->botonNuevaSustitucion_Finalizar, SIGNAL(clicked()), nuevoPartido);
```

```
    // Nuevo Gol
    nuevoGol->addTransition(this->ui->botonNuevoGol_Volver, SIGNAL(clicked()), nuevoPartido);
    nuevoGol->addTransition(this->ui->botonNuevoGol_GolAnotado, SIGNAL(clicked()), nuevoGolMarcado);
    nuevoGol->addTransition(this->ui->botonNuevoGol_GolEncajado, SIGNAL(clicked()), nuevoGolRecibido);

    // Nuevo Gol Marcado
    nuevoGolMarcado->addTransition(this->ui->botonNuevoGolMarcado_Cancelar, SIGNAL(clicked()), nuevoGol);
    nuevoGolMarcado->addTransition(this->ui->botonNuevoGolMarcado_Finalizar, SIGNAL(clicked()), nuevoGol);

    // Nuevo Gol Recibido
    nuevoGolRecibido->addTransition(this->ui->botonNuevoGolRecibido_Cancelar, SIGNAL(clicked()), nuevoGol);
    nuevoGolRecibido->addTransition(this->ui->botonNuevoGolRecibido_Finalizar, SIGNAL(clicked()), nuevoGol);

    // Nueva Tarjeta
    nuevaTarjeta->addTransition(this->ui->botonNuevaTarjeta_Volver, SIGNAL(clicked()), nuevoPartido);
    nuevaTarjeta->addTransition(this->ui->botonNuevaTarjeta_TarjetasRecibidas, SIGNAL(clicked()),
nuevaTarjetaRecibida);
    nuevaTarjeta->addTransition(this->ui->botonNuevaTarjeta_TarjetasProvocadas, SIGNAL(clicked()),
nuevaTarjetaProvocada);

    // Nueva Tarjeta Recibida
    nuevaTarjetaRecibida->addTransition(this->ui->botonNuevaTarjetaRecibida_Cancelar, SIGNAL(clicked()),
nuevaTarjeta);
    nuevaTarjetaRecibida->addTransition(this->ui->botonNuevaTarjetaRecibida_Finalizar, SIGNAL(clicked()),
nuevaTarjeta);

    // Nueva Tarjeta Provocada
    nuevaTarjetaProvocada->addTransition(this->ui->botonNuevaTarjetaProvocada_Cancelar, SIGNAL(clicked()),
nuevaTarjeta);
    nuevaTarjetaProvocada->addTransition(this->ui->botonNuevaTarjetaProvocada_Finalizar, SIGNAL(clicked()),
nuevaTarjeta);

    // Nueva Lesión
    nuevaLesion->addTransition(this->ui->botonNuevaLesion_Cancelar, SIGNAL(clicked()), nuevoPartido);
    nuevaLesion->addTransition(this->ui->botonNuevaLesion_Finalizar, SIGNAL(clicked()), nuevoPartido);

    // Nueva Incidencia
    nuevaIncidencia->addTransition(this->ui->botonNuevaIncidencia_Cancelar, SIGNAL(clicked()), nuevoPartido);
    nuevaIncidencia->addTransition(this->ui->botonNuevaIncidencia_Finalizar, SIGNAL(clicked()), nuevoPartido);

    // Editar Partido
    editarPartido->addTransition(this->ui->botonEditarPartido_Cancelar, SIGNAL(clicked()), estadoPartidos);
    editarPartido->addTransition(this->ui->botonEditarPartido_Finalizar, SIGNAL(clicked()), estadoPartidos);
    editarPartido->addTransition(this->ui->botonEditarPartido_AlineacionInicial, SIGNAL(clicked()),
editarAlineacionInicial);
    editarPartido->addTransition(this->ui->botonEditarPartido_Sustituciones, SIGNAL(clicked()), editarSustitucion);
    editarPartido->addTransition(this->ui->botonEditarPartido_Goles, SIGNAL(clicked()), editarGol);
    editarPartido->addTransition(this->ui->botonEditarPartido_Tarjetas, SIGNAL(clicked()), editarTarjeta);
    editarPartido->addTransition(this->ui->botonEditarPartido_Lesiones, SIGNAL(clicked()), editarLesion);
    editarPartido->addTransition(this->ui->botonEditarPartido_Incidencias, SIGNAL(clicked()), editarIncidencia);

    // Editar Alineación Inicial
    editarAlineacionInicial->addTransition(this->ui->botonEditarAlineacionInicial_Cancelar, SIGNAL(clicked()),
editarPartido);
    editarAlineacionInicial->addTransition(this->ui->botonEditarAlineacionInicial_Finalizar, SIGNAL(clicked()),
editarPartido);

    // Editar Sustitucion
    editarSustitucion->addTransition(this->ui->botonEditarSustitucion_Cancelar, SIGNAL(clicked()), editarPartido);
    editarSustitucion->addTransition(this->ui->botonEditarSustitucion_Finalizar, SIGNAL(clicked()), editarPartido);

    // Editar Gol
    editarGol->addTransition(this->ui->botonEditarGol_Volver, SIGNAL(clicked()), editarPartido);
```

```
    editarGol->addTransition(this->ui->botonEditarGol_GolesMarcados, SIGNAL(clicked()), editarGolMarcado);
    editarGol->addTransition(this->ui->botonEditarGol_GolesRecibidos, SIGNAL(clicked()), editarGolRecibido);

    // Editar Gol Marcado
    editarGolMarcado->addTransition(this->ui->botonEditarGolMarcado_Cancelar, SIGNAL(clicked()), editarGol);
    editarGolMarcado->addTransition(this->ui->botonEditarGolMarcado_Finalizar, SIGNAL(clicked()), editarGol);

    // Editar Gol Recibido
    editarGolRecibido->addTransition(this->ui->botonEditarGolRecibido_Cancelar, SIGNAL(clicked()), editarGol);
    editarGolRecibido->addTransition(this->ui->botonEditarGolRecibido_Finalizar, SIGNAL(clicked()), editarGol);

    // Editar Tarjeta
    editarTarjeta->addTransition(this->ui->botonEditarTarjeta_Volver, SIGNAL(clicked()), editarPartido);
    editarTarjeta->addTransition(this->ui->botonEditarTarjeta_TarjetaRecibida, SIGNAL(clicked()),
editarTarjetaRecibida);
    editarTarjeta->addTransition(this->ui->botonEditarTarjeta_TarjetaProvocada, SIGNAL(clicked()),
editarTarjetaProvocada);

    // Editar Tarjeta Recibida
    editarTarjetaRecibida->addTransition(this->ui->botonEditarTarjetaRecibida_Cancelar, SIGNAL(clicked()),
editarTarjeta);
    editarTarjetaRecibida->addTransition(this->ui->botonEditarTarjetaRecibida_Finalizar, SIGNAL(clicked()),
editarTarjeta);

    // Editar Tarjeta Provocada
    editarTarjetaProvocada->addTransition(this->ui->botonEditarTarjetaProvocada_Cancelar, SIGNAL(clicked()),
editarTarjeta);
    editarTarjetaProvocada->addTransition(this->ui->botonEditarTarjetaProvocada_Finalizar, SIGNAL(clicked()),
editarTarjeta);

    // Editar Lesión
    editarLesion->addTransition(this->ui->botonEditarLesion_Cancelar, SIGNAL(clicked()), editarPartido);
    editarLesion->addTransition(this->ui->botonEditarLesion_Finalizar, SIGNAL(clicked()), editarPartido);

    // Editar Incidencia
    editarIncidencia->addTransition(this->ui->botonEditarIncidencia_Cancelar, SIGNAL(clicked()), editarPartido);
    editarIncidencia->addTransition(this->ui->botonEditarIncidencia_Finalizar, SIGNAL(clicked()), editarPartido);

    ////////////////////////////////////////////////////
    // Añadimos los estados a la máquina declarada ////////
    ////////////////////////////////////////////////////
    machine->addState(estadoInicio);

    machine->addState(estadoEquipos);
    machine->addState(nuevoEquipo);
    machine->addState(editarEquipo);
    machine->addState(verDetallesEquipo);

    machine->addState(estadoMiEquipo);
    machine->addState(estadoJugadores);
    machine->addState(nuevoJugador);
    machine->addState(editarJugador);
    machine->addState(verDetallesJugador);
    machine->addState(verPartidosJugados);
    machine->addState(verMinutosDisputados);
    machine->addState(verGolesMarcados);
    machine->addState(verGolesRecibidos);
    machine->addState(verTarjetasRecibidas);
    machine->addState(verTarjetasProvocadas);
    machine->addState(verLesiones);
    machine->addState(estadoPalmares);
    machine->addState(nuevoTitulo);
    machine->addState(editarTitulo);
```

```
    machine->addState(infoMiEquipo);
    machine->addState(rendimientoMiEquipo);
    machine->addState(golesMarcadosPorMinuto);
    machine->addState(golesRecibidosPorMinuto);
    machine->addState(amonestacionesPorMinuto);

    machine->addState(estadoEntrenamiento);
    machine->addState(nuevoEntrenamiento);
    machine->addState(nuevoEjercicio);
    machine->addState(editarEjercicio);
    machine->addState(comentarEjercicio);
    machine->addState(incidenciasEjercicio);
    machine->addState(historicoEntrenamientos);

    machine->addState(estadoCompeticion);
    machine->addState(nuevaCompeticion);
    machine->addState(estadoPartidos);
    machine->addState(historicoPartidos);
    machine->addState(nuevoPartido);
    machine->addState(nuevaAlineacionInicial);
    machine->addState(nuevaSustitucion);
    machine->addState(nuevoGol);
    machine->addState(nuevoGolMarcado);
    machine->addState(nuevoGolRecibido);
    machine->addState(nuevaTarjeta);
    machine->addState(nuevaTarjetaRecibida);
    machine->addState(nuevaTarjetaProvocada);
    machine->addState(nuevaLesion);
    machine->addState(nuevaIncidencia);
    machine->addState(editarPartido);
    machine->addState(editarAlineacionInicial);
    machine->addState(editarSustitucion);
    machine->addState(editarGol);
    machine->addState(editarGolMarcado);
    machine->addState(editarGolRecibido);
    machine->addState(editarTarjeta);
    machine->addState(editarTarjetaRecibida);
    machine->addState(editarTarjetaProvocada);
    machine->addState(editarLesion);
    machine->addState(editarIncidencia);

    // Ajustamos el estado Inicial de la Máquina de Estados "machine" y la iniciamos
    machine->setInitialState(estadoInicio);
    machine->start();

}
```

- Tabla 6 -

El método implementado para construir la máquina de estados mostrado en la Tabla 6, se invoca desde la función principal del programa, de modo que la ejecución del mismo está dirigida por esta función. En lo sucesivo, y para concluir con este apartado, mostraremos en un ejemplo la secuencia que sigue la aplicación para realizar transiciones entre estados.

Al iniciar la aplicación, se invoca al método ConstruirMaquinaEstados() y éste crea la máquina y determina que el estado inicial de la misma es estadoincio (machine->setInitialState(estadoInicio)). Al entrar en este estado, se determina que la acción que responde a este evento es la que marque el slot startInicio (connect(estadoInicio, SIGNAL(entered()), this, SLOT (startInicio()))). El slot startInicio() indica que se ponga el atributo state a 0 y que se invoque a la funcón que gestiona los cambios de estado de la máquina (state = 0; stateChanged(state)). En este punto, el método stateChanged, en primer lugar, hace invisibles todos los QFrames, para, posteriormente, verificar cual es el estado pasado como parámetro a

la función (0 en este caso), y el QFrame correspondiente a ese estado se hace visible al usuario. Este proceso es el seguido cada vez que la máquina de estados alcanza cualquiera de los estados que la forman, siendo fundamental pues, el uso del Qt State Machine Framework para el correcto funcionamiento de la aplicación.

Nótese que el código presentado en este apartado variará para añadir funcionalidad a la aplicación, pero el esqueleto de la máquina de estados que rige su comportamiento será el mismo.

3.2. BASE DE DATOS

La gestión de información que la aplicación maneja procede de un origen de datos concreto, una base de datos previamente creada.

De la documentación consultada acerca del uso de bases de datos en aplicaciones Qt y la gestión y manejo sobre las mismas desde el código fuente, averiguamos que hay una serie de bases de datos soportadas por el sistema que son las mostradas en la tabla 7.

Nombre del Driver	Descripción
QDB2	IBM DB2 (versión 7.1 y superiores)
QIBASE	Borland InterBase
QMYSQL	MySQL
QOCI	Oracle Call Interface Driver
QODBC	Open DataBase Connectivity – Microsoft SQL Server
QPSQL	PostgreSQL (versión 7.3 y superiores)
QSQLITE2	SQLITE versión 2
QSQLITE	SQLITE versión 3
QTDS	Sybase Adaptive Server

- Tabla 7 -

Debido a la posibilidad de tener una base de datos “in-process”, lo cual significa poder trabajar con la base de datos sin tener un servidor de bases de datos (como por ejemplo sucede en MySQL), para el desarrollo del proyecto, nos decantamos por el uso de bases de datos SQLite. Además, cabe destacar, que desde el punto de vista del manejo de la fuente de información, este tipo de bases de datos son sensibles y presentan restricciones en el caso de múltiples transacciones por parte de múltiples usuarios, y en este caso este contratiempo no nos va a afectar debido a que la base de datos va a ser para una aplicación monousuario.

Destacar de las bases de datos SQLite en primer lugar, que no tiene unos tipos definidos como pudiera tenerlos otro tipo de bases de datos (en lugar de CHAR, VARCHAR(n), etc., tiene campos como PK INTEGER, Text, etc.) y por eso, Qt interpreta los campos como cadenas de texto o strings. Veamos en la tabla 8 el código SQL que da lugar a la base de datos que gestiona los datos del sistema, generado automáticamente por el programa SQLite Man, en el cual se soporta la versión SQLite 3.5.4:

```
CREATE TABLE "Equipo" (
   "idEquipo" INTEGER PRIMARY KEY NOT NULL,
   "nombre" TEXT NOT NULL,
   "dirEstadio" TEXT NOT NULL,
   "poblacion" TEXT NOT NULL,
   "telefono" TEXT NOT NULL,
   "email" TEXT NOT NULL,
   "equipajes" TEXT NOT NULL
);
```

```
CREATE TABLE "Incidencia" (
   "idIncidencia" INTEGER PRIMARY KEY NOT
NULL,
   "partido" INTEGER NOT NULL,
   "incidencia" TEXT NOT NULL
);
CREATE TABLE "Partido" (
   "idPartido" INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL,
```

```
CREATE TABLE "MiEquipo" (
   "idMiEquipo" INTEGER PRIMARY KEY NOT NULL,
   "nombre" TEXT NOT NULL,
   "dirEstadio" TEXT NOT NULL,
   "poblacion" TEXT NOT NULL,
   "telefono" TEXT NOT NULL,
   "email" TEXT NOT NULL,
   "equipajes" TEXT NOT NULL
);
CREATE TABLE "Palmares" (
   "idTitulo" INTEGER PRIMARY KEY NOT NULL,
   "nombreCompeticion" TEXT NOT NULL,
   "tempInicio" TEXT NOT NULL,
   "tempFin" TEXT NOT NULL,
   "posicion" TEXT NOT NULL
);
CREATE TABLE "Alineacion" (
   "idAlineacion" INTEGER PRIMARY KEY NOT NULL,
   "partido" INTEGER NOT NULL,
   "jugador1" TEXT NOT NULL,
   "jugador2" TEXT NOT NULL,
   "jugador3" TEXT NOT NULL,
   "jugador4" TEXT NOT NULL,
   "jugador5" TEXT NOT NULL,
   "jugador6" TEXT NOT NULL,
   "jugador7" TEXT NOT NULL,
   "jugador8" TEXT NOT NULL,
   "jugador9" TEXT NOT NULL,
   "jugador10" TEXT NOT NULL,
   "jugador11" TEXT NOT NULL
);

CREATE TABLE "Sustitucion" (
   "idSustitucion" INTEGER PRIMARY KEY NOT NULL,
   "partido" INTEGER NOT NULL,
   "jugadorIn" TEXT NOT NULL,
   "jugadorOut" TEXT NOT NULL,
   "minuto" INTEGER NOT NULL
);
CREATE TABLE "GolRecibido" (
   "idGolRecibido" INTEGER PRIMARY KEY NOT NULL,
   "partido" INTEGER NOT NULL,
   "minuto" INTEGER NOT NULL,
   "comentarios" TEXT NOT NULL,
   "jugador1" TEXT NOT NULL,
   "jugador2" TEXT NOT NULL,
   "jugador3" TEXT NOT NULL,
   "jugador4" TEXT NOT NULL,
   "jugador5" TEXT NOT NULL,
   "jugador6" TEXT NOT NULL,
   "jugador7" TEXT NOT NULL,
   "jugador8" TEXT NOT NULL,
   "jugador9" TEXT NOT NULL,
   "jugador10" TEXT NOT NULL,
   "jugador11" TEXT NOT NULL,
   "resultadoProv" TEXT NOT NULL
);
CREATE TABLE "TarjetaRecibida" (
   "idTarjetaRecibida" INTEGER PRIMARY KEY NOT NULL,
   "partido" INTEGER NOT NULL,
   "minuto" INTEGER NOT NULL,
   "jugador" TEXT NOT NULL,
   "tarjeta" TEXT NOT NULL,
   "competicion" TEXT NOT NULL,
   "rival" TEXT NOT NULL,
   "campo" TEXT NOT NULL,
   "dia" TEXT NOT NULL,
   "mes" TEXT NOT NULL,
   "anyo" TEXT NOT NULL,
   "resultado" TEXT NOT NULL
);
CREATE TABLE sqlite_sequence(name,seq);
CREATE TABLE "Entrenamiento" (
   "idEntrenamiento" INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL,
   "dia" TEXT NOT NULL,
   "mes" TEXT NOT NULL,
   "anyo" TEXT NOT NULL
);
CREATE TABLE "Ejercicios" (
   "idEjercicio" INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL,
   "descripcion" TEXT NOT NULL,
   "duracion" TEXT NOT NULL,
   "comentario" TEXT NOT NULL,
   "incidencia" TEXT NOT NULL,
   "idEntrenamiento" TEXT NOT NULL
);
CREATE TABLE "JugadorPartido" (
   "idJugadorPartido" INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL,
   "idPartido" INTEGER NOT NULL,
   "idJugador" INTEGER NOT NULL,
   "minutos" INTEGER NOT NULL
);
CREATE TABLE "Competicion" (
   "idCompeticion" INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL,
   "nombre" TEXT NOT NULL,
   "tempInicio" TEXT NOT NULL,
   "tempFin" TEXT NOT NULL,
   "normativa" TEXT NOT NULL
);
CREATE TABLE "Jugador" (
   "idJugador" INTEGER PRIMARY KEY
AUTOINCREMENT NOT NULL,
   "nombre" TEXT NOT NULL,
   "dorsal" TEXT NOT NULL,
   "demarcacion" TEXT NOT NULL,
   "direccion" TEXT NOT NULL,
   "poblacion" TEXT NOT NULL,
   "cp" TEXT NOT NULL,
   "telefono" TEXT NOT NULL,
   "diaNac" TEXT NOT NULL,
   "mesNac" TEXT NOT NULL,
   "anyoNac" TEXT NOT NULL,
   "anyoIngreso" TEXT NOT NULL,
   "partidosJugados" INTEGER NOT NULL,
   "minutosDisputados" INTEGER NOT NULL,
   "golesMarcados" INTEGER NOT NULL,
   "golesRecibidos" INTEGER NOT NULL,
   "tarjetasRecibidas" INTEGER NOT NULL,
   "tarjetasProvocadas" INTEGER NOT NULL
);
CREATE TABLE golmarcado (
   "idGolMarcado" INTEGER PRIMARY KEY,
```

```
   "comentarios" TEXT NOT NULL
);
CREATE TABLE "TarjetaProvocada" (
   "idTarjetaProvocada" INTEGER PRIMARY KEY NOT NULL,
   "partido" INTEGER NOT NULL,
   "minuto" INTEGER NOT NULL,
   "jugador" TEXT NOT NULL,
   "tarjeta" TEXT NOT NULL,
   "comentarios" TEXT NOT NULL
);
CREATE TABLE "Lesion" (
   "idLesion" INTEGER PRIMARY KEY NOT NULL,
   "partido" INTEGER NOT NULL,
   "jugador" TEXT NOT NULL,
   "minuto" INTEGER NOT NULL,
   "descripcion" TEXT NOT NULL
);
```

```
   "partido" TEXT,
   "minuto" INTEGER,
   "goleador" TEXT,
   "comentarios" TEXT,
   "resultadoProv" TEXT
);
```

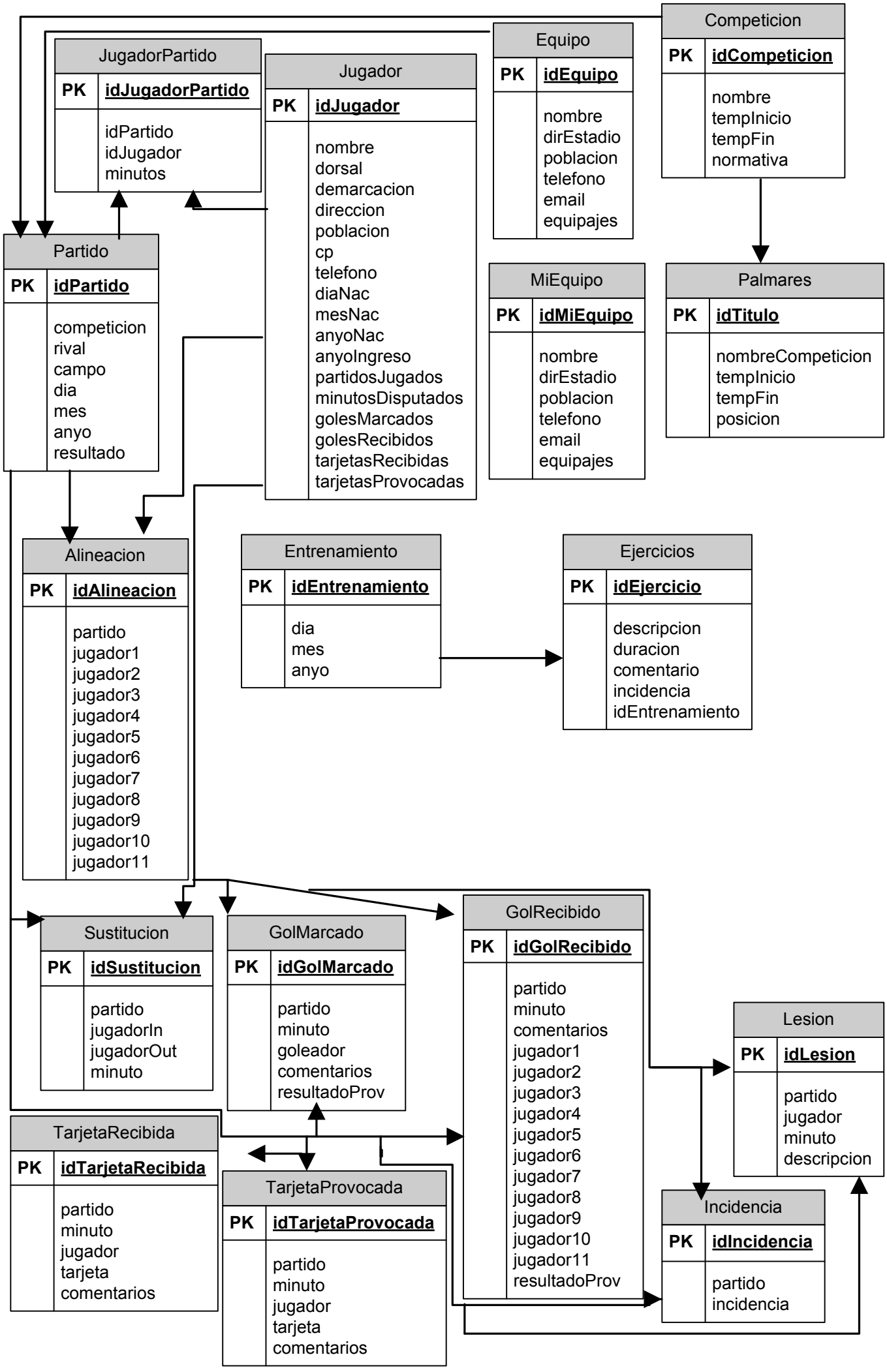
- Tabla 8 -

De la tabla 8, podemos extraer la siguiente conclusión a la cual debemos llegar. SQLite no implemente integridad referencial, esto significa que la gestión de claves ajenas, borrados en cascada, actualizaciones a nulo, etc. deben ser gestionadas con múltiples querys desde la lógica del programa, tarea, por tanto, del programador, que deberá estar atento a las acciones a realizar cada vez que una acción sea efectuada sobre la base de datos, especialmente, cuando se actualicen o se eliminen registros de las tablas de la base de datos.

Por otra parte, y aunque en apartados siguientes se explicará con mayor grado de detalles, la conexión a la base de datos desde el programa se efectúa de una manera en la cual pueden suceder dos cosas. La primera de ellas, es que, tras introducir el nombre de la base de datos a abrir y a la que conectarse, ésta no exista y se cree una, que estaría en blanco. En segundo lugar, y opción que se emplea en esta aplicación, es que al introducir el nombre de la base de datos en el código, ésta ya exista en el directorio correspondiente y, por tanto, el programa la abre y empieza a trabajar con los datos que esta misma contiene. Si esto no fuera así, las tablas se deberían crear tras la conexión ya que la base de datos estaría vacía, y no tiene sentido en el caso de querer almacenar información a largo plazo. Se contempla, de todos modos, la posibilidad de crear una base de datos nueva, para lo cual, se deberá introducir el código SQL de creación de tablas presentado en la tabla 8, para que, entonces sí, el usuario comience a trabajar con una base de datos vacía, y obviamente, adecuada a la aplicación si así lo desea.

Dicho todo lo anterior, es de especial interés observar cuales son las tablas de la base de datos y las relaciones y referencias entre ellas. Así pues, de una tabla podrá salir una flecha que llega a otra tabla, lo que significa que la tabla donde llega la fecha tiene algún campo (fácilmente deducible) que es clave ajena que hace referencia a la clave primaria de la tabla que hace de origen de la flecha. Este esquema tiene una vital importancia presentarlo en este punto ya que sobre él se apoya la integridad referencial mantenida por el desarrollador desde la lógica del programa y mediante código no gestionado por el Sistema de Gestión de Bases de Datos (SGBD). Asimismo, y para terminar, añadir que cualquier tipo de transacción que conlleve acciones transparentes al usuario y que pueden afectar al contenido de la base de datos, como por ejemplo un borrado en cascada que elimine gran cantidad de datos, será comunicado al usuario para que éste tome conciencia de lo que va a suceder y que decida si desea continuar o no.

Las siguientes tablas han sido obtenidas a partir del estudio y el análisis de las clases y las relaciones, así como de las multiplicidades de las mismas en el diagrama de clases presentado en el punto 2.1. de este documento.


Competicion
PK idCompeticion
nombre
tempInicio
tempFin
normativa
Equipo
PK idEquipo
nombre
dirEstadio
poblacion
telefono
email
equipajes
JugadorPartido
PK idJugadorPartido
idPartido
idJugador
minutos
Jugador
PK idJugador
nombre
dorsal
demarcacion
direccion
poblacion
cp
telefono
diaNac
mesNac
anyoNac
anyoIngreso
partidosJugados
minutosDisputados
golesMarcados
golesRecibidos
tarjetasRecibidas
tarjetasProvocadas
Partido
PK idPartido
competicion
rival
campo
dia
mes
anyo
resultado
MiEquipo
PK idMiEquipo
nombre
dirEstadio
poblacion
telefono
email
equipajes
Palmares
PK idTitulo
nombreCompeticion
tempInicio
tempFin
posicion
Alineacion
PK idAlineacion
partido
jugador1
jugador2
jugador3
jugador4
jugador5
jugador6
jugador7
jugador8
jugador9
jugador10
jugador11
Entrenamiento
PK idEntrenamiento
dia
mes
anyo
Ejercicios
PK idEjercicio
descripcion
duracion
comentario
incidencia
idEntrenamiento
Sustitucion
PK idSustitucion
partido
jugadorIn
jugadorOut
minuto
GolMarcado
PK idGolMarcado
partido
minuto
goleador
comentarios
resultadoProv
GolRecibido
PK idGolRecibido
partido
minuto
comentarios
jugador1
jugador2
jugador3
jugador4
jugador5
jugador6
jugador7
jugador8
jugador9
jugador10
jugador11
resultadoProv
Lesion
PK idLesion
partido
jugador
minuto
descripcion
TarjetaRecibida
PK idTarjetaRecibida
partido
minuto
jugador
tarjeta
comentarios
TarjetaProvocada
PK idTarjetaProvocada
partido
minuto
jugador
tarjeta
comentarios
Incidencia
PK idIncidencia
partido
incidencia

3.3. MÁQUINA DE ESTADOS FINAL

Antes de adentrarnos en detalles específicos del desarrollo de los métodos que cubrirán los requisitos previamente estudiados para la aplicación, es conveniente hacer hincapié en una serie de eventualidades acontecidas a lo largo del proceso de programación de “Soccer Manager 2010”.

En el proceso de diseño se presupuso que el contenido de las celdas era accesible mediante el clic sobre las mismas. De este modo, si por ejemplo, en una celda se encuentra el nombre de un jugador, el usuario al clicar sobre esa celda podría lanzar a ejecución cualquier procesamiento que tuviese como entrada dicha cadena de caracteres. Sin embargo, esta presuposición es errónea debido a que las tablas que en la aplicación se emplean son del tipo QTableView y prohíben este procesamiento (al contrario del comportamiento que presentan las tablas del tipo QTableWidget). El problema pues, surge en el tipo de tabla escogida para el desarrollo de la funcionalidad; elección por otra parte obligada, ya que el único tipo de tabla que soporta la programación tomando como origen la información de una base de datos es el comentado.

Así pues, presentado este problema, nos encontramos en la tesitura de tener que añadir pantallas nuevas a la aplicación, lo cual representa también un incremento en el número de estados de la máquina. La siguientes figuras muestran la corrección del problema presentado haciendo uso de una ventana intermedia en los casos necesarios.

En primer lugar, tenemos la pantalla en la cual se encuentra la tabla, y de la cual no podemos seleccionar ninguna fila y utilizar su contenido haciendo clic en ella. Por tanto, en este caso, en vez de pulsar una fila y luego “Editar Jugador”, lo que haremos será simplemente pulsar “Editar Jugador”.

- Figura 4-

Tras pulsar el citado botón, llegaremos a una pantalla intermedia no diseñada inicialmente por la presuposición errónea previamente explicada, en la cual deberemos seleccionar el jugador cuya información queremos ver y editar. Todos los jugadores presentes en la tabla se encontrarán identificados por su nombre en el ComboBox de la pantalla. Seleccionamos el deseado y pulsamos continuar. Figura 5.

-Figura 5-

Por último, vemos la información relativa al jugador que en la pantalla anterior escogimos y podemos proceder a editarla. Al finalizar pulsaremos “Editar”. Ver Figura 6.

- Figura 6 –

Lo anteriormente expuesto se hace extensivo a todos los módulos del programa que en alguna de sus funcionalidades hayan necesitado la presente corrección. En todos los casos el problema se ha abordado y se ha resuelto de esta manera.

Por otra parte, hay que destacar también como novedad respecto al diseño inicial la inclusión de una serie de estados nuevos en la máquina. La aparición de estos estados tiene su fundamento en la necesidad de que haya determinados momentos y puntos en la aplicación que cuando sean alcanzados ejecuten una determinada transacción en la base de datos. En el diseño inicial sólo se contemplaban los estados que llevaran a cabo la dirección de la interfaz gráfica en función de la máquina de estados, sin tener en cuenta las transacciones. Destacar de ellos que en el código fuente todos estos estados son de la forma [estado*] y se encuentran todos declarados en la parte que podemos llamar final del tramo de código destinado a la declaración de estados y sus acciones cuando son alcanzados. Asimismo, hacer notar de estos estados, que, a diferencia de los demás, el slot que ejecutan cuando son alcanzados no es de la forma [start*()]. Veamos la declaración de estos estados en la tabla 9.

```
////////////////////////////////////////////////////////////////////////////////////////////
// Estados derivados de la gestión de datos. No indicados en el diagrama de la memoria ////
////////////////////////////////////////////////////////////////////////////////////////////

  // Equipos
  QState *estadoEquipoInsertado = new QState();
  connect(estadoEquipoInsertado, SIGNAL(entered()), this, SLOT(equipoInsertado()));

  QState *estadoEquipoEliminado = new QState();
  connect(estadoEquipoEliminado, SIGNAL(entered()), this, SLOT(equipoEliminado()));

  QState *estadoEquipoEditado = new QState();
  connect(estadoEquipoEditado, SIGNAL(entered()), this, SLOT(equipoEditado()));

  // Entrenamientos y Ejercicios
  QState *estadoEntrenamientoInsertado = new QState();
  connect(estadoEntrenamientoInsertado, SIGNAL(entered()), this, SLOT(entrenamientoInsertado()));

  QState *estadoEjercicioInsertado = new QState();
  connect(estadoEjercicioInsertado, SIGNAL(entered()), this, SLOT(ejercicioInsertado()));

  QState *estadoEjercicioEditado = new QState();
  connect(estadoEjercicioEditado, SIGNAL(entered()), this, SLOT(ejercicioEditado()));

  QState *estadoEjercicioEliminado = new QState();
  connect(estadoEjercicioEliminado, SIGNAL(entered()), this, SLOT(ejercicioEliminado()));

  QState *estadoEjercicioComentado = new QState();
  connect(estadoEjercicioComentado, SIGNAL(entered()), this, SLOT(ejercicioComentado()));

  QState *estadoIncidenciaEjercicioInsertada = new QState();
  connect(estadoIncidenciaEjercicioInsertada, SIGNAL(entered()), this, SLOT(ejercicioIncidencia()));

  QState *estadoEntrenamientoEliminado = new QState();
  connect(estadoEntrenamientoEliminado, SIGNAL(entered()), this, SLOT(entrenamientoEliminado()));

  QState *editarEntrenamientoNuevoEjercicioInsertado = new QState();
  connect(editarEntrenamientoNuevoEjercicioInsertado, SIGNAL(entered()), this, SLOT(ejercicioInsertado()));

  QState *editarEntrenamientoEjercicioEditado = new QState();
  connect(editarEntrenamientoEjercicioEditado, SIGNAL(entered()), this, SLOT(ejercicioEditado()));

  QState *editarEntrenamientoEjercicioEliminado = new QState();
  connect(editarEntrenamientoEjercicioEliminado, SIGNAL(entered()), this, SLOT(ejercicioEliminado()));

  QState *editarEntrenamientoEjercicioComentado = new QState();
  connect(editarEntrenamientoEjercicioComentado, SIGNAL(entered()), this, SLOT(ejercicioComentado()));

  QState *editarEntrenamientoIncidenciaEjercicioInsertada = new QState();
  connect(editarEntrenamientoIncidenciaEjercicioInsertada, SIGNAL(entered()), this, SLOT(ejercicioIncidencia()));

  // Mi Equipo
  QState *estadoMiEquipoRegistrado = new QState();
  connect(estadoMiEquipoRegistrado, SIGNAL(entered()), this, SLOT(miEquipoInsertado()));

  QState *estadoMiEquipoEditado = new QState();
  connect(estadoMiEquipoEditado, SIGNAL(entered()), this, SLOT(miEquipoEditado()));

  QState *estadoMiEquipoEliminado = new QState();
  connect(estadoMiEquipoEliminado, SIGNAL(entered()), this, SLOT(miEquipoEliminado()));

  // Palmarés
  QState *estadoTituloInsertado = new QState();
```

```
connect(estadoTituloInsertado, SIGNAL(entered()), this, SLOT(tituloInsertado()));

QState *estadoTituloEditado = new QState();
connect(estadoTituloEditado, SIGNAL(entered()), this, SLOT(tituloEditado()));

QState *estadoTituloEliminado = new QState();
connect(estadoTituloEliminado, SIGNAL(entered()), this, SLOT(tituloEliminado()));

// Jugadores
QState *estadoJugadorInsertado = new QState();
connect(estadoJugadorInsertado, SIGNAL(entered()), this, SLOT(jugadorInsertado()));

QState *estadoJugadorEditado = new QState();
connect(estadoJugadorEditado, SIGNAL(entered()), this, SLOT(jugadorEditado()));

QState *estadoJugadorEliminado = new QState();
connect(estadoJugadorEliminado, SIGNAL(entered()), this, SLOT(jugadorEliminado()));

QState *mostrandoDatosPersonales = new QState();
connect(mostrandoDatosPersonales, SIGNAL(entered()), this, SLOT(mostrarDatosPersonales()));

// Competición
QState *estadoCompeticionInsertada = new QState();
connect(estadoCompeticionInsertada, SIGNAL(entered()), this, SLOT(competicionInsertada()));

QState *estadoCompeticionEliminada = new QState();
connect(estadoCompeticionEliminada, SIGNAL(entered()), this, SLOT(competicionEliminada()));

QState *estadoCompeticionEditada = new QState();
connect(estadoCompeticionEditada, SIGNAL(entered()), this, SLOT(competicionEditada()));

// Partidos
QState *estadoPartidoInsertado = new QState();
connect(estadoPartidoInsertado, SIGNAL(entered()), this, SLOT(partidoInsertado()));

QState *estadoIncidenciaRegistrada = new QState();
connect(estadoIncidenciaRegistrada, SIGNAL(entered()), this, SLOT(incidenciaRegistrada()));

QState *estadoAlineacionInsertada = new QState();
connect(estadoAlineacionInsertada, SIGNAL(entered()), this, SLOT(alineacionInsertada()));

QState *estadoLesionRegistrada = new QState();
connect(estadoLesionRegistrada, SIGNAL(entered()), this, SLOT(lesionInsertada()));

QState *estadoLesionEliminada = new QState();
connect(estadoLesionEliminada, SIGNAL(entered()), this, SLOT(lesionEliminada()));

QState *estadoTarjetaRecibidaInsertada = new QState();
connect(estadoTarjetaRecibidaInsertada, SIGNAL(entered()), this, SLOT(tarjetaRecibidaInsertada()));

QState *estadoTarjetaRecibidaEliminada = new QState();
connect(estadoTarjetaRecibidaEliminada, SIGNAL(entered()), this, SLOT(tarjetaRecibidaEliminada()));

QState *estadoTarjetaProvocadaInsertada = new QState();
connect(estadoTarjetaProvocadaInsertada, SIGNAL(entered()), this, SLOT(tarjetaProvocadaInsertada()));

QState *estadoTarjetaProvocadaEliminada = new QState();
connect(estadoTarjetaProvocadaEliminada, SIGNAL(entered()), this, SLOT(tarjetaProvocadaEliminada()));

QState *estadoSustitucionInsertada = new QState();
connect(estadoSustitucionInsertada, SIGNAL(entered()), this, SLOT(sustitucionInsertada()));

QState *estadoSustitucionEliminada = new QState();
```

```
connect(estadoSustitucionEliminada, SIGNAL(entered()), this, SLOT(sustitucionEliminada()));

QState *estadoGolMarcadoInsertado = new QState();
connect(estadoGolMarcadoInsertado, SIGNAL(entered()), this, SLOT(golMarcadoInsertado()));

QState *estadoGolMarcadoEliminado = new QState();
connect(estadoGolMarcadoEliminado, SIGNAL(entered()), this, SLOT(golMarcadoEliminado()));

QState *estadoGolRecibidoInsertado = new QState();
connect(estadoGolRecibidoInsertado, SIGNAL(entered()), this, SLOT(golRecibidoInsertado()));

QState *estadoGolRecibidoEliminado = new QState();
connect(estadoGolRecibidoEliminado, SIGNAL(entered()), this, SLOT(golRecibidoEliminado()));

QState *estadoPartidoFinalizado = new QState();
connect(estadoPartidoFinalizado, SIGNAL(entered()), this, SLOT(partidoFinalizado()));

QState *estadoPartidoEliminado = new QState();
connect(estadoPartidoEliminado, SIGNAL(entered()), this, SLOT(partidoEliminado()));

QState *estadoEditarNuevaLesion = new QState();
connect(estadoEditarNuevaLesion, SIGNAL(entered()), this, SLOT(lesionInsertada()));

QState *estadoEditarEliminarLesion = new QState();
connect(estadoEditarEliminarLesion, SIGNAL(entered()), this, SLOT(lesionEliminada()));

QState *estadoEditarIncidencia = new QState();
connect(estadoEditarIncidencia, SIGNAL(entered()), this, SLOT(incidenciaRegistrada()));

QState *estadoEditarNuevaSustitucion = new QState();
connect(estadoEditarNuevaSustitucion, SIGNAL(entered()), this, SLOT(sustitucionInsertada()));

QState *estadoEditarEliminarSustitucion = new QState();
connect(estadoEditarEliminarSustitucion, SIGNAL(entered()), this, SLOT(sustitucionEliminada()));

QState *estadoEditarNuevoGolMarcado = new QState();
connect(estadoEditarNuevoGolMarcado, SIGNAL(entered()), this, SLOT(golMarcadoInsertado()));

QState *estadoEditarEliminarGolMarcado = new QState();
connect(estadoEditarEliminarGolMarcado, SIGNAL(entered()), this, SLOT(golMarcadoEliminado()));

QState *estadoEditarNuevoGolRecibido = new QState();
connect(estadoEditarNuevoGolRecibido, SIGNAL(entered()), this, SLOT(golRecibidoInsertado()));

QState *estadoEditarEliminarGolRecibido = new QState();
connect(estadoEditarEliminarGolRecibido, SIGNAL(entered()), this, SLOT(golRecibidoEliminado()));

QState *estadoEditarNuevaTarjetaRecibida = new QState();
connect(estadoEditarNuevaTarjetaRecibida, SIGNAL(entered()), this, SLOT(tarjetaRecibidaInsertada()));

QState *estadoEditarEliminarTarjetaRecibida = new QState();
connect(estadoEditarEliminarTarjetaRecibida, SIGNAL(entered()), this, SLOT(tarjetaRecibidaEliminada()));

QState *estadoEditarNuevaTarjetaProvocada = new QState();
connect(estadoEditarNuevaTarjetaProvocada, SIGNAL(entered()), this, SLOT(tarjetaProvocadaInsertada()));

QState *estadoEditarEliminarTarjetaProvocada = new QState();
connect(estadoEditarEliminarTarjetaProvocada, SIGNAL(entered()), this, SLOT(tarjetaProvocadaEliminada()));

QState *estadoFinPartidoEditado = new QState();
connect(estadoFinPartidoEditado, SIGNAL(entered()), this, SLOT(partidoFinalizado()));
```

- Tabla 9-

En la tabla observamos lo anteriormente comentado, ya que si tomamos como ejemplo la siguiente declaración:

QState *estadoEditarEliminarTarjetaRecibida = new QState();

connect(estadoEditarEliminarTarjetaRecibida, SIGNAL(entered()), this,
SLOT(tarjetaRecibidaEliminada()));

Vemos claramente marcado en naranja y en rosa las peculiaridades mencionadas en el párrafo previo a la tabla 9.

Por otra parte, destacar que todos y cada uno de estos estados tienen como fin único ejecutar la transacción que el slot correspondiente determine, y transitarán automáticamente al finalizar las acciones correspondientes a otro estado. Este paso intermedio se omitió en el diseño inicial, y es por eso que se ha decidido incluir un fragmento de código fuente que describe el comportamiento de una parte de la máquina de estados que responde a esta estructura de ejecución, así como un pequeño diagrama que representa la máquina de estados en ese fragmento. El código se muestra en la tabla 10.

```
//////////////////////////
// Mi Equipo ////////////
 //////////////////////////
   estadoMiEquipo->addTransition(this->ui->botonMiEquipo_Volver, SIGNAL(clicked()), estadoInicio);
   estadoMiEquipo->addTransition(this->ui->botonMiEquipo_Jugadores, SIGNAL(clicked()), estadoJugadores);
   estadoMiEquipo->addTransition(this->ui->botonMiEquipo_Palmares, SIGNAL(clicked()), estadoPalmares);
   estadoMiEquipo->addTransition(this->ui->botonMiEquipo_EditarInformacion, SIGNAL(clicked()), infoMiEquipo);
   estadoMiEquipo->addTransition(this->ui->botonMiEquipo_Rendimiento, SIGNAL(clicked()),
rendimientoMiEquipo);
   estadoMiEquipo->addTransition(this->ui->botonMiEquipo_Eliminar, SIGNAL(clicked()),
estadoMiEquipoEliminado);

   estadoMiEquipoEliminado->addTransition(estadoMiEquipo);

   // Jugadores
   estadoJugadores->addTransition(this->ui->botonJugadores_Volver, SIGNAL(clicked()), estadoMiEquipo);
   estadoJugadores->addTransition(this->ui->botonJugadores_NuevoJugador, SIGNAL(clicked()), nuevoJugador);
   estadoJugadores->addTransition(this->ui->botonJugadores_EditarJugador, SIGNAL(clicked()), jugadorAEditar);
   estadoJugadores->addTransition(this->ui->botonJugadores_EliminarJugador, SIGNAL(clicked()),
eliminarJugador);
   estadoJugadores->addTransition(this->ui->botonJugadores_VerDatosPersonales, SIGNAL(clicked()),
verDatosPersonales);
   estadoJugadores->addTransition(this->ui->botonJugadores_VerDetalles, SIGNAL(clicked()),
jugadorRendimiento);

   // Nuevo Jugador
   nuevoJugador->addTransition(this->ui->botonNuevoJugador_Cancelar, SIGNAL(clicked()), estadoJugadores);
   nuevoJugador->addTransition(this->ui->botonNuevoJugador_Finalizar, SIGNAL(clicked()),
estadoJugadorInsertado);

   estadoJugadorInsertado->addTransition(estadoJugadores);

   // Jugador A Editar
   jugadorAEditar->addTransition(this->ui->botonJugadorAEditar_Cancelar, SIGNAL(clicked()), estadoJugadores);
   jugadorAEditar->addTransition(this->ui->botonJugadorAEditar_Continuar, SIGNAL(clicked()), editarJugador);

   // Editar Jugador
   editarJugador->addTransition(this->ui->botonEditarJugador_Cancelar, SIGNAL(clicked()), jugadorAEditar);
   editarJugador->addTransition(this->ui->botonEditarJugador_Editar, SIGNAL(clicked()), estadoJugadorEditado);
```

```
    estadoJugadorEditado->addTransition(estadoJugadores);

    // Eliminar Jugador
    eliminarJugador->addTransition(this->ui->botonEliminarJugador_Cancelar, SIGNAL(clicked()), estadoJugadores);
    eliminarJugador->addTransition(this->ui->botonEliminarJugador_Eliminar, SIGNAL(clicked()),
estadoJugadorEliminado);

estadoJugadorEliminado->addTransition(estadoJugadores);
```

- Tabla 10-

Como se observa en la tabla 10, se ha resaltado en rojo aquellas transiciones en las cuales se ven involucrados los estados a los que estamos haciendo referencia. En la página siguiente se muestra un fragmento del diagrama que se correspondería al fragmento de código fuente presentado en la tabla 10.

Del diagrama destacar que marcamos con (*) los estados nuevos, y las transiciones automáticas de éstos al terminar las acciones la indicamos como λ-transición. Por otra parte, en el mismo diagrama vemos indicado con (*) los estados intermedios introducidos para realizar procesos tales como editar o eliminar, cuya aparición, como se ha explicado previamente , se debe a la limitación respecto al manejo de la QTableView y su capacidad para dejar trabajar con los contenidos de sus filas haciendo clic.

Así pues, estas son las novedades más destacables de la máquina de estados que dirige el funcionamiento de la aplicación, y la explicación sobre el fragmento de código explicado se hace extensible a toda la aplicación en los puntos en los que se ha dado el mismo problema. Es fácilmente observable en el código fuente, ya que el problema se ha abordado de esta manera en todos y cada uno de los casos.

Destacar también, que la aplicación sigue siendo dirigida por la máquina de estados del mismo modo que se explicó en la parte correspondiente al diseño de interfaz gráfica sin funcionalidad. Cada estado hace una llamada a procedimiento de un slot que actualizará la variable global state y con el nuevo valor de la misma se invocará a void MainWindow::stateChanged(int state) que determinará cuál es el frame visible, y además ahora incoporará en cada stado las acciones a realizar por la interfaz en sí misma para ese estado (vaciar los campos de texto, rellenarlos con los datos de algún jugador, buscar en la base de datos la información del último partido para mostrar su resultado, etc.). En la tabla 11 se observa un ejemplo del fragmento del método stateChanged con las acciones a realizar en él.

```
void MainWindow::stateChanged(int state) {
    // Todos los frames están ocultos
    this->ui->inicioFrame->setVisible(false);
    this->ui->equiposFrame->setVisible(false);
……..

    // Equipos
    else if (state == 1){
        // Menú Equipos
        gestionEquipos->inicializarTablaEquipos(this->ui->tablaEquipos);

        this->ui->nuevoNombreEquipoTextField->setText("");
        this->ui->NuevoEquipoDirEstadioTextField->setText("-");
        this->ui->nuevoEquipoPoblacionTextField->setText("-");
        this->ui->nuevoEquipoTelefonoTextField->setText("-");
        this->ui->nuevoEquipoEMailTextField->setText("-");
        this->ui->nuevoEquipoEquipajesTextField->setText("-");

        this->ui->equiposFrame->resize(500,260);
        this->ui->equiposFrame->setVisible(true);
    }
```

-Tabla 11-

Faltan estados para completar la máquina de estados referente a este módulo de la aplicación. Este diagrama en este punto es de carácter ilustrativo e identificador de estados análogos.

3.4. MÓDULO MI EQUIPO

El módulo “Mi Equipo” es el principal de la aplicación desde el punto de vista del arranque de la misma. Desde este bloque se gestiona, registra y consulta todo aquello relativo al equipo que el usuario dirige.

Al iniciar la aplicación por primera vez, nada más ser insalada, el usuario encuentra ante sí la pantalla de inicio con los botones correspondientes a cada uno de los cuatro módulos que componen la aplicación (Mi Equipo, Equipos, Competición y Entrenamientos), pero, al no tener su equipo registrado, el único botón habilitado es el que le indica el acceso al menú de Mi Equipo.

El menú “Mi Equipo” da opción a acceder a cinco bloques dentro del módulo que son: Registrar/Editar Información de Mi Equipo, Eliminar Mi Equipo, Jugadores, Rendimiento y Palmarés. A continuación procedemos a explicarlos por separado y con detalle.

En primer lugar encontramos la subsección “Registrar/Editar Información de Mi Equipo”. Este bloque se presenta de manera diferente en la aplicación dependiendo de si el equipo manejado y dirigido por el usuario está dado de alta en el sistema o no. Esto se describe de la siguiente froma en el código de la aplicación: Al entrar al estado que activa la ventana relativa al módulo Mi Equipo, la primera acción que se realiza es una consulta sobre la base de datos del sistema para verificar la existencia de un equipo propio en el mismo. Si la consulta a la base de datos resulta positiva, es decir, hay un equipo ya registrado, el botón correspondiente a la subsección del presente módulo que estamos tratando mostrará el texto “Registrar Mi Equipo”, dando acceso, por tanto a una ventana que solicitará al usuario el nombre de su equipo, la dirección del estadio, la población, el teléfono de las oficnas, la dirección de correo electrónico y los equipajes que el equipo propio viste, destacando además que ninguno de los campos solicitados puede quedar vacío, aunque,por el contrario, si deseamos que alguno no refleje la información solicitada, podemos rellenarlo con un guión y el sistema dará el registro del equipo por válido. Una vez registrado el equipo, ya se tiene acceso al resto de subsecciones del módulo Mi Equipo, destacando además, que el botón donde antes se leía “Registrar Mi Equipo” ahora tiene por texto “Editar Información de Mi Equipo”, dando lugar éste a una pantalla exactamente igual a la del registro del equipo, pero con los campos ya completados con los datos que del propio equipo estuvieran en ese momento registrados y vigentes en el sistema.

Por el contrario, el usuario encontrará en este módulo la funcionalidad que le permita elimir del sstema su propio equipo cuando éste deje de trabajar en él. Mediante la acción del botón Eliminar Mi Equipo, el usuario eliminará todos los datos que en el sistema estuvieran registrados, es decir, que vaciaría la base de datos que hace de almacén de información de la aplicación, quedando esta última en un estado idéntico al que presenta nada más ser instalada en el dispositivo. Este procedimiento se lleva a cabo de la manera indicada atendiendo a la razón de que un entrenador no puede dirigir más de un equipo simultáneamente, y el sistema sólo admite, por tanto, un equipo propio registrado, y toda la información que se almacene en el sistema será relativa al equipo en ese momento almacenado como propio. Debido a este hecho, al eliminar el equipo propio, toda la información almacenada como consecuencia de la dirección del mismo será también borrada de la base de datos, ya que ésta, aunque permaneciera en la misma, no sería accesible por el usuario a través de la aplicación, por tanto esa información residual que quedaría en el sistema se ha decidió eliminarla al 100 % para mejorar el rendimiento de la aplicación a medio y largo plazo.

El tercer apartado de los cinco que componen mi equipo, es aquel en el cual se almacena el Plamarés del equipo. El palmarés de un equipo es el elenco de títulos conseguidos como consecuencia de su participación en una competición. De esta definición extraemos por tanto una restricción tan clara como obvia, y no es otra que para poder añadir un título al palmarés, debe haber registrada una competición

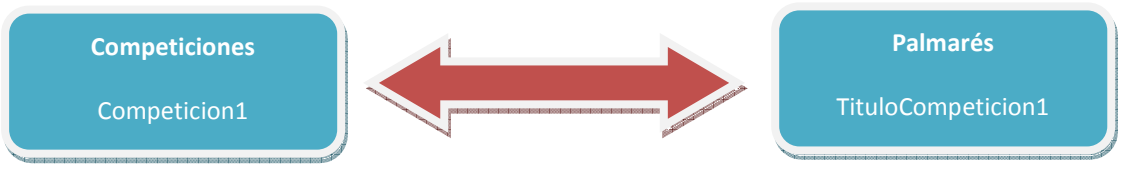
a la cual se le pueda asociar un título o una clasificación final en la misma. A partir de esa restricción, el funcionamiento de este subapartado es bastante elemental, contando el usuario con las tres funcionalidades básicas de insertar un título, modificar la información de un título registrado en el palmarés, y la eliminación de los mismos.

La inserción de un nuevo título consiste en seleccionar una competición a la que asociar el título. Se selecciona el nombre de la competición y las temporadas de inicio y fin de la misma, y finalmente el usuario añade la posición en la cual quedó su equipo en dicha competición, pudiéndose introducir la posición tanto en letras como en valor númerico, se deja a criterio del usuario el modo de empleo en este caso. Una vez introducidos todos los datos, se procede a validar el ingreso de los datos en el sistema, consultando la aplicación de manera interna y transparente al usuario dos hechos: la competición a la que añadir el título debe existir, y en segundo lugar, si se cumple el primer criterio, hay que comprobar que la competición seleccionada no tenga ya asociado un título. En caso de que una de estas dos reglas no se cumplan, la aplicación devuelve un error por pantalla al usuario comunicándole la condición infringida, y por supuesto, sin insertar los datos previamente introducidos.

De un modo similar, la edición de los títulos comienza con la selección de una competición a editar, la cual debe existir, ya que si el usuario selecciona en los QComboBox una competición inexistente, la aplicación lo gestiona mostrando un mensaje de error por pantalla e invitando al usuario a volver al menú de palmarés. Si todo transcurre correctamente, se muestra al usuario una pantalla con la competición que está editando y la posición que en ésta ocupó, siendo este último dato el único que se puede editar de un título.

La eliminación de títulos se hace del mismo modo que la edición, es decir, se selecciona una competición cuyo título se eliminará, y si todo es correcto se eliminará previa pregunta de confirmación a la cual el usuario deberá responder afirmativamente; en caso contrario, se mostrará el error y el título no será insertado.

Destacar además, que otro modo de eliminar un título del palmarés es mediante la eliminación de la competición a la cual pertenece el mismo.



-Vínculo entre las competiciones y el palmarés en la aplicación-

En penúltimo lugar, encontramos la parte de rendimiento del equipo donde se presentan una serie de datos estadísticos generales tales como los partidos jugados, de los jugados cuantos se saldaron con victoria, cuantos con empate y cuantos con derrota, así como los goles marcados, los encajados, la diferencia entre estos dos últimos datos, las amonestaciones y las expulsiones. Todos estos datos se calcular y se myestran en la interfaz mediante consultas a la base de datos en las cuales se cuentan los registros de goles marcados, los de goles recibidos, las tarjetas recibidas cuyo color es amarillo, o las ue son rojas o segunda amarilla, y por tanto, expulsión, y se cuentan los goles de cada partido para averiguar si el partido se ganó o se perdió en función de los goles encajados y los marcados en el mismo.

A partir de estas estadísticas generales, encontramos tres nuevos bloques en los cuales se nos desglosan los goles recibidos según minuto, los goles marcados según minuto y las tarjetas recibidas según minuto. Los rangos de minutos estudiados en estas nuevas estadísticas más específicas son los siguientes: [1-5], [6-22], [23-45], [46-50], [51-67], [68-85], [86-90], [91-120 ó Prórroga], coincidiendo estos rangos con los primeros y últimos cinco minutos de cada parte del partido, así como el intervalo que recoge del minuto 6 al 22 y del 23 al 40 de cada parte del mismo, dividiendo, por tanto, cada mitad del partido en cuatro intervalos temporales. Asimismo, además de mostrarse el número de goles marcados, recibidos o tarjetas recibidas (desglosadas en amonestaciones y expulsiones), se muestra también el porcentaje que representa la cantidad mostrada según el intervalo de tiempo. Ver figuras 7 y 8.

- Figura 7 –

- Figura 8-

En este caso, es fácil observar que el rendimiento del equipo está vinculado a los partidos que se disputan (Módulo Competición, que se explicará más adelante) y a los goles y tarjetas que en ellos se almacenan.

Por último, para terminar con el módulo Mi Equipo, encontramos la sección de los jugadores pertenecientes al equipo que el usuario dirige. En este apartado, el usuario podrá insertar y eliminar jugadores, editar la información relativa a los mismos y mostrar su información personal, así como la opción de analizar el rendimiento de los mismos consultando una serie de estadísticas y de tablas de datos que en párrafos próximos detallaremos.

Para insertar un jugador el sistema solicitará al usuario la información relativa al jugador a insertar, y estos datos serán su nombre, su dorsal, la demarcación que éste ocupa en el campo, su dirección y la población donde vive, el código postal y el teléfono de contacto (ambos datos tienen que ser forzosamente un valor numérico, obviamente sin decimales), la fecha de nacimiento y el año de infgreso en el club, año que, por razones obvias, debe ser posterior al año de nacimiento del jugador. Destacar que ningún campo de los que se muestran en el formulario de registro puede quedar vacío, de manera que si algún dato desconocido, su campo se debe rellenar única y exclusivamente con un guión (-) para que el sistema dé como válida la inserción. Los datos relativos a partidos jugados, minutos disputados, goles marcados, goles recibidos, tarjetas provocadas y tarjetas recibidas son inicializados a 0 de manera invisible al usuario nada más darse de alta el jugador en la aplicación.

La edición de los datos del jugador tiene como partida la selección del jugador cuya información pretende ser editada. Una vez seleccionado el jugador, se mostrará en pantalla una ventana idéntoca a la de inserción, pero en los campos del formulario se mostrarán los datos correspondientes al jugador en cuestión para que el usuario vea la información vigente y pueda modificar los campos que considere oportunos, considerando las mismas reglas en la introducción de datos que para la parte de la inserción del mismo.

Destacamos como restricciones en el apartado de jugadores que cada jugador debe tener su nombre como único, es decir, no pueden registrarse dos jugadores con el mismo nombre, y además cada dorsal sólo podrá ser ocupado por un jugador. Estas restricciones son analizadas por el sistema cuando el usuario co0nfirma la inserción o la edición de los datos de un jugador, y en caso de que los datos introducidos por el usuario violen alguna de estas dos reglas, se mostrará el error correspondiente en pantalla y se anulará la transacción que el usuario pretendía llevar a cabo.

La eliminación de los jugadores es un caso particular. El proceso se inicia seleccionando el jugador a eliminar, y tras la confirmación de que se desea llavar a cabo el borrado, lo que hace el sistema es actualizar los datos del jugador, cambiándole el dorsal a 0 (ningún jugador tiene opción de llevarlo, ya que los dorsales van del 1 al 45) y modificando su nombre al de “Jugador NPE” (No Pertenece al Equipo), con el fin de no alterar los datos que se presentarán en el apartado de Histórico de Partidos. En este apartado que se explicará en puntos venideros de esta memoria, se muestran las alineaciones, goles, etc. de los partidos disputados según la fecha y el rival que seleccione el usuario, de modo que si se elimina el jugador, esta información se mostraría sesgada, ya que por ejemplo, aunque el partido hubiese terminado 2-0, si el jugador eliminado hubiese marcado un gol, en el histórico, al pulsar goles marcados sólo aparecería una tupla tomada de la base de datos. Destacar que a ojos del usuario, el jugador ha sido eliminado y nunca más será accesible, ni para editarlo, ni para consultar sus estadísticas particulares, ni podremos ver sus datos personales, ni será mostrado en la tabla del menú jugadores, de manera que la única manifestación del jugador en el sistema será en el histórico de partidos y mediante un nombre “forzado”, que por otra parte, tendrán todos los jugadores que sean eliminados.

-Vínculo entre el módulo Jugador (al eliminar) y el histórico de Partidos-

Respecto al apartado “Mostrar Datos Personales”, su funcionamiento es simple. Se selecciona del QComboBox el nombre del jugador cuyos datos se quiere consultar (hay que recordar que el nombre de cada jugador es único) y se pulsa la tecla contigua al mismo que dice “Mostrar Datos”. Tras pulsar dicho botón, el sistema realiza una búsqueda en la base de datos consultando el dorsal, la demarcación, la dirección, la población, el código postal, el teléfono, la fecha de nacimiento y el año de ingreso en el club del jugador cuyo nombre está en el QComboBox en el momento en que éste se acciona, y se rellena cada campo del formulario con el dato correspondiente.

Por último, el usuario tendrá la opción de acceder a las estadísticas personales del jugador que seleccione. En estas estadísticas se mostrará numéricamente el número de partidos jugador por él, los minutos que ha disputado, los goles que ha marcado y los que el equipo ha encajado mientras el jugador seleccionado estaba participando activamente en el partido desde el terreno de juego, así como las tarjetas que ha recibido y que ha provocado. Ver figura 9.

-Figura 9-

Como se observa en la figura 9, además de mostrar los datos numéricos, se presentan la posibilidad de ahondar un poco más en el análisis de la estadísticas, viendo detalladamente tuplas de la base de datos con los siguientes datos en cada una de ellas:

- Partidos que ha disputado (Ver partidos): Rival, campo, competición.
- Minutos Disputados (Ver minutos): Rival, campo, competición, minutos.
- Goles Marcados (Ver goles marcados): Rival, campo, competición, minuto del gol, resultado provisional.
- Goles Recibidos (Ver goles recibidos): Rival, campo, competición, minuto del gol recibido, resultado provisional.

- Tarjetas Recibidas (Ver tarjetas recibidas): Rival, campo, competición, minuto, tipo de tarjeta, comentarios anotados.
- Tarjetas Provocadas (Ver tarjetas provocadas): Rival, campo, competición, minuto, tipo de tarjeta, comentarios anotados.
- Lesiones : Fecha, rival, minuto, descripción de la lesión. Destacar de las lesiones que éstas no se presentan numéricamente y que el botón para verlas se habilita cuando en la base de datos hay una lesión asociada al jugador que se está analizando. Esta consulta se hace al alcanzar el estado de la máquina correspondiente al rendimiento del jugador seleccionado.

Nótese que todos estos botones están habilitados o inhabilitados en función del valor numérico de la estadística en cuestión, es decir, si una estadística numérica tiene como valor 0, el botón correspondiente se inhabilitará, ya que la tabla que mostraría estaría vacía, y no interesa al usuario.

Todos los datos de las estadísticas de los jugadores son obtenidos de manera similar a como se obtiene el rendimiento del equipo, de modo que los jugadores también tienen un vínculo claramente marcado con los partidos del modo competición, ya que lo que en estos últimos acontece se ve reflejado en las estadísticas indicviduales de cada jugador.

3.5. MÓDULO EQUIPOS

Este segundo bloque de la aplicación que a continuación procedemos a describir es el que se encarga básicamente de la gestión de los equipos que serán rivales del equipo del usuario y que por tanto, podrán participar en partidos pertenecientes a cualquiera de las competiciones registradas contra el equipo que dirige el usuario.

Destaca del menú de equipos que consta de cuatro botones que dan acceso a las diversas funcionalidades y procesamientos que con ellos se pueden llevar a cabo, así como de una tabla que contiene los nombres de todos los equipos registrados en el sistema para que éstos sean de rápida consulta para el usuario, pudiendo ver qué equipos son rivales potenciales simplemente observando los datos de la misma.

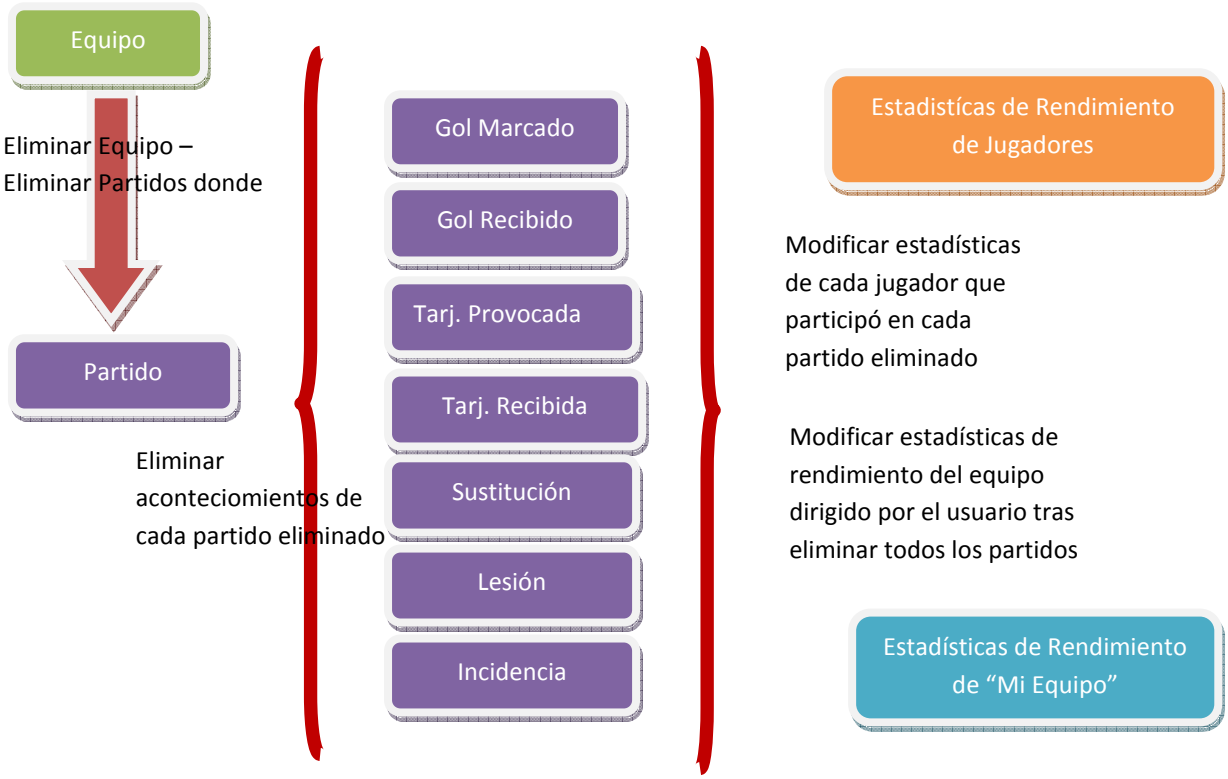
Dicho esto, podemos proceder a analizar y detallar las funcionalidades que este módulo ofrece para la gestión de los datos de los equipos.

En primer lugar, lógicamente, el usuario tendrá la posibilidad de dar de alta tantos equipos como desee o considere oportuno, siendo la única restricción el hecho de que no puede haber dos equipos con el mismo nombre. Para completar la inserción, el usuario deberá introducir en el sistema el nombre del equipo, la dirección del estadio donde el rival lleva a cabo sus partidos como local, la población, el teléfono, que obligatoriamente debe ser un número entero, la dirección de correo electrónico y los colores de la equipación que el equipo viste. Todos estos datos son de inserción obligatoria, pudiéndose reemplazar el dato real en caso de ser desconocido o de quererse omitir por parte del usuario por un guión (-), en caso de dejar algún campo en blanco o no introducir los datos con un formato válido (por ejemplo, números y letras en el teléfono), el sistema devolverá por pantalla el error cometido, invitando al usuario a insertar de nuevo el equipo.

La edición de los equipos es similar a la inserción. Al pulsar el botón “Editar Equipo”, el sistema nos conducirá, por la transición de la máquina de estados activada por dicho evento, a una ventna donde

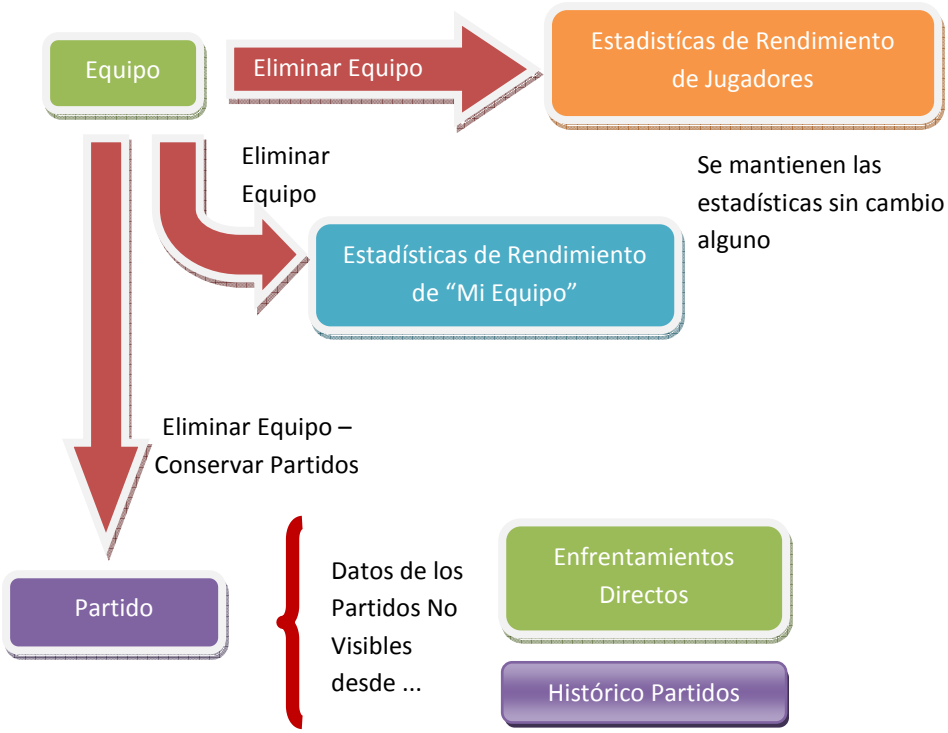
seleccionaremos el nombre del equipo cuya información procederemos a editar (cabe recordar que el nombre de cada equipo es único, y por tanto, no hay lugar a ambigüedades entre los equipos dados de alta en el sistema). En caso de no haber ningún equipo registrado en el sistema, aparecerá un mensaje en esta nueva ventana que indicará que el procedimiento solicitado no se puede llevar a cabo por dicho motivo, e instará a usuario a volver al menú de equipos, ya que el botón "Volver" será el único que permanezca habilitado bajo esas circunstancias. Una vez seleccionado correctamente el equipo a editar, se mostrará al usuario una pantalla similar a la de inserción, pero con los campos del formulario completados con los datos del equipo seleccionado que estuviera vigentes en el sistema, proceso que se obtiene como resultado de una consulta a la base de datos introduciendo como parámetro el nombre del equipo, acción que se realiza como resultado de alcanzar el estado correspondiente a la edición de un equipo, es decir, cada vez que el usuario desee modificar la información de un equipo y lo haya seleccionado, al pasar al alcanzar el siguiente estado, el sistema realizará la consulta pertinente y mostrará los datos oportunos. Por otra parte destacar que, obviamente las restricciones y los campos que el formulario de edición muestra son exactamente los mismos que rigen el comportamiento de la funcionalidad de insertar equipo.

El tercero de los procedimientos que se pueden llevar a cabo con un equipo es, lógicamente, eliminarlo. El borrado de un equipo tiene una peculiaridad ya que admite dos posibilidades, el borrado que en el código se denomina sencillo, y el borrado en cascada.
En la ventna de eliminación se deja bien claro mediante un rótulo que por defecto, el borrado que se efectuará será un borrado en cascada, teniendo éste como consecuencia la eliminación de los partidos en los cuales el equipo seleccionado, y por tanto de todos los acontecimientos del mismo, es decir, goles, lesiones, tarjetas recibidas y provocadas... afectando todas estas eliminaciones a las estadísticas tanto del rendimiento del equipo como de las estadísticas de los jugadores.



- Esquema de módulos afectados por el borrado en cascada-

El otro caso de eliminación posible es el borrado sencillo, el cual se habilita mediante la activación del CheckBox (que por defecto está inactivo) de la propia ventana de eliminación donde se muestra el mensaje “Sí, deseo conservar los partidos”. Al eliminar mediante esta opción, se elimina el equipo pero los partidos se conservan y ni las estadísticas de los jugadores ni del equipo dirigido por el usuario se ven afectadas. Desde las tablas de las estadísticas específicas de cada jugador (ver partidos Jugador, ver minutos disputados, etc.) se podrá observar como aparecen lo partidos jugados contra el equipo eliminado, los minutos jugados en los mismos, y los hechos relevantes de los partidos que hubiese tenido al jugador en cuestión como protagonista, si bien estos partidos no podrán ser visualizados ni desde el bloque de histórico de partidos ni desde el que analiza dentro del módulo equipo los enfrentamientos directos con un determinado rival, ya que desde ambas funcionalidades sólo se da opción a ver los partidos contra rivales vigentes en el sistema.



-Esquema de borrado de equipo sencillo-

Por último, desde el módulo equipos se tiene acceso a la funcionalidad que permite analizar el comportamiento y el rendimiento del equipo dirigido por el usuario contra un rival en concreto, siendo éste necesariamente, y como ya se dejó entrever en párrafos anteriores de este mismo punto, un equipo que este dado de alta en el equipo en el momento en que la consulta de esta funcionalidad se hace efectiva.

Por tanto, en este apartado, el usuario selecciona previamente el rival cuyos enfrentamientos directos quiere analizar, y el sistema le conducirá a una ventana donde el usuario obtendrá la información numérica de los partidos jugados contra dicho rival que se saldaron con victoria, los que lo hicieron con empate y cuántos acabaron en derrota. Asimismo, en esta ventana se muestra una tabla cuyo contenido es resultado de una consulta ala base de datos en la cual se analizan los partidos en los que el equipo seleccionado fue rival, devolviéndose en ella parámetros como el campo donde se jugó el partido, el resultado final del mismo, en qué competición ambos equipos se vieron las caras, y la fechade cada uno

de los partidos que contra ese equipo hayan sido disputados hasta el momento de la consulta. Se facilita la fecha ya que además es un dato interesante debido a la posibilidad que ofrece esta funcionalidad de saltar al historial de partidos, en el cual, una vez se haya visto los resultados y el balance contra el equipo rival, se podrá analizar más profundamente todos los hechos que acontecieron todos y cada uno de sus enfretamientos, y para ello , en el histórico de partidos se hace imprescindible introducir como parámetros de búsqueda la fecha y el rival que se procederán a analizar.

3.6. MÓDULO ENTRENAMIENTO

El módulo entrenamiento es el único de los existentes en el sistema en el que el usuario podrá trabajar con sesiones de entrenamiento a su antojo sin la necesidad de que haya jugadores insertados en el equipo ni equipos rivales. Los entrenamientos son sesiones de trabajo preparadas para el grupo dirigido por el usuario, de manera que el único prerrequisito que se exige para poder insertar, editar o eliminarlos es que el usuario haya dado de alta su propio equipo en el sistema desde el módulo “Mi Equipo”. Este razonamiento obedece a razones de planificación a medio plazo, es decir, podría darse la situación de que el usuario al registrar su equipo esté más preocupado de planificar los entrenamientos de pretemporada que de insertar los datos de todos los jugadores, labor que podría llevar a cabo con detenimiento más adelante cuando tenga que preparar partidos.

Una vez introducidos en la materia que este bloque aborda, destacar que el menú de entrenamiento proporciona acceso a cuatro grandes funcionalidades (Nuevo Entrenamiento, Editar Entrenamiento, Eliminar Entrenamiento e Histórico Entrenamientos), y apoya a las mismas mediante un calendario, incluido como ayuda al usuario para ver las fechas en las que programar entrenamientos. Este componente visual ha sido insertado desde Qt Designer, ya que es un componente propio del diseño de interfaces gráficas con Qt, y tiene una única funcionalidad que es la de soporte visual.

Llegados a este punto, el usuario tiene como primera opción dar de alta un entrenamiento, que tendrá que obedecer a la restricción de ser único en el día en que se programa, es decir, el usuario no podrá programar dos entrenamientos en el mismo día, y por este hecho, la búsqueda de un entrenamiento en el sistema, se hace internamente mediante consultas a la base de datos introduciendo como parámetro la fecha. Por otra parte, destacar de los entrenamientos que todos tienen un código identificador en la base de datos que los hace únicos, y se componen por un conjunto de ejercicios, cada uno asociado aun determinado entrenamiento. Debido a la relación entre entrenamientos y ejercicios y la multiplicidad de la misma (1:N, un entrenamiento puede tener múltiples ejercicios y cada ejercicio sólo puede pertenecer a un entrenamiento), en la tabla “Ejercicios” de la base de datos se incluye un campo “idEntrenamiento” el cual asocia cada ejercicio con su entrenamiento, siendo así más sencillo el manejo de cada sesión de entrenamiento con sus ejercicios.

Así pues, para insertar un ejercicio, en primer lugar deberemos insertar el entrenamiento, de modo que en una pantalla con el aspecto de la figura 10, el usuario deberá introducir la fecha del entrenamiento y confirmar que desea iniciar el mismo pulsando el botón “Empezar entrenamiento”. Al recibir este evento, si no hay ningún entrenamiento ya programado para ese día, el sistema introduce en la base de datos un entrenamiento con la fecha seleccionada y le asigna un identificador que servirá en adelante para insertárselo a los ejercicios que en el entrenamiento recién empezado vayan a realizarse.

-Figura 10-

En este momento, tras haberse iniciado el entrenamiento, al usuario se le ofrece el abanico de posibilidades que se observan en la parte izquierda de la figura 10. De esta manera, el usuario podrá empezar añadiendo ejercicios al entrenamiento rellenando el formulario correspondiente en el que deberá dar una descripción de cada ejercicio que se añada y determinar también la duración (valor numérico) en minutos de los mismos. Al editar ejercicios y al insertarlos se debe tener en cuenta que en cada entrenamiento no puede haber dos o más ejercicios con el mismo nombre/descripción, es decir, que cada ejercicio se identifica única e inequívocamente mediante su nombre y su identificador de entrenamiento.

Respecto al borrado de ejercicios, cabe destacar que se selecciona el ejercicio mediante su nombre en una lista desplegable que se ofrece en la interfaz gráfica. Para ejecutar el borrado definitivo del mismo, el sistema ejecuta una acción delete sobre la base de datos sobre el ejercicio con el nombre citado y el identificador de entrenamiento correspondiente a la fecha del entrenamiento vigente.

Cada ejercicio posee la opción de ser comentado o de que le sea añadida una incidencia. Explicamos conjuntamente estas dos funcionalidades porque su funcionamiento es análogo. Al accionar el botón que nos conduce a cualquiera de ellas, el sistema nos conduce a una ventana donde seleccionamos el ejercicio que comentar o al cual deseamos añadirle una incidencia, y al confirmar el ejercicio, el sistema nos muestra una ventana con la descripción y la duración del ejercicio y el comentario/incidencia que por defecto se añade a la base de datos cuando éste es creado (nótese que un ejercicio está compuesto por una descripción, su duración, comentarios, incidencias y el identificador del entrenamiento en la base de datos), el cual, por tanto, al confirmar el comentario o la incidencia será actualizado, es decir, el método que ejecuta la transacción hace una operación “update” en vez de un “insert” sobre la base de datos del sistema. Destacar que cada ejercicio sólo puede tener un único comentario y una única incidencia, de modo que si el usuario quere añadir cosas a comentar o más de una incidencia al ejercicio, deberá escribir el comentario/incidencia a continuación del texto del anterior, no deberá borrar lo antiguo para introducir lo nuevo, porque de ese modo estará eliminando la información que más tiempo llevaba en el sistema.

Una vez llevados a cabo todos los movimientos posibles en el sistema dentro del entrenamiento que se está llevando a cabo, el usuario puede terminar el mismo accionado el botón “Finalizar Entrenamiento”, haciendo notorio el hecho de que un entrenamiento puede terminar sin ejercicios por el hecho de que en cualquier otro momento éste puede ser editado.

La edición del entrenamiento comienza con la selección de una fecha de todas las que se muestran en la tabla con los entrenamientos realizados, la que se corresponda n el entrenamiento a editar (Ver Figura 11). Al confirmar que deseamos editar el entrenamiento, el sistema nos dirigirá a una pantalla similar a la de nuevo entrenamiento (Ver Figura 12) y el usuario podrá entonces añadir ejercicios, editar o eliminar los existentes o comentar y añadir incidencias a su antojo de la misma manera que lo hacía cuando el entrenamiento se empezaba de cero.

-Figura 11-

-Figura 12-

La eliminación de los entrenamientos se lleva a cabo en una ventana similar a la de la figura 11, en la cual se muestran todos los entrenamientos realizados según sus fechas y se da la opción al usuario de seleccionar la fecha del entrenamiento a eliminar, si confirma y la fecha coincide con la de un entrenamiento, el sistema averigua la identificación del mismo y lo elimina, borrando a continuación todos los ejercicios del sistema cuyo identificador de entrenamiento coincida con el del entrenamiento eliminado, es decir, que la eliminación del entrenamiento implica la eliminación de los ejercicios; por

este motivo, ya que sus tiempos de vida están relacionados, en el diagrama de clases de la página 11, entre las clases entrenamiento y ejercicios hay una agregación referencial inclusiva y no referencial, en la cual el entrenamiento es el continente de los ejercicios, y éstos son los contenidos.

-Esquema de borrado de Entrenamientos-

Por último, se puede consultar cualquier información relativa a los entrenamientos que hayan tenido lugar desde el histórico de los entrenamientos. De este modo, el usuario selecciona una fecha y al pulsar los botones de ver ejercicios, ver comentarios o ver incidencias, éstas se muestran en la tabla de la misma ventana. Ver Figura 13.

-Figura 13-

En este procedimiento, la máquina de estados recibe un evento en función del evento que se le accione y éste hace transitar a la máquina a un estado que ejecuta la acción correspondiente, consultando en la base de datos los ejercicios cuyo identifoicador de entrenamiento coincide con el identificador del entrenamiento de la fecha seleccionada, o los comentarios o incidencias de los mismos, según cual sea el evento recibido. Al terminar dicha transacción y mostrar los resultados en la tabla, el estado ha completado sus acciones y vuelve automáticamente al estado del estado de entrenamientos a la espera de recibir un nuevo evento. Estas acciones se reflejan en el código fuente de la aplicación mediante lo especificado en la tabla 12.

```
// Histórico entrenamientos
   QState *historicoEntrenamientos = new QState();
   connect(historicoEntrenamientos, SIGNAL(entered()), this, SLOT(startHistoricoEntrenamientos()));

   QState *historicoEntrenamientosVerEjercicios = new QState();
   connect(historicoEntrenamientosVerEjercicios, SIGNAL(entered()), this,
SLOT(historicoEntrenamientosVerEjercicios()));

   QState *historicoEntrenamientosVerComentarios = new Qstate();
   connect(historicoEntrenamientosVerComentarios, SIGNAL(entered()), this,
SLOT(historicoEntrenamientosVerComentarios()));
```

```
    Qstate *historicoEntrenamientosVerIncidencias = new Qstate();
    connect(historicoEntrenamientosVerIncidencias, SIGNAL(entered()), this,
SLOT(historicoEntrenamientoVerIncidencias()));

[Los slots historicoEntrenamientoVerEjercicios, historicoEntrenamientoVerComentarios y ]
[historicoEntrenamientoVerIncidencias llaman a los métodos verEjerciciosEntrenamiento]
[verComentariosEntrenamiento y verIncidenciasEntrenamiento implementados en el fichero entrenamiento.cpp]

[.....]

// Histórico Entrenamientos
    historicoEntrenamientos->addTransition(this->ui->botonHistoricoEntrenamientos_Volver, SIGNAL(clicked()),
estadoEntrenamiento);

    historicoEntrenamientos->addTransition(this->ui->botonHistoricoEntrenamientos_VerEjercicios,
SIGNAL(clicked()), historicoEntrenamientosVerEjercicios);

    historicoEntrenamientos->addTransition(this->ui->botonHistoricoEntrenamientos_VerComentarios,
SIGNAL(clicked()), historicoEntrenamientosVerComentarios);

    historicoEntrenamientos->addTransition(this->ui->botonHistoricoEntrenamientos_VerIncidencias,
SIGNAL(clicked()), historicoEntrenamientosVerIncidencias);

    historicoEntrenamientosVerEjercicios->addTransition(historicoEntrenamientos);
    historicoEntrenamientosVerComentarios->addTransition(historicoEntrenamientos);
    historicoEntrenamientosVerIncidencias->addTransition(historicoEntrenamientos);
```

-Tabla 12-

3.7. MÓDULO COMPETICIÓN

El módulo de competición es el último y uno de los más complejos de la aplicación. En este bloque se dan lugar las funcionalidades básicas del manejo de competiciones que el equipo del usuario disputará, así como los partidos que éste llevará a cabo contra otros equipos, previamente registrados en el sistema desde el bloque correspondiente.

En primer lugar, para hacer funcionar el módulo, y teniendo como prerrequisito que, obviamente, el usuario ha dado de alta su equipo en el sistema, es dar de alta una competición, ya que todo partido que se desee disputar debe pertenecer a una, incluso los partidos amistosos están obligados a ser incluidos en una competición a la que el usuario podría llamar a su antojo y que bien podría ser Amistosos 2009/2010 y así sucesivamente dependiendo de las temporadas. Así pues, mediante la acción "Nueva Competición" el usuario registrará una competición por medio de la introducción en el sistema de los parámetros de la misma, que en este caso serán el nombre de la competición, la temporada de la misma (año de inicio y año de conclusión), y si éste lo desea la normativa de la competición, no pudiéndo quedar este campo en blanco pero sí siendo posible "escapar" de él rellenándolo con un guión.

La edición de una competición consiste en seleccionar la misma y actualizar los datos que de ella estuvieran registrados.

Como restricción de funcionamiento de estas dos funcionalidades, se prohíbe bajo ningún concepto que al finalizar el rellenado de los datos de la competición y proceder a realizar la transacción correspondiente enj el sistema, la competición que se vaya a insertar o los nuevos datos de la editada

coincidan en nombre y temporada con otra competición existente, siendo esta violación de las reglas notificada por pantalla al usuario y uedando anulada la acción que éste quería emprender.

Una vez dada de alta una competición, se le podrá añadir un título, y sólo uno, a la misma desde la funcionalidad Palmarés previamente explicada y detallada en el módulo Mi Equipo, siendo el tiempo de vida máximo de este título el mismo que el de la competición, es decir, el título podrá ser eliminado desde la funcionalidad propia del palmarés para eliminar títulos, pero en el momento que se decida eliminar una competición que tenga asociado un título, el título se eliminara automáticamente sin opción ninguna a que el usuario lo conserve.

El borrado de competiciones es de las tres funcionalidades básicas de su gestión el que más complejidad, por lo amplio de su casuística de uso, entraña. Tal y como sucedía con la eliminación de los equipos (páginas 106 y 107), el borrado de una competición puede tener un efecto sencillo o simple, o por el contrario puede desencadenar una serie de eventos en cascada que afectarán a otros bloques del sistema.

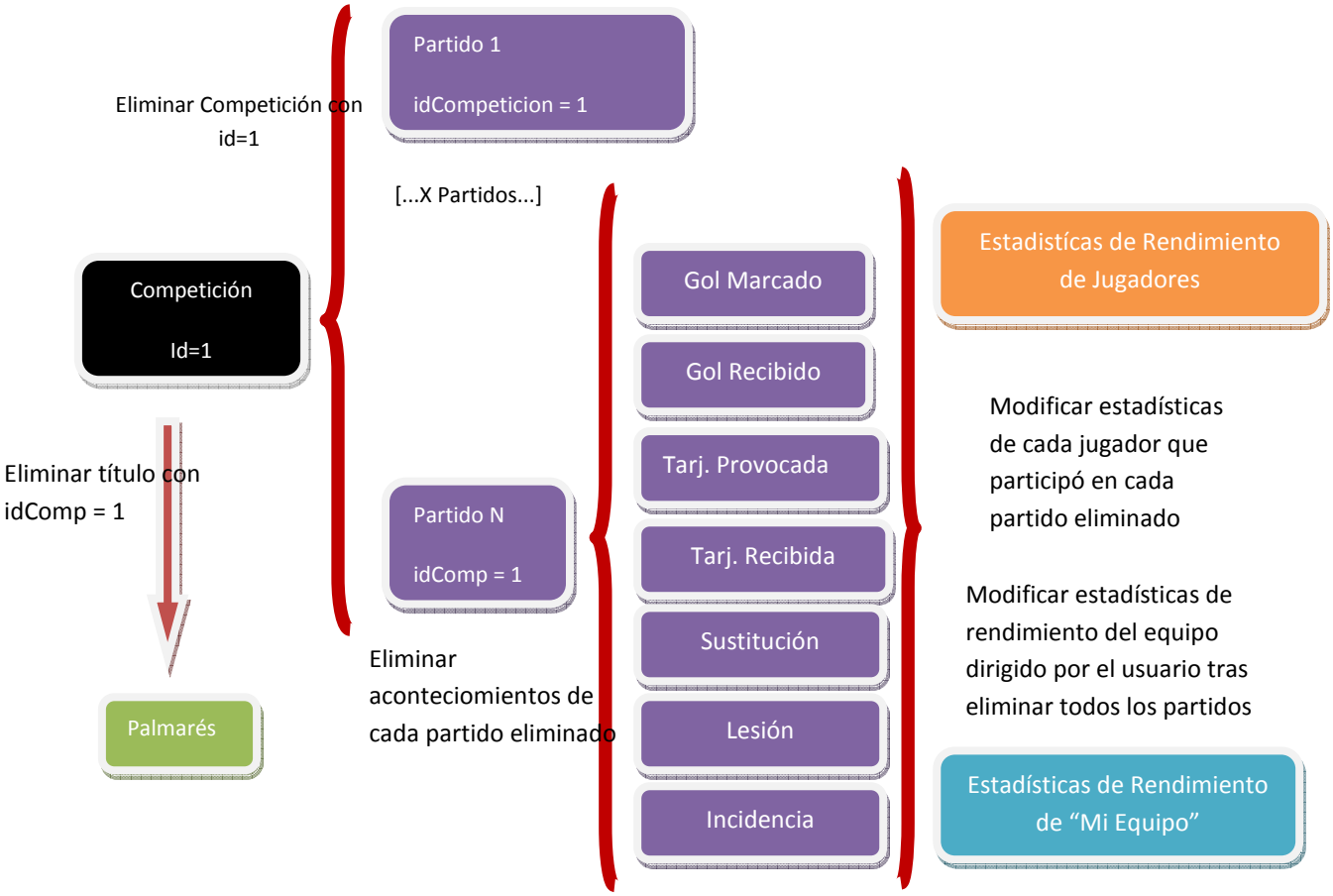
Por defecto, los partidos de la competición a eliminar no se conservan a no ser que el usuario indique al sistema lo contrario. La pantalla de eliminación ofrece un checkbox con el correspondiente mensaje explicando las opciones que el usuario tienen al eliminar la competición, así como las consecuencias de emplear una u otra (Ver Figura 14).

-Figura 14-

Pos defecto, la opción que se le da al usuario, es la de la eliminación en cascada por una cuestión de lógica, es decir, lo normal es que si un entrenador elimina una competición sea porque se ha equivocado en la misma y no haya muchos partidos en ella, y si el borrado se hace en una competición con múltiples partidos entendemos que el usuario desea que esta competición quede desestimada al 100%, al encontrar mucho sentido a la opción de eliminar una competición y conservar todo lo que esta trajo consigo.

Si el usuario elimina la competición tal y como se indica en la figura 14, el borrado de la misma se hará en cascada. El sistema buscará el código de identificación de la competición a eliminar, y una vez localizado, rastreará todos los partidos cuyo identificador que especifica a qué competición pertenecen coincida con el de la competición a eliminar. Para cada uno de estos partidos se procederá a su eliminación del sistema, analizándose en estos todas las eventualidades acontecidas y actualizándose las

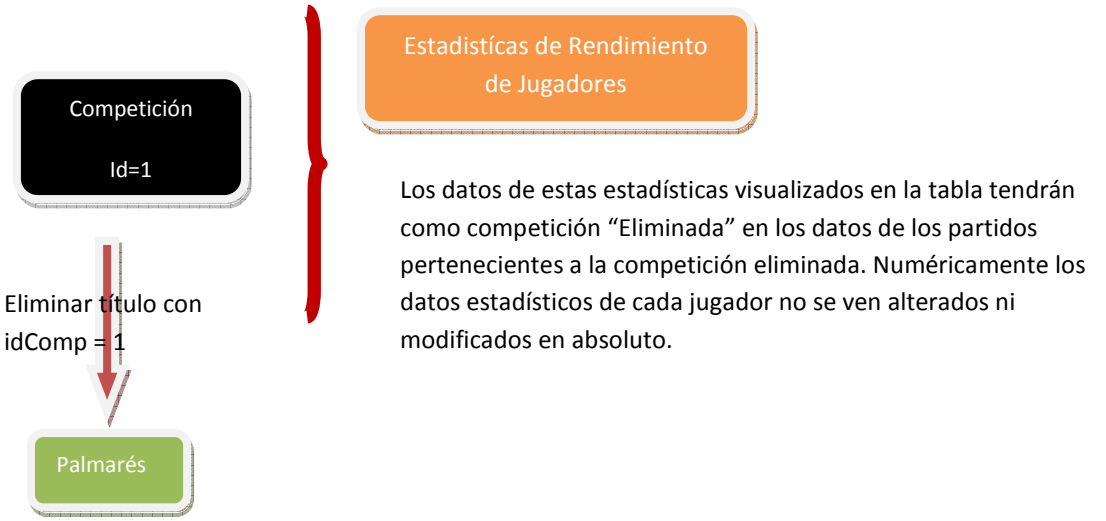
estadísticas personales de cada jugador que en él participó y actualizándose así mismo las estadísticas generales del equipo del usuario, es decir, si un jugador marcó dos goles en un partido de la competición eliminada, tras el borrado, el jugador tendrá en sus estadísticas dos goles menos, y las estdísticas del equipo tendrán dos goles menos a favor, un partido menos jugado, etc.



-Equema de borrado en cascada de competición-

Si por el contrario, el usuario seleccionase la opción de conservar los partidos, el procedimiento sería más sencillo desde el punto de vista de la programación al no tener que contemplar actualizaciones en tantos bloques diferentes del sistema. El sistema procedería a averiguar el código de identificación de la competición a eliminar y buscaría el título asociado a ella, si lo hubiese, en el palmarés y lo eliminaría. A continuación, la máquina en vez de eliminar la competición, realiza un proceso casi idéntico al del borrado de jugadores, es decir, no elimina la competición, la actualiza dándole como nombre y como temporadas de inicio y fin la palabra "Eliminada". Esta competición no volverá a ser visible ni accesible por el usuario, y el motivo de este proceder es que al visualizar las tablas de estadísticas concretas de los jugadores (Ver Partidos Jugados, Ver Minutos Disputados, ...), en los datos de partidos correspondientes a la competición eliminada de manera "simple", aparecerá como competición "Eliminada", "Eliminada", "Eliminada". Destacar de este proceder, que los partidos de esta competición jamás serán accesibles desde el histórico de partidos, ya que, como se detallará más adelante, el histórico contempla los partidos de competiciones no eliminadas, de modo que los datos de estos partidos solo podrán ser accedidos individualmente en las estadísticas de cada jugador, pero no se dispondrá de una visión global de cada partido de la competición eliminada. Asimismo, no se podrá eliminar estos partidos de la

competición borrada a no ser que se elimine el equipo del usuario, caso en que la base de datos quedará 100% vacía.



-Esquema de borrado sencillo de una competición-

A continuación, una vez explicada la sección de competiciones y teniendo como prerrequisitos el equipo del usuario dado de alta, algún equipo rival registrado, una competición que albergue el partido y al menos 11 jugadores registrados en el sistema, podremos proceder a comenzar un nuevo partido.

En la sección de partidos se muestra el resultado del último partido disputado, es decir, el último partido que fue comenzado y finalizadop desde la opción "Nuevo Partido". Si un partido es editado, no necesariamente tendrá que ser el útlimo registrado, y por tanto, su resultado no será el que figure en la pantalla de partidos.

Al seleccionar la opción "Nuevo Partido", el usuario debe asociar al mismo una competición, un rival, determinar si se juega como local o como visitante y una fecha para el mismo. Cuando todos estos datos estén correctamente introducidos, y teniendo en cuenta no violar la regla que indica que solo puede haber un partido por día, el usuario pulsará el botón "Comenzar Partido", y en ese momento el sistema añadirá a la tabla Partido una nueva fila con los datos introducidos, asignando automáticamente un identifocador único al partido que servirá par, a continuación vincular a él los eventos que en el partido tengan lugar.

Nada más comenzar, el usuario nada más tendrá opción a determinar la alineación inicial del mismo, y hasta que ésta no esté confirmada y sea correcta, es decir, que un mismo jugador no aparezca dos veces, no podrá seguir. Al confirmar la alineación, el sistema introduce en la tabla Alineación los datos de los 11 elegidos y vincula la alineación introducida al partido mediante el código identificador del partido que previamente comentábamos, y que, obviamente, quedará reflejado en la alineación como un campo de la tabla. Asimismo, justificamos este proceder, porque durante el partido hay una estructura que rige los acontecimientos del mismo, que es la que se muestra en la tabla 13, en la cual se muestra un fragmento de código donde se define y se declara esta estructura en el fichero mainwindow.h, ya que es en mainwindow.cpp donde se domina y maneja la misma. También en la tabla mostramos la inicialización de la misma en el constructor de la ventana (mainwindow.cpp).

```
// Declaración (MAINWINDOW.H)
   struct jugadorEnPartido {
      int idJugador;
      int minutoEntra;
      int minutoSale;
      bool estaJugando;
      bool haSidoSustituido;
   };

   jugadorEnPartido jP[18];

//Inicialización (MAINWINDOW.CPP)
   for (int i = 0; i < 18; i++){
      jP[i].idJugador = -1;
      jP[i].minutoEntra = 0;
      jP[i].minutoSale = 0;
      jP[i].estaJugando = false;
      jP[i].haSidoSustituido = false;
   }
```

-Tabla 13-

Al confirmarse la alineación inicial, las 11 primeras posiciones de jP toman los valores correspondientes de id de cada jugador que participa en el partido desde el inicio, y pondrán el campo está jugando de la estructura a true.

En este momento el partido ha comenzado, y cada eventualidad que en él acontezca se determinará como correct o incorrecto dependiendo de los datos de jP.

Explicaremos las posibles eventualidades que se pueden registrar en un partido de menor a mayor complejidad computacional y de programación.

En primer lugar se pueden registrar incidencias sin ningún tipo de problema, es decir, el usuario pìncha en el botón incidencias, y a medida que vaya anotando datos, estos serán actualizados en la base de datos, ya que no se puede añadir dos incidencias por partido, pero si se puede actualizar o “completar” una existente.

En segundo lugar, el módulo Lesión nos permite insertar lesiones a cualquier jugador, esté jugando o no, sin restricción alguna de tiempo, es decir, en cualquier minuto se puede lesionar, independientemente del resto de eventualidades. Para registrar la lesión, el usuario sólo tiene que introducir en la ventana correspondiente que jugador se ha lesionado, el minuto y la posible descripción de la lesión que éste pudiera padecer. La eliminación de la misma se llevará a cabo de una manera muy sencilla, ya que en la opción “Eliminar Lesión” se muestra tabla con las lesiones registradas durante el partido (cada jugador solo puede tener una lesión por partido) y el minuto de la misma, por tanto, introduciendo en el sistema el nombre del jugador cuya lesión será eliminada, el sistema la localiza fácilmente mediante el código identificador del jugador y el código identificador del partido en la tabla Lesion y la elimina sin mayor dificultad.

El resto de bloques dentro del partido ya tienen una mayor complejidad porque el tiempo y el hecho de que estén jugando, así como otros acontecimientos del partido pueden influir en que una acción que se desea tomar no sea correcta.

Comenzaremos por los goles. En la aplicación se pueden tanto añadir como eliminar goles marcados, así como goles recibidos.

Para añadir un gol a favor, el usuario tendrá que ir a Goles → Goles Marcados → Nuevo Gol Marcado. En ese punto, el usuario deberá introducir como datos de entrada el goleador, el minuto y una breve descripción del gol en caso de considerarlo oportuno, en caso contrario, con dejar el campo de comentarios con un guión, pero no vacío, es suficiente. Para confirmar el gol a favor y que éste sea añadido al sistema, la máquina analiza en primer lugar si el minuto en que se registra el gol es posterior al último en que se registró un gol anotado, recibido, una sustitución o una tarjeta, y en caso de no ser así, se rechaza la inserción dadno un mensaje de error al usuario. En segundo lugar, se analiza la estructura jP, busca en ella la id del jugador goleador y comprueba que el jugador está jugando en el momento del partido (jP[i].estaJugando=true), y si finalmente es así y todo ha discurrido en condiciones favorables, se cuentan los goles marcados hasta el momento, los goles recibidos en el partido, si el equipo es local o visitante, y tras verificar todos estos datos, inserta el gol marcado añadiendo en la base de datos el resultado provisional del partido en el momento en que este gol se registró, para facilitar la labor al entrenador y que éste pueda observar la importancia de los goles anotados or un jugador, ya que no es lo mismo marcar el gol que abre el partido, que el gol que hace que el equipo complete una goleada. Por el contrario, para eliminar un gol marcado, el usuario tendrá que ir a Goles → Goles Marcados → Eliminar Gol Marcado, y, observando la tabla en que se muestran los autores de goles y los minutos de los mismos en el partido, sdeberá seleccionar únicamente el último gol marcado para eliminarlo, ya que si no es el último que se registró, el sistema prohibirá terminantemente su eliminación.

Llegados a este punto se puede explicar que la aplicación prohíbe cualquier alteración en el orden temporal normal de los acontecimientos, es decir, un segundo gol no podrá se anotado antes que un tercero, ni podrá ser anulado el segundo gol cuando ya se dio validez al tercero. La aplicación está pensada para ser untilizada con un orden lógico en cuanto a tiempo, y cualquier intento de violación del orden temporal (del 0 al 90 o 120) será considerada como anomalía y por tanto, se mostrará error y se cancelará cualquier transacción que se quisiera iniciar.

Una de las secciones en las que la estructura jP cobra mayor relevancia es al registrar un gol en contra. El sistema registra, de cara a las estadísticas individuales del rendimiento de los jugadores, los jugadores que están en el campo en el momento de encajarse un gol. El gol encajado se registrará introduciendo un minuto y una descripción, y tras ello, el sistema, al igual que con los goles marcados, comprobará que no hay nada registrado tras ese minuto, y en caso de ser correcta esta premisa, la máquina recorrerá la estructura jP y para cada jugador que esté jugando añadirá su identificador en la fila correspondiente de la tabla gol recibido correspondiente al gol encajado que está siendo registrado, es decir, por ejemplo se encaja un gol en el minuto 78, la máquina recorre todos los jugadores en jP y analiza: jP[i].estaJugando=true, entonces toma jP[i].idJugador y la registra como uno de los jugadores en el campo en el momento de recibirse el gol; si por el contrario jP[i].estaJugando=false, ignora ese jugador y va al siguiente. A continuación, si todo es correcto, se actua tal como sucede con los goles marcados, es decir, se cuentan los goles marcados hasta el momento, los goles recibidos en el partido, si el equipo es local o visitante, y tras verificar todos estos datos, inserta el gol recibido añadiendo en la base de datos el resultado provisional del partido en el momento en que este gol se registró. Ni que decir tiene que por tanto, que el único gol que se podrá eliminar será el último recibido, y el motivo de esto, es, como se explicó, preservar la integridad temporal de los hechos durante el partido. Se eliminará accediendo a la sección Goles → Goles Recibidos → Eliminar Gol Recibido, y observando la tabla que indica todos los goles recibidos en el partido, simplemente habrá que introduir el minuto del último gol recibido y eliminarlo, o simplemente cancelar la acción y salir sin hacer ningún borrado.

Las tarjetas recibidas y provocadas tienen un funcionamiento similar, ya en ambos casos, para registrarlas, se tiene en cuenta el minuto de la tarjeta, el jugador, bien amonestado o que provoca la tarjeta, el color de la tarjeta y los comentarios que se consideren oportunos por parte del usuario.

La diferencia entre ambos tipos de tarjeta, reside en que en el caso de las recibidas el tratamiento es un poco más laborioso, ya que hay que comprobar además del tiempo, como ya se ha repetido anteriormente y a partir de ahora se obviará porque siempre hay que tenerlo en cuenta, las tarjetas previas del jugador, es decir, no se puede mostrar una segunda amarilla a un jugador que no ha recibido la primera, y si un jugador recibe dos amarillas o una roja directa hay que modificar su registro en jP modificando el campo estaJugado a false y el de minutoSale al minuto en que recibla tarjeta que le acarrea la expulsión del terreno de juego.

Para eliminar las tarjetas se exige la introducción del tipo de tarjeta, el minuto y el jugador que la provocó o la recibió., y en caso de poderse realizar la eliminación de la misma en condiciones lógicas y razonables, ésta se llevará a cabo, y en caso contrario, se cancelará la acción y se mostrará un mensaje con el motivo de la cancelación al usuario para que corrija la anomalía en caso de ser posible, o bien para que se dé cuenta de que está intentando cometer una violación flagrante de las reglas que rigen el comportamiento del programa.

En último lugar, podremos registrar sustituciones y eliminarlas. El funcionamiento es bien sencillo a nivel de lógica temporal, por lo que procedemos a expliacr el comportamiento de la aplicación basándonos en el código fuente de la misma. Cuando se registra o se elimina una sustitución, la estructura jP tiene un papel capital. Existen las siguientes restricciones: un jugador sustituido tiene que estar en el campo, y un jugador que ya salió del campo como producto de una sustitución no puede vvolver a entrar (campo de jP haSidoSustituido). Por tanto, al registrar una sustitución, el sistema recorre la estructura jP y se ciñe a los campos de los jugadores en que jP[i].idJugador y jP[j].idJugador se correspondan con el jugador que entra y el que sale. Si todo transcurre correctamente, al jugador saliente se le asigna un minuto de salida del campo y se ajustan los campos de estaJugando=false y haSidoSutituido=true. El jugador que entra, si todo es correcto, no tendrá su id registrada en nigún campo de la estructura jP porque aún no ha entrado en el campo, de manera que al recorrer la estructura, al primer ítem que contenga como idJugador=-1 se le signará la id del Jugador entrante, el campo estaJugando=true y el minutoEntra se ajusta al momento de la sustitución.

La eliminación se sustituciones sólo se podrá llevar a cabo siguiendo dos criterios: sólo se podrá eliminar la última sustitución, y a demás ésta no debe haberse producido antes de encajar un gol, ya que ese gol recibido tiene asociados unos jugadores en el terreno de juego que se verían desvirtuados y serían erróneos si se permitiera el borrado en dichas circunstancias.

Una vez llevados a cabo todos los acontecimientos del partido, el usuario puede terminar el mismo accionando “Finalizar Partido”. En ese momento, la máquina recorre el vector jP y a cada jugador que esté jugando y su minuto de salida (minutoSale) sea igual a 0, lo sobreescribe a 90, y calcula la diferencia entre minutoEntra y minutoSale. Una vez calculada esta diferencia y teniendo el código del jugador (idJugador) y el código del partido, inserta los datos del jugador, el partido y los minutos que éste disputó en el mismo en la tabla JugadorPartido. Repite el procedimiento para cada uno de los jugadores que participó en el partido.

Se pueden dar diversos errores debido a la naturaleza del propio dispositivo, ya que, por ejemplo, el usuario puede recibir una llamada en medio del uso de la aplicación o la batería se puede acabar durante el partido, o simplemente que el usuario salga de la aplicación si terminar el partido. Estos errores el programa los trata de la siguiente manera: Al entrar de nuevo al bloque de partidos, el sistema analiza el útkimo partido registrado y busca en la tabla JugadorPartido filas con el código identificador del partido para verificar que éste ha terminado. Al no haber terminado de manera normal, el sistema no encontrará ninguna fila que se corresponda a lo esperado en la tabla JugadorPartido, y por tanto mostrará un mensaje diciendo que el partido no ha terminado y que se puede continuar mediante la funcionalidad de editar partidos. Asimismo, destacamos que en la edición

de partidos lo único que no se puede editar es la alineación inicial, por tanto, si la aplicación se cerró antes de determinar la alineación inicial, el partido se vuelve inmanejable y el sistema motrará un mensaje en un label en la pantalla de partidos advirtiendo al usuario de que lo borre y lo comience de nuevo, ya que ni ha detectado las filas en JugadorPartido, ni ha encontrado una alineación asociada al partido.

La edición de partidos tiene como fin continuar con el tiempo de partido a partir del último minuto en el que un hecho relevante aconteció (tarjetas, goles, sustituciones).

El usuario tienen la opción de ver todos los partidos jugados y seleccionar el que quiere editar, y al confirmar, el sistema reestructura el vector jP de ese partido y permite al usuario continuar añadiendo y eliminao eventos tal y como ocurría y como se explicó para la opción “Nuevo Partido”. La única restricción adicional, como ya se comentó previamente, reside en la no posibilidad de editar la alineación inicial porque si hay eventos registrados sería inmanejable el tiempo de partido y as acciones que los jugadores protagonizan en el mismo. Al concluir la edición del partido mediante “Finalizar Partido”, el sistema en primer lugar elimina todas las filas de JugadorPartido que hubiera (en caso de existir porque en su momento el partido se finalizase correctamente y no por una situación de error), y registraría de nuevo las que fuesen pertinentes acorde al contenido final del vector jP tal y como se hacía al finalizar un partido desde la sección “Nuevo Partido”.

Estas dos funcionalidades son las más complejas de la sección de partidos ya que la posibilidad de eliminar partidos y el histórico de los mismos son bastante obvios.

Respecto a la eliminación de partidos, el usuario tiene la posibilidad de decidir que partido eliminar, y cuando toma la decisión, el sistema comienza a hacer un barrido por todos los jugadores que en él participaron (JugadorPartido). Para cada uno, resta los minutos diputados en el partido y el número de partidos jugados por el jugador desciende en un una unidad. Tras esta actualización, se eliminan las filas de la tabla JuadorPartido que referencian al que se está eliminando. La eliminación lleva un proceso en cascada que continua. El sistema continua analizando las tablas correspondientes a los eventos de partido (TarjetaProvocada, TarjetaRecibida, Lesion, GolMarcado y GolRecibido) y parar cada jugador que aparece en ellas protagonizando una acción, actualiza la información pertinente en las estadísticas del jugador, por ejemplo, si un jugador ha anotado un gol, se actualizará la información del jugador con golesMarcados=golesMarcados-1, y se eliminará esa fila de la tabla GolMarcado, y así con todas las tablas de acciones de partido donde hubiese algún contenido relevante al cual afectase el borrado del partido. Para concluir la transición, el sistema elimina automáticamente las filas de las tablas Sustitucion, Incidencia y Alineacion, ya que lo que en ellas se contempla, o bien no afecta a las estadísticas de los jugadores y del equipo (Incidencia), o bien ya se ha contemplado (minutosJugados en la tabla JugadorPartido, no importan para el caso Alineacion y Sustitucion porque su influencia ya ha sido previamente contemplada). La actualización de las estadísticas del equipo dirigido por el usuario se lleva a cabo automáticamente, ya que ésta se lleva a cabo por conteo de las filas en las tablas correspondientes para averiguar los valores numéricos, y por tanto, al hacer los conteos, encontrará menos filas ya que se han eliminado todas las que hacían referencia al partido eliminado.

Por último, el histórico de partido da acceso a consultar la alineación inicial, las sustituciones, los goles marcados, etc. de los partidos que, muy importante, estén registrados en el sistema y la competición a la cual pertenecen no haya sido eliminada. La implementación de esta funcionalidad es sencilla ya que se basa en consultas a la base de datos usando como referencia el código identificador del partido en cuestión (el sistema pide rival y fecha para averiguarlo), y buscando de cada tabla referenciada al partido mediante ese código, la información solicitada por el usuario. Destacar que pude haber partidos disputados hace tiempo, y de cuyas acciones pudo ser protagonista un jugador ya eliminado. Estas acciones serán mostradas en las tablas identficando a dicho jugador como “Jugador NPE” (No pertenece

al equipo), de manera que si hay dos goles a favor y uno lo marcó un jugador que ya no pertenece a la disciplina del equipo dirigido por el usuario, en ese gol aparecerá como autor “Jugador NPE” (Recordar borrado de jugadores en páginas 103 y 104).

4. FASE DE MIGRACIÓN AL DISPOSITIVO MÓVIL

4.1. NOKIA Qt SDK Beta

La fase de migración al dispositivo móvil ha sido bien sencilla debido a las facilitadas aportadas por el entorno Qt Creator Nokia SDK Beta. Este entorno fue lanzado el 27 de abril, apenas tres semanas antes de que este proceso de paso al dispositivo se llevara a cabo.

Descatacr antes de detallar el método empleado en sí, las dificultades encontrados en lo que respecta a las proporciones de los objetos en el dispositivo.

Hubo que hacer la migración en numerosas ocasiones debido a la falta de concordancia entre los resultados obtenidos en el simulador y los obtenidos en el dispositivo, sin comentar la versión PC en la cual las dimensiones de los ítems gráficos de la interfaz quedan claramente desproporcionadas. Desde mi punto de vista, un entorno tan bueno y tan amigable para trabajar, no puede permitirse imponer esta traba al programador en los tramos finales del desarrollo de la aplicación, ya que ralentiza más de lo deseado el proceso. Por otra parte, destacar que aunque el código es reutilizable para múltiples plataformas, lo cual es fantástico y un ahorro de tiempo y unas facilidades y posibilidades brutales de cara al futuro, no es menos cierto que el desarrollo gráfico de la aplicación no lo es, ya que dependiendo de la salida del mismo (PC o móvil, y según las dimensiones del móvil, ya que por ejemplo un N900 o un 5800 XpressMusic son iguales entre ellos ni son igual que un N97) hay que redimensionar y reajustar las dimensiones de la aplicación para que ésta quede con un aspecto apropiado. Se puede observar el resultado de teber las mismas dimensiones para PC y para móvil al contruir el proyecto, y se observará la poca armonía estética que presenta la versión PC, al haber quedado esta con las dimensiones apropiadas para visualizar la aplicación de manera correcta en el dispositivo móvil Nokia N97.

Destacar en este punto, un aspecto técnico derivado de los sensores del móvil, concretamente del acelerómetro que calcula la orientación del dispositivo para orientasy redimensionar la aplicación. La aplicación desarrollada está pensada para ir metiéndole datos y consultarlos, por lo cual, en la mayor parte del tiempo se entiende que el usuario deberá estar escribiendo con el teclado, y al hacerlo la aplicación se muestra en horizontal. Es por este motivo, que desde el código se ha tomado la determinación de forzar el mostrado de la aplicación siempre en horizontal independientemente de la orientación del dispositivo. Para este fin se han utilizado unas librerías especiales cuyo código se indica marcado en rojo en la tabla 14.

```
// CÓDIGO DE MAIN.CPP

#include <QtGui/QApplication>
#include <QtSql>

#ifdef Q_OS_SYMBIAN
#include <eikenv.h>
#include <eikappui.h>
#include <aknenv.h>
#include <aknappui.h>
#endif

#include "mainwindow.h"
#include "conexionBD.h"


int main(int argc, char *argv[])
{
   QApplication a(argc, argv);
   if (!createConnection())
      return 1;

   MainWindow w;

   // Código Específico Symbian
   #ifdef Q_OS_SYMBIAN

   CAknAppUi* appUi = dynamic_cast<CAknAppUi*> (CEikonEnv::Static()->AppUi());
   TRAPD(error,
   if (appUi) {
      // Bloquear la orientación de la aplicación a modo landscape (horizontal)
      appUi->SetOrientationL(CAknAppUi::EAppUiOrientationLandscape);
   }
   );
   #endif


   w.construirMaquinaEstados();
   w.show();

   return a.exec();
}

// Para incluir las librerías al proyecto ha habido que añadir información al final del fichero PFC.pro
// LÍNEA AÑADIDA A PFC.PRO
LIBS += -lcone -leikcore -lavkon
```

-Tabla 14-

El código en rojo es específico de Symbian, la aplicación detecta si se está ejecutando en un sistema operativo Symbian, y si es así, procede a bloquear el aspecto de la aplicación y ponerlo en modo de orientación Landscape (horizontal, hay dos modos más: Portrait (vertical) y Autoorientation). Decir también que se encontró una instrucción para realizar la misma acción en dispositivos Maemo, que no se puedo ejecutar ya que el dispositivo era Symbian y la compilación fallaba, pero que sustituiría a las líneas rojas dentro de la función main que era la siguiente:

```
w.setAttribute(Qt::WA_Maemo5LandscapeOrientation, true);
```

Pudiendo esta ser a su vez para vertical y auto-rotación las instrucciones:

```
w.setAttribute(Qt::WA_Maemo5PortraitOrientation, true);
w.setAttribute(Qt::WA_Maemo5AutoOrientation, true);
```

Llegados a este punto, procedemos a detallar el proceso de migración al dispositivo, para el cual nos hemos basado en el fichero index.html que se halla en C:\[Directorio de instalación de Nokia Qt SDK]\NokiaQtSDK\Symbian\readme. En él se nos detalla los pasos a seguir, que son los siguientes:

En primer lugar, se asumen como prerrequisitos cumplidos tener instalado el Nokia Ovi Suite y el cable USB para conectar teléfono y PC (todo ello disponible en la caja en la cual venía el dispositivo al adquirirlo).

A continuación, con el cable conectado y el Ovi Suite activo, hay que preparar el dispositivo móvil instalándole Qt y TRK, que es una aplicación necesaria para realizar la conexión entre el Qt Creator del PC y el dispositivo móvil. Parar instalar ambos conjuntos, el se debe acceder a Inicio → Nokia Qt SDK → Symbian Install Qt to Symbian device (para instalar Qt en el dispositivo), y tras ello, para instalar TRK, se debe acceder a Inicio → Nokia Qt SDK → Symbian Install TRK to Symbian device

Una vez instalado todo tendremos el dispositivo preparado para recibir la aplicación desde Qt Creator. Para que eso sea posible, tendremos que preparar TRK para habilitar la conexión vía USB entre Qt Creator desde el PC y el dispositivo. Para ello, seguir los siguientes pasos:

Desde el menú de programas en el dispositivo, lanzar TRK y rechazar la conexión mediante Bluetoooth (Figura 15). La aplicación presentará el siguiente aspecto (Figura 16).

-Figura 15-

-Figura 16-

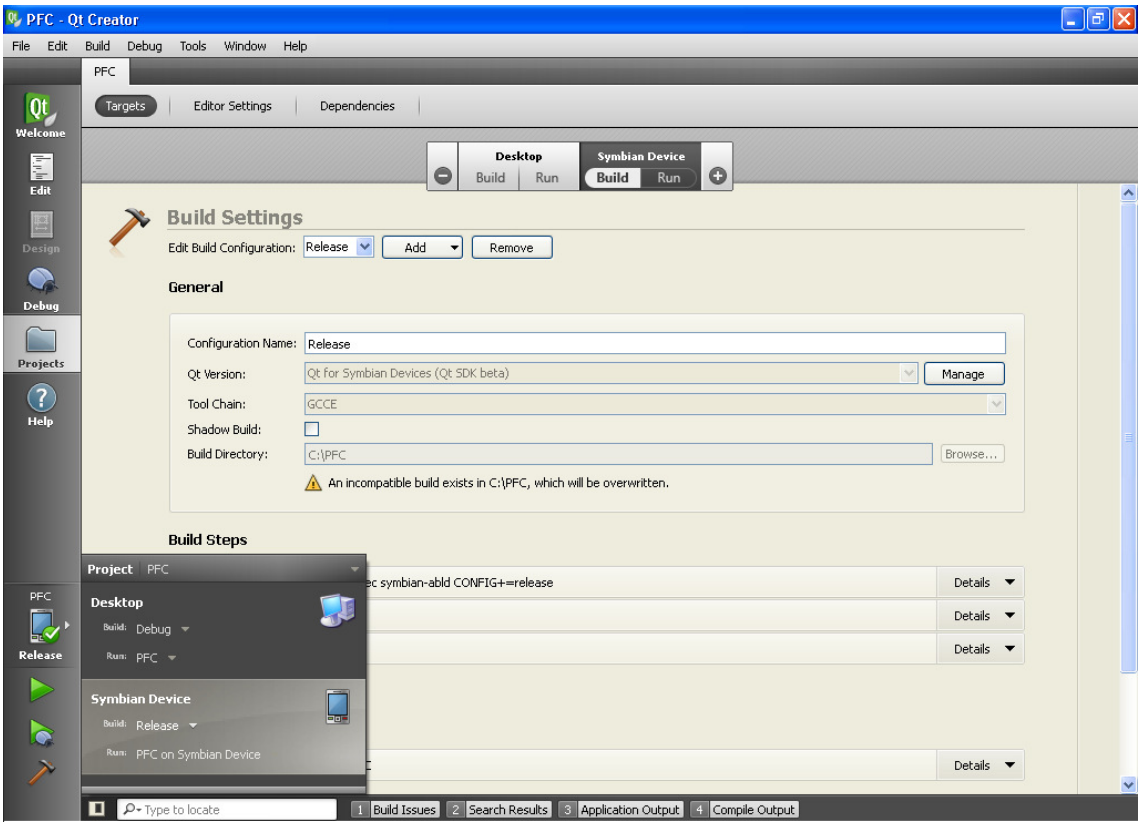
Usaremos la opción de menú para cambiar el modo de conexión a USB (Figura 17), quedando ahora la aplicación del modo que se indica en la figura 18.

-Figura 17-

-Figura 18-

En este momento, el dispositivo ya está preparado para recibir la información y ejecutar el programa, por tanto, falta configurar Qt Creator parar que determine como objetivo de la construcción de la aplicación el dispositivo Symbian.

Con el proyecto abierto en Qt Creator, nos vamos a la opción Projects, y desde allí, en el apartado Targets, añadimos pulsando la tecla (+) que aparece al lado de Qt Simulator e indicamos como “target” Symbian Device. A continuación, en la parte izquierda de la ventana del entorno, desde el cuarto botón empezando desde bajo, seleccionaremos que el objetivo será el Symbian Device en modo Release, y si todo esta bien conectado, el dibujo tendrá un “Check it ” blanco sobre fondo verde (Ver Figura 19), pero si no lo está saldrá una X blanca sobre fondo rojo.



-Figura 19-

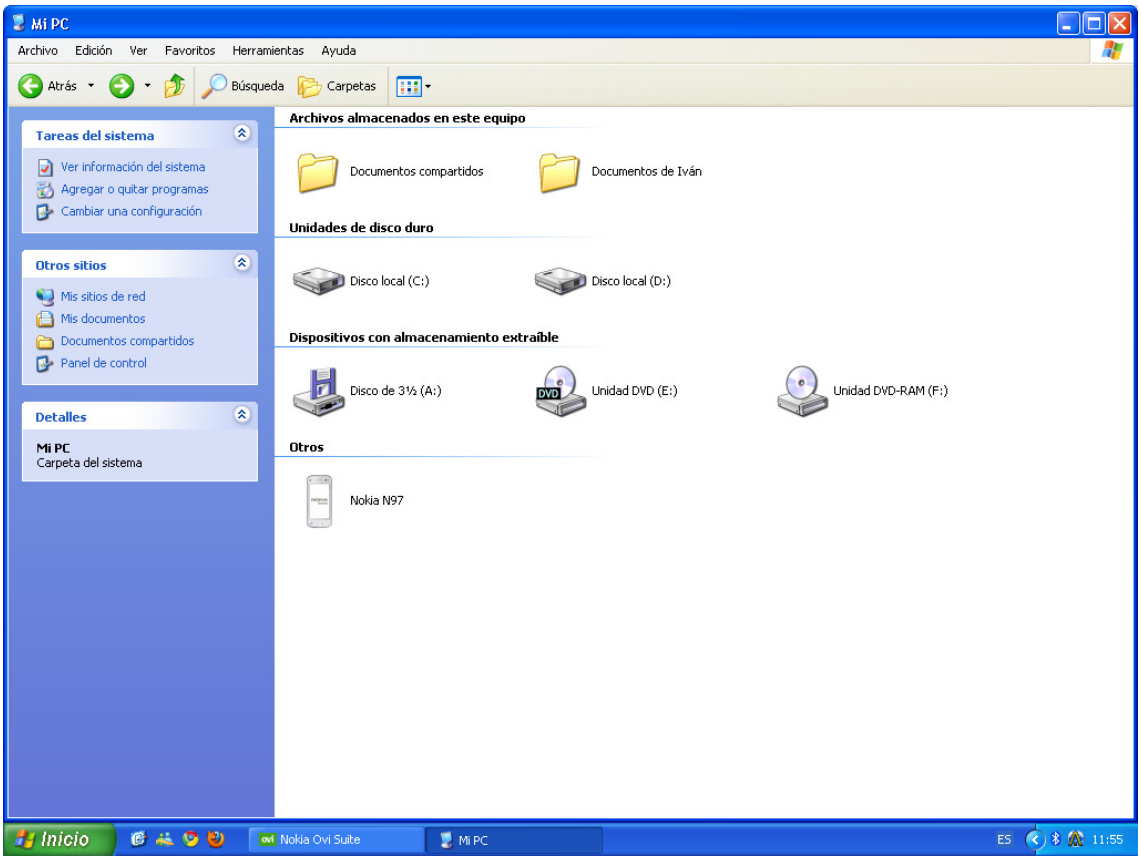
Despues de estos pasos segudos ya está todo preparado, y con lanzar la ejecución del programa, obtendremos como resultado que éste entrará en ejecución en el dispositivo móvil.

4.2. INSTALACIÓN EN EL DISPOSITIVO MÓVIL

Si todo el proceso explicado en el apartado anterior ha transcurrido con éxito, en el directorio del PC donde se encuentra el proyecto, se habrán creado multitud de archivos nuevos derivados del proceso, pero sobre todo, se habrá generado el archivo de instalación PFC.sis que servirá para distribuir la aplicación sin necesidad de conectar ya el dispositivo al Qt Creator y lanzar desde allí la aplicación.

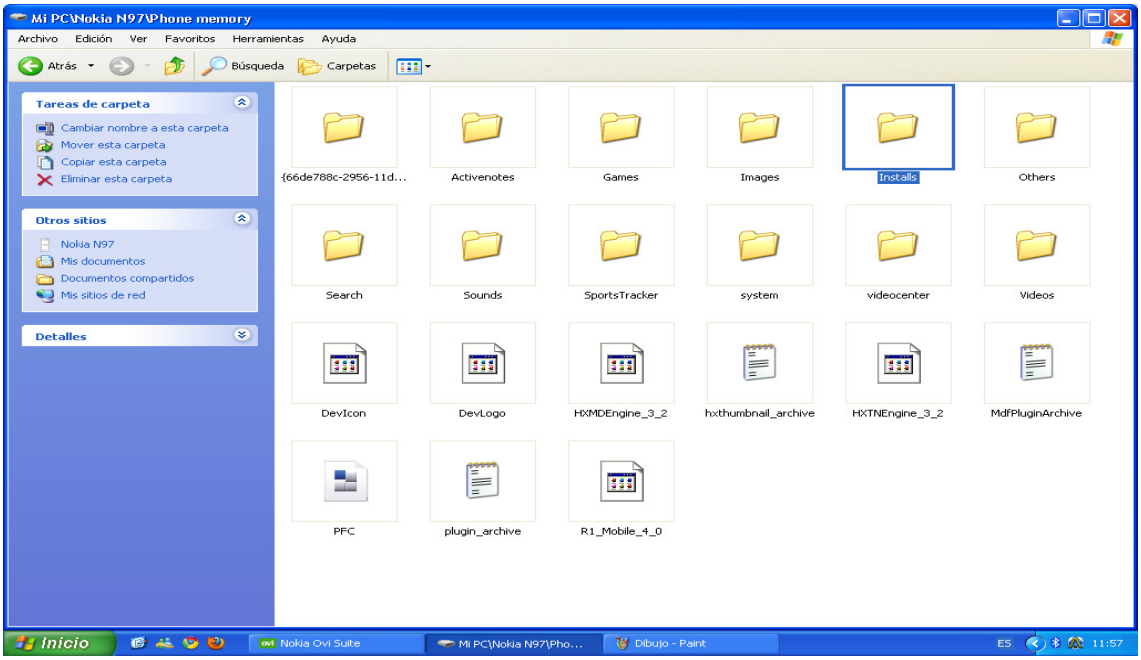
De este modo, disponiendo ya de PFC.sis, podemos distribuir la aplicación a cualquier persona y ésta, sin necesidad de tener Qt Creator ni nada por el estilo, solo con su cable USB y Nokia Ovi Suite, podrá realizar la instalación de la misma en su dispositivo. La instalación la tendrá que realizar de la siguiente manera:

Conectar su dispositivo al PC y acceder a Mi PC (Figura 20).



-Figura 20-

Accediendo a su Nokia N97, accediendo a Phone Memory (no a Mass Memory) encontrará un directorio llamado Installs (Figura 21).



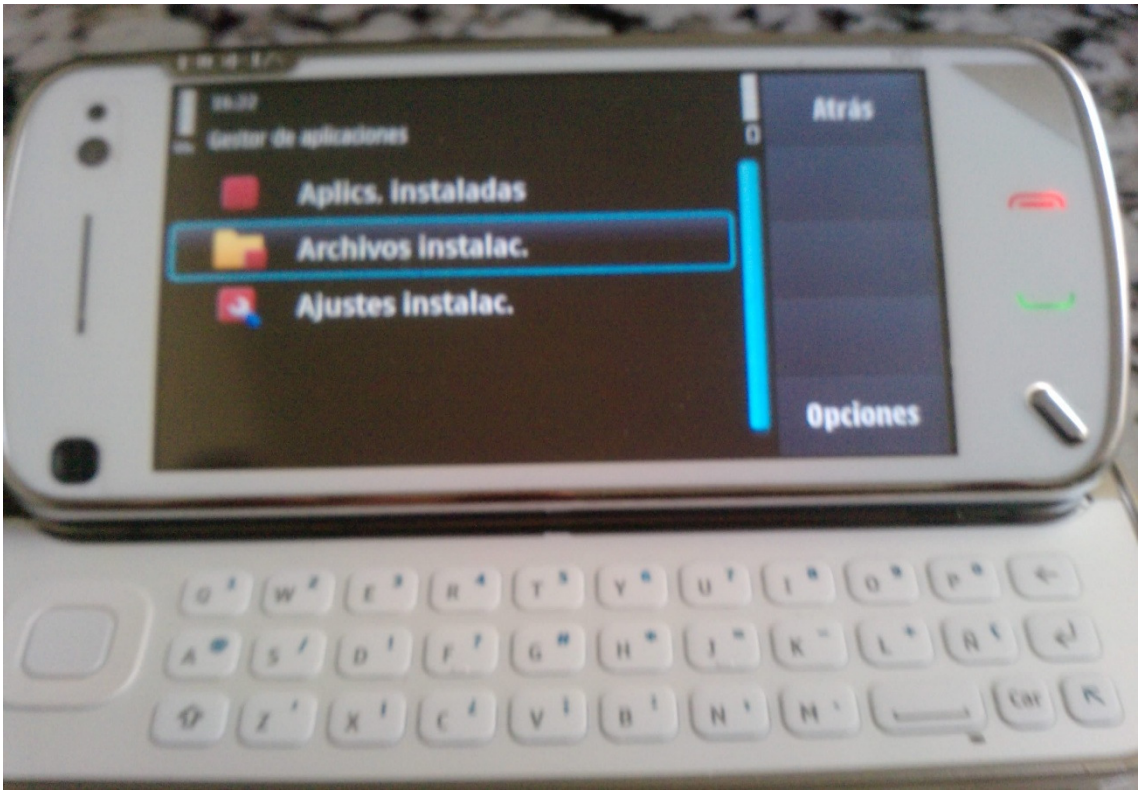
-Figura 21-

Dentro de ella tendrá que depositar el archivo PFC.sis y desconectar ya su dispositivo del PC. Toda la labor con el ordenador esta hecha, ahora tiene que instalarse desde el teléfono.

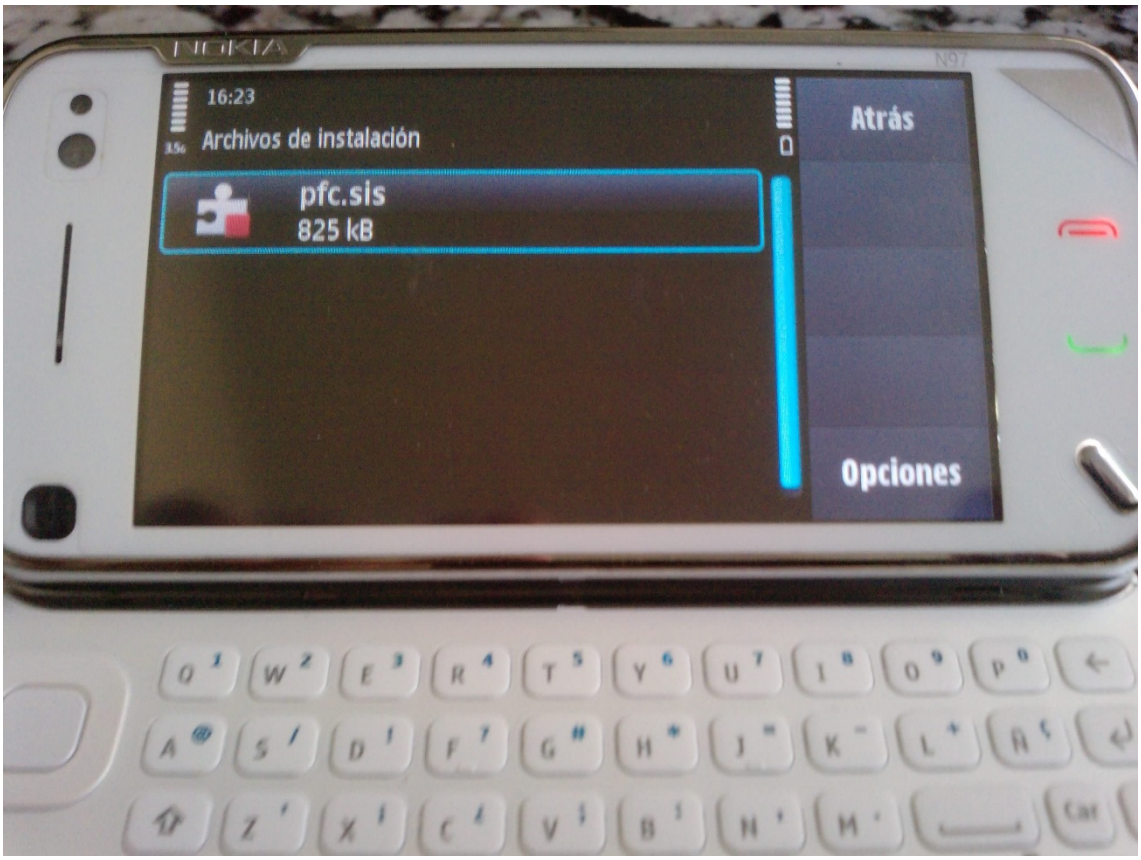
Centrado únicamente en el dispositivo, el usuario que está instalando la aplicación deberá seguir los siguientes pasos: Acceder a Ajustes (Figura 22) → Gestor de Aplicaciones (Figura 23) → Archivos de Instalacion (Figura 24), y allí encontrarán PFC.sis (Figura 25).

-Figura 22-

-Figura 23-



-Figura 24-



-Figura 25-

En este punto, desde el menú Opciones seleccionando “Instalar”, el programa quedará instalada en el dispositivo móvil y accesible para su lanzamiento y ejecución desde el apartado de Aplicaciones del dispositivo para comenzar a usar la aplicación.

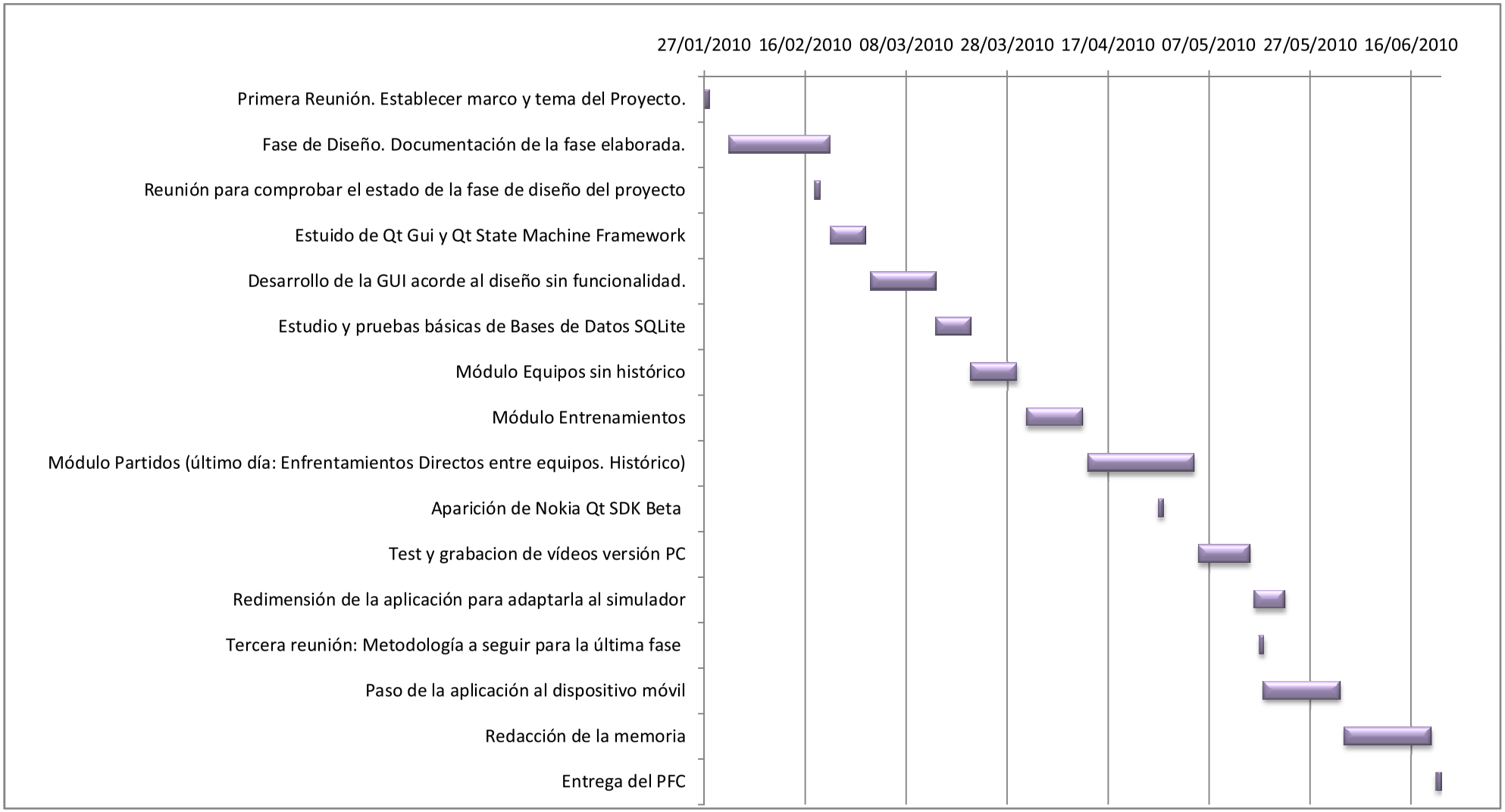
5. PLANIFICACIÓN

Llegados a este punto en que se ha descrito minuciosamente la aplicación desarrollada como objeto de este proyecto Final d Carrera, es momento de detallar la distribución temporal y las diferentes fases y momentos que se han afrontado parar completar la implementación.

A continuación se mostará una tabla con las tareas realizadas y las fechas que ésta comprendieron, así como la representación gráfica de la misma en un diagrama de Grantt.

Planificación de Trabajo según Tareas y Fechas

Tarea	Fecha Inicio	Duración	Fecha Fin
Primera Reunión. Establecer marco y tema del Proyecto.	27/01/2010	1	28/01/2010
Fase de Diseño. Documentación de la fase elaborada.	01/02/2010	20	21/02/2010
Reunión para comprobar el estado de la fase de diseño del proyecto	18/02/2010	1	19/02/2010
Estuido de Qt Gui y Qt State Machine Framework	21/02/2010	7	28/02/2010
Desarrollo de la GUI acorde al diseño sin funcionalidad.	01/03/2010	13	14/03/2010
Estudio y pruebas básicas de Bases de Datos SQLite	14/03/2010	7	21/03/2010
Módulo Equipos sin histórico	21/03/2010	9	30/03/2010
Módulo Entrenamientos	01/04/2010	11	12/04/2010
Módulo Partidos (último día: Enfrentamientos Directos entre equipos. Histórico)	13/04/2010	21	04/05/2010
Aparición de Nokia Qt SDK Beta	27/04/2010	1	28/04/2010
Test y grabacion de vídeos versión PC	05/05/2010	10	15/05/2010
Redimensión de la aplicación para adaptarla al simulador	16/05/2010	6	22/05/2010
Tercera reunión: Metodología a seguir para la última fase	17/05/2010	1	18/05/2010
Paso de la aplicación al dispositivo móvil	18/05/2010	15	02/06/2010
Redacción de la memoria	03/06/2010	17	20/06/2010
Entrega del PFC	21/06/2010	1	22/06/2010


27/01/2010
16/02/2010
08/03/2010
28/03/2010
17/04/2010
07/05/2010
27/05/2010
16/06/2010
Primera Reunión. Establecer marco y tema del Proyecto.
Fase de Diseño. Documentación de la fase elaborada.
Reunión para comprobar el estado de la fase de diseño del proyecto
Estuido de Qt Gui y Qt State Machine Framework
Desarrollo de la GUI acorde al diseño sin funcionalidad.
Estudio y pruebas básicas de Bases de Datos SQLite
Módulo Equipos sin histórico
Módulo Entrenamientos
Módulo Partidos (último día: Enfrentamientos Directos entre equipos. Histórico)
Aparición de Nokia Qt SDK Beta
Test y grabacion de vídeos versión PC
Redimensión de la aplicación para adaptarla al simulador
Tercera reunión: Metodología a seguir para la última fase
Paso de la aplicación al dispositivo móvil
Redacción de la memoria
Entrega del PFC

REFERENCIAS Y BIBLIOGRAFIA

BIBLIOGRAFIA

C++ estándar. Programación con el estándar ISO y la Biblioteca de Plantillas (STL).
Enrique Hernández Orallo, Jose Hernández Orallo, Mª Carmen Juan Lizandra.
Paraninfo. Thomson Learning.

Bases de Datos Relacionales.
Matilde Celma Giménez, Juan Carlos Casamayor Ródenas, Laura Mota Herranz.
Pearson Prentcie Hall.

Interfaces Gráficas de Usuario y Qt.
Dr. J. B. Hayet.

Getting Stated with the Nokia Qt SDK.
Documentación Web Nokia.

C++ GUI Programming with Qt4.
Jasmin Blanchette, Mark Summerfield.
Pearson Prentice Hall.

REFERENCIAS

(De acuerdo a las indicaciones de estándar ISO, afirmamos que a día 19 de Junio de 2010 estas referencias web citadas están disponibles en la red).

Desarrollando un status bar plugin para Maemo:
http://people.igalia.com/msanchez/talks/20080703-guadec-es-maemo-sb-plugin.pdf

Qt State Machine Framework:
http://qt.nokia.com/products/appdev/add-on-products/catalog/4/Utilities/qt-state-machine-framework

Qt4 y SQLite <<Desarrollo Libre>>: http://desarrollolibre.wordpress.com/2008/06/10/qt4-y-sqlite/

SQLite Foreign Key Triggers: http://justatheory.com/computers/databases/sqlite/foreign_key_triggers.htm

Free Qt Apps:
http://qt-apps.org/index.php?xsortmode=new&logpage=0&xcontentmode=4210x4211x4212x4213x4214x4220x4221x4222x4223x4224x4230x4231x4232x4233x4234x4235x4236x4240x4241x4242x4243x4250x4251x4252x4253x4254x4260x4261x4270x4271x4272x4273x4280x4281x4282x4283x4284x4285x4289x4296x4297x4298x4299&page=2

Tutorial de Qt: http://www.kdehispano.es/?q=content/tutorial-de-qt-cap%C3%ADtulo-5

Informe sobre librerías Qt: http://www.elai.upm.es/spain/Investiga/GCII/personal/vcorte/informeqt.PDF

QtSqlTableModel Class Reference: http://doc.qt.nokia.com/4.0/qsqltablemodel.html

Qt Sql Module Drivers: http://doc.trolltech.com/3.3/sql-driver.html

SQLite Integridad Referencial: http://sqlite-latino.blogspot.com/2008/12/integridad-referencial.html

Uso de SQLite: http://programadoresnocturnos.bligoo.com/content/view/437406/Uso-de-SQLite.html

Diseño de Bases de Datos Relacionales: http://usuarios.multimania.es/cursosgbd/UD4.htm

Modelo Entidad Relación: http://www.fdi.ucm.es/profesor/milanjm/BDSI0304/Tema02-ModER.pdf

Tutorial de SQL: http://www.asptutor.com/zip/sql.pdf

Información acerca de QTableView:
http://stackoverflow.com/questions/1230222/selected-rows-line-in-qtableview-copy-to-qclipboard

Qt 4. Model/View Programming: http://doc.qt.nokia.com/4.0/model-view-programming.html

Printing in Qt: http://doc.qt.nokia.com/4.2/printing.html

Forum Nokia. Tools Center: Nokia Qt SDK: http://www.forum.nokia.com/Library/Tools_and_downloads/

Noticias Meego:
http://www.portaltic.es/movilidad/software/noticia-nokia-intel-lanzan-primera-version-meego-20100531125912.html
http://www.imatica.org/bloges/2010/05/280564672010.html

Qt Mobility 1.0: http://doc.qt.nokia.com/qtmobility-1.0-beta/#sensors

Qt Simulating Sensors: http://doc.qt.nokia.com/qt-simulator-beta/simulator-sensors.html

Qt 4.6.2. Maemo 5 Rotation Example: http://doc.qt.nokia.com/qt-maemo-4.6/maemo5-rotation.html

Lock application orientation in Qt for Symbian:
http://wiki.forum.nokia.com/index.php/CS001517_-_Lock_application_orientation_in_Qt_for_Symbian

Code Example for Portrait/Landscape Autorotate Layout in Qt:
http://wiki.forum.nokia.com/index.php/Code_Example_for_Portrait_/_Landscape_Autorotate_Layout_in_Qt