UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

"ARQUITECTURA DE APLICACIONES ANDROID-WEB BASADA EN LA LIBRERÍA VOLLEY"

TRABAJO FIN DE GRADO

Junio - 2017

AUTOR: Edwin Molina Manzaneda

DIRECTORES: Miguel Onofre Martinez Rach
Hector Francisco Migallon Gomis

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1.- INTRODUCCIÓN

En este capítulo se explicará, justificará, se conocerán los objetivos y se explicará los límites del proyecto.

1.1.- Justificación del proyecto

Cuando se aprende a desarrollar programas o servicios, uno de los objetivos que se busca siempre, es que el código sea fácil de comprender para el desarrollador ya que es vital que la aplicación o servicio se pueda mantener y actualizar. Si el desarrollador genera código según la necesidad, sin orden alguno y con una mezcla importante de lenguajes de programación, obviamente generará grandes problemas en su mantenimiento, sobre todo si la aplicación es muy compleja.

Puede que ese problema en proyectos pequeños no sea tan importante, a veces una sola persona es capaz de mantener pequeños programas que no siguen ningún patrón de diseño. Un pequeño manual o simplemente añadir comentarios dentro de un código que esté suficientemente ordenado, pueden evitar muchos inconvenientes.

El verdadero problema ocurre cuando ese pequeño proyecto se convierte en algo grande o cuando muchas personas están trabajando al mismo tiempo en un proyecto que irá evolucionando constantemente.

Se tendría que separar la lógica del programa en muchas capas durante su desarrollo, pudiendo incluso reutilizar parte o toda esa lógica en otros proyectos, permitiendo así incluso trabajar a muchas personas de manera paralela y obtener resultados en el menor tiempo posible.

Entonces hace falta seguir una estrategia para conseguir todo ello. Por lo tanto, en este proyecto se pretende mostrar una estrategia basada en un modelo de capas con un ejemplo real y funcional en un prototipo desarrollado eficientemente.

1.1.1.- Antecedentes

En la carrera de ingeniería informática se aprende diferentes técnicas que ayudan a mejorar el proceso de creación de aplicaciones o servicios. Sin embargo, los lenguajes de programación se suelen aprender de manera separada, cada uno con su peculiaridad y complejidad. Por lo tanto si se desea desarrollar un programa o servicio fuera del ámbito académico, las necesidades obligan a utilizar y mezclar distintas herramientas y lenguajes de programación.

Además, en ámbitos generales se aprende a entender cómo los dispositivos trabajan en red aunque no siempre se es consciente de todos los recursos que realmente se pueden llegar a consumir.

Si nos centramos en Android, no siempre se aprende a desarrollar aplicaciones que trabajen en red, muchas de estas aplicaciones son gestionadas de manera local. Hoy en día es difícil tener aplicaciones que no utilicen los recursos necesarios para acceder a internet en algún momento, por ejemplo para poder registrar en un servidor los datos o la actividad que el usuario realiza en la aplicación y tener esos datos siempre disponibles.

1.1.2.- Soluciones

Para poder desarrollar una aplicación Android con acceso a Internet y que sea capaz de comunicarse con un servidor en internet, se ha elegido la librería Volley dentro de una serie de herramientas que se analizarán posteriormente, con el objetivo de mejorar eficientemente la cantidad de recursos que consume una aplicación Android en el dispositivo móvil por el simple hecho de enviar y recibir información desde internet.

Todas las aplicaciones eventualmente requieren ampliar su funcionalidad o aplicar algún tipo de mantenimiento. Una manera de ayudar a gestionar fácilmente el desarrollo de aplicaciones de alta complejidad, con flexibilidad para ser actualizadas, facilidad para realizar un correcto mantenimiento y tener el código (muchas veces en distintos lenguajes de programación) bien organizado, es siguiendo el modelo de capas Modelo Vista Controlador.

Siguiendo este patrón de diseño se desarrollará una aplicación web y una aplicación Android, cuyo objetivo es dar un servicio de información en tiempo real de partidos de football a varios usuarios, siendo este servicio un ejemplo claro de uso del modelo de capas (Modelo Vista Controlador) utilizando para su desarrollo distintos lenguajes de programación.

Por lo tanto se aplicará el modelo de capas en dos ámbitos de trabajo distintos. Uno para el desarrollo web con los lenguajes de programación Php, Ajax (Javascript), Html, Css y MySql. Y otro ámbito para el desarrollo de la aplicación Android con los lenguajes de programación Android (Java), Php y MySql.

1.2.- Objetivos

Para poder desarrollar la aplicación web y Android, se van a definir unos objetivos claros que permitan el correcto desarrollo de las aplicaciones.

1.2.1.- Objetivos Principales

El objetivo principal del proyecto es poder desarrollar un patrón de diseño de trabajo basado en el modelo de capas Modelo Vista Controlador, se explicará y utilizará este método de trabajo desarrollando un prototipo de una aplicación web y Android, aplicando

todo los conceptos aprendidos durante la carrera y descubriendo otros conceptos que podrán hacer posible el desarrollo eficiente de la aplicación.

Como segundo objetivo, hay que tener en cuenta que la aplicación Android trabajará en red, la aplicación Android se tiene que comunicar con un servidor, enviando y recibiendo información; se busca por lo tanto eficiencia a la hora de gestionar los recursos de red que tienen los distintos dispositivos que trabajan con el sistema operativo Android, la librería Volley será un gran aliado en ese aspecto.

1.2.2.- Objetivos Secundarios

En general, todas las actividades y servicios que se programan en Android se ejecutan en el hilo principal que interactúa con los componentes de la interfaz de Android (actualizando el estado de cómo se ven en pantalla). Sin embargo se pueden crear distintos hilos de ejecución y uno de los retos es controlar el flujo del programa entre los diferentes hilos eficientemente, comunicando y compartiendo datos entre los hilos de procesamiento creados para trabajar con el hilo principal de ejecución.

1.3.- Límites del proyecto

Como se ha explicado en los objetivos del proyecto, el modelo de capas Modelo Vista Controlador y la librería Volley, serán los principales protagonistas de este proyecto, sin embargo la aplicación Android y web final que siguen este patrón de diseño, serán parte de un prototipo que simplemente pretende demostrar su correcto desarrollo.

Tanto el servicio web como la aplicación Android van a tener una interfaz simple quizá no atractiva, cuya funcionalidad principal permitirá resolver los problemas de los usuarios.

Tampoco se pretende obtener beneficio económico del servicio desarrollado, a pesar de que el problema es real y existen clientes reales que lo podrían utilizar, la aplicación desarrollada es simplemente un ejemplo de uso del patrón de diseño que se propondrá más adelante.

Por lo tanto, con mucho más tiempo, se puede llegar a ampliar las funcionalidades y el desarrollo del servicio web y la aplicación Android más allá de lo que se hará en este proyecto, por ejemplo, utilizando distintas librerías y herramientas que permitan mejorar el aspecto visual, para hacer el producto final más atractivo.

También se puede ampliar el proyecto haciendo un estudio de usabilidad y viabilidad económica, que demostrará lo mucho que queda por mejorar en la aplicación final (web y Android) para poder publicarlo oficialmente.

Se puede incluso hacer que la aplicación se desarrolle de tal forma que se pueda utilizar en varias plataformas como iOS, Windows phone, BlackBerry OS, Firefox OS, Symbian, etc. Por ejemplo utilizando PhoneGap/Apache Cordova, una herramienta muy popular para crear interfaces en Html5, Css y JavaScript permitiendo que el resultado final de la compilación se ejecute en distintas plataformas.

Aunque se sabe que trabajar con ese tipo de herramientas no es eficiente comparado con la eficiencia que se puede obtener programando con las herramientas nativas, el resultado puede ser muy interesante desde otros puntos de vista. Por ejemplo, saber que la aplicación pueda trabajar en múltiples plataformas, permite expandir el número de posibles clientes. Otro ejemplo obvio es que se puede desarrollar aplicaciones mucho más rápido, ahorrando mucho tiempo invertido en el aprendizaje de varios lenguajes de programación.

2.- ANTECEDENTES Y ESTADO DE LA CUESTIÓN

Este capítulo se centrará en la búsqueda y análisis de aplicaciones Android para conocer hasta qué punto solucionan el problema del usuario dentro del contexto que se expondrá más adelante.

Por desgracia es muy difícil buscar información sobre las aplicaciones que indiquen qué arquitectura de diseño han utilizado cuando se han desarrollado, por lo tanto se analizará simplemente la funcionalidad de las aplicaciones existentes en el mercado, haciendo pruebas en un dispositivo móvil que funciona con el sistema operativo Android 4.4.2 (KitKat), con un procesador ARMv7 a 13000 MHz quad core, 500 MB de Ram, resolución de pantalla de 480x854 y 240 dpi.

2.1.- Situación actual de la empresa

Con el desarrollo del prototipo final de este proyecto, se pretende solucionar el problema que tienen los usuarios a la hora de mostrar la información en tiempo real de los eventos que ocurren en uno o varios partidos de football de cualquier categoría de cualquier torneo.

Actualmente los torneos de football de categorías profesionales disponen de las herramientas necesarias para difundir los resultados de los partidos de un torneo grande en tiempo real, como webs dedicadas, aplicaciones, revistas, periódicos o programas de tv. Sin embargo los torneos de football que no tengan esas herramientas no son capaces de ofrecer esos servicios de difusión de información.

Se sabe que muchas veces cuando se organiza un torneo de football y los equipos no disponen de recursos, normalmente los organizadores del torneo suelen recurrir a un tablón de anuncios con los resultados de los partidos, suele ser esa la fuente principal de difusión de información, pero esa fuente sólo se muestra cuando la jornada de partidos finaliza, no siendo inmediato, muchas veces la información se publica varias horas o varios días más tarde.

Sin embargo existe constancia de que actualmente algunos equipos comparten los resultados de sus partidos en las redes sociales como Facebook o Twitter, pero no todos los equipos hacen eso y no se puede llegar a obtener toda la información del torneo de manera centralizada y actualizada.

Se sabe también que los partidos de football disponen de una mesa de control, con autoridades que controlan todas las incidencias de los partidos, la idea es que esas autoridades formen parte de la difusión de la información, registrando las incidencias y eventos del partido de manera oficial y al mismo tiempo podrán utilizar la aplicación

Android para difundir la información, así los demás usuarios podrán ver en sus dispositivos móviles todas las incidencias tan pronto suceda algo.

En este proyecto será necesario disponer de dispositivos móviles (smartphones) que funcionen con Android y que esos dispositivos puedan acceder a internet.

2.2.- Herramientas disponibles en el mercado

La fuente de búsqueda de aplicaciones es Google Apps [1], se tomará en cuenta que las aplicaciones Android sean gratuitas y se valorará el top 5 de aplicaciones según la opinión de los usuarios (mejores ratings).

Existen muchas aplicaciones Android que permiten la difusión de información de torneos profesionales, como Live Football, Live Streaming Football, Football Live Score, etc. Todos acceden a unas grandes bases de datos que ofrecen los resultados de las grandes competiciones profesionales de football del mundo.

Por lo tanto, al margen de esas aplicaciones y para este caso lo ideal es buscar aplicaciones que permitan personalizar y crear todo tipo de torneos (profesionales o no profesionales), para luego difundir la información en tiempo real de los eventos de los partidos a todos los usuarios públicos.

Según esos criterios, las aplicaciones Android a analizar son:

- Best Tournament Manager
- The Tournaments Manager
- Bracket Maker & Tournament App
- Football Tournament Maker
- Fixture Maker

2.2.1.- Best Tournament Manager

Según el autor se puede administrar todo tipo de torneos con diferentes tipos de combinaciones: liga single, liga de múltiples grupos, preliminares y eliminatorias dobles. Se puede combinar un primer grupo con una o más rondas y un segundo grupo para llegar a una final.

Se puede elegir el deporte y cuantos equipos por grupo podrán pasar a la siguiente fase de un torneo real. Se puede personalizar los equipos para luego crear las tablas de encuentros de manera manual o aleatoria y anotar los resultados. Se puede crear torneos con un gran número de participantes y exportar posiciones, grupos o resultados para compartirlos en las redes sociales.

2.2.1.1.- Prueba de la aplicación

Se ha comprobado que la aplicación funciona sin conexión a internet para poder crear y administrar torneos, a continuación se personalizará un torneo de football con participantes aleatorios para jugar un torneo de un solo grupo de todos contra todos. En la figura 1 se puede apreciar el menú principal de la aplicación.

Figura 1: Best Tournament Manager pantalla principal

Antes de crear el torneo es recomendable crear los participantes, en la opción "My Teams" se puede elegir "custom Participants" para crear participantes personalizados, se pone el nombre, ranking (no importante) y se puede elegir un icono o imagen que represente al participante.

Como ejemplo se va a crear un torneo de prueba llamado "football sub 15", permitiendo personalizar el nombre, el tipo de torneo (Soccer, basket, volley, ice hockey o tenis de mesa), también permite añadir un icono representativo del torneo.

El siguiente paso es poner el escenario del torneo (grupos, preliminares o doble eliminación), elegir el número de participantes, el número de grupos, el número de veces en los que cada participante se enfrentará con otro (sólo un encuentro, dos o tres encuentros) y el número de finalistas.

Finalmente se puede elegir los participantes, pudiendo ser totalmente personalizados o también se puede elegir el participante de una lista predefinida con los nombres de distintos países o elegir "other logos" para elegir otros participantes de competiciones conocidas.

Una vez personalizado las características del torneo la aplicación permite crear automáticamente los encuentros (fixture) de manera aleatoria, en la siguiente pantalla se verá la tabla de puntos del torneo, como se ve en la figura 2.

Figura 2: Best Tournament Manager pantalla del torneo

Para añadir los resultados de los encuentros disputados, se debe elegir la opción "View Rounds", un partido y poner el resultado correspondiente. Una vez añadido se vuelve a la pantalla principal donde están los resultados de todos los participantes del grupo.

El torneo creado se puede editar, copiar, eliminar y está disponible offline. Para finalizar el torneo, se puede elegir "Next" y mostrará en pantalla el equipo que ha ganado el torneo como se aprecia en la figura 3.

Figura 3: The Best Tournament Manager pantalla fin del torneo

El torneo se puede exportar como una página Html o una imagen, sea cual sea la opción elegida, se puede luego imprimir o compartir, es recomendable elegir una imágen ya que toma una imagen completa del torneo y la comparte, exportar como Html no hace nada (La exportación en Html presenta problemas en su funcionamiento).

Si se elige compartir, se podrá ver los distintos modos de compartir que ofrece Android (según las aplicaciones instaladas), en la prueba se ha optado por el uso de gmail, cabe destacar que el mensaje estará en cola mientras no haya conexión a internet, una vez conectado a internet se enviará el mensaje con una imagen de los resultados del torneo.

2.2.2.- The Tournaments Manager

Según el autor esta aplicación está pensada para torneos y partidas amateur disputadas entre amigos como: King of the hill, torneos de FIFA, PES, etc. Se puede retomar las partidas en cualquier momento, se puede guardar los resultados y almacenarlos en

históricos para ver las estadísticas de los partidos, se puede también compartir la información por Facebook y admite hasta 32 jugadores por torneo.

2.2.2.1.- Prueba de la aplicación

Comienza introduciendo el nombre del usuario, la aplicación funciona online y offline. Como se ve en la figura 4, desde el menú se puede crear torneos, ver torneos y cambiar ajustes.

Figura 4: The Tournaments Manager pantalla principal

Para crear un torneo bastará con poner el nombre y el número de participantes, si se ha creado participantes en otros torneos, se puede elegir y añadir esos mismos participantes.

Los modos de juego son:

- King of the hill: permite elegir partidas uno a uno donde el vencedor compite con otro participante y el nuevo vencedor compite con otro hasta que no quede más que un participante.
- League one leg: una liga típica (round-robin) donde todos se enfrenta contra todos al menos una vez.
- League two leg: igual que league one leg pero con una segunda vuelta.
- Brackets: es la típica partida en el que el vencedor de una partida compite con el vencedor de otra partida, requiere un número de participantes igual a 4, 8, 16 o 32.

Cada vez que se accede a ver el torneo creado, como se ve en la figura 5, se observa siempre las estadísticas del torneo, sea cual sea la modalidad elegida. En la misma pantalla están las opciones de borrar torneo, ver el historial de partidos, volver al menú principal y muestra el siguiente partido programado para poder introducir el resultado de ese partido.

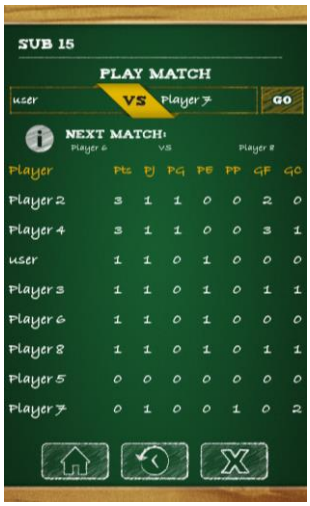


Figura 5: The Tournaments Manager pantalla del torneo

Es una aplicación sencilla, fácil de entender y utilizar, ideal para partidas rápidas entre pocas personas; lo malo es que no posee manera de compartir los resultados del torneo con otras personas.

2.2.3.- Bracket Maker & Tournament app

El autor advierte que la aplicación necesita conexión a Internet para gestionar los torneos. La aplicación permite: organizar competiciones deportivas, crear perfil de organizador y gestionar las ligas y torneos, disponible para todo tipo de deportes, sin límites de equipos ni jugadores, permite el control sobre las estadísticas de jugadores y equipos, la generación automática de calendarios, la gestión de fechas y lugar de juego.

2.2.3.1.- Prueba de la aplicación

Es necesario tener acceso a internet, el primer paso es crear un usuario para poder tener siempre disponibles los datos de los torneos y competiciones creadas de cada usuario, en la figura 6 se puede ver el menú del usuario registrado en la aplicación.

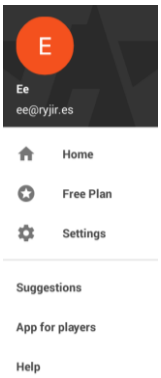


Figura 6: Bracket Maker & Tournament App menú de usuario

Para crear el torneo basta con nombrar el torneo y elegir una amplia gama de deportes (baseball, beach tennis, bowling, chess, curling, etc), juegos (Call of duty, Counter Strike, Fifa, Dota, etc) y otras actividades (1 vs 1, equipos o parejas, etc).

Distintos modos de competición:

- League mode: una competición (round-robin) de todos contra todos.
- Playoff mode: participante que pierde es eliminado.
- Divisions: competición en diferentes divisiones (diferentes ligas) con la opción de jugar un playoff para definir un campeón de cada división.
- Groups + Playoff: varios grupos con un playoff para definir un campeón.
- Manual mode: donde se indica las preferencias de la competición, puede ser League (round-robin) o playoff (knock-out).

Una vez creado el torneo se debe añadir los participantes, que pueden ser importados o creados manualmente. La versión gratuita permite crear los equipos pero sin poner logos, sólo deja poner el nombre del equipo, hay una pestaña para ver la tabla de encuentros creada automáticamente en el que se puede añadir los resultados de cada partido. En la figura 7 se puede apreciar la tabla de resultados de una competición.

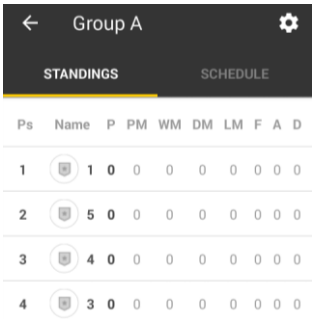


Figura 7: Bracket Maker & Tournament App pantalla del torneo

Se puede ver las estadísticas de cada torneo pero sólo en la versión PRO, existe una opción que permite ver las competiciones en la web gratis por 15 días, como se puede apreciar en la figura 8. La versión PRO no tiene límite de días, se puede administrar los torneos tanto desde la web como en la aplicación Android, pero está disponible sólo para el usuario registrado.

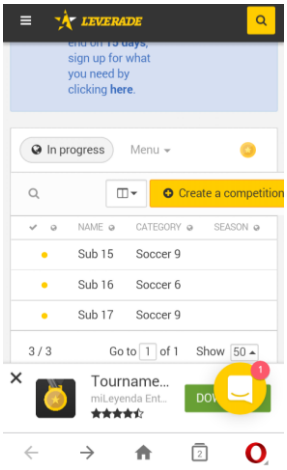


Figura 8: Bracket Maker & Tournament App web del torneo

2.2.4.- Football Tournament Maker

Según el autor se pueden crear torneos en modo round robin, knockout, crear un sitio web del torneo, ver estadísticas, hacer backup y restauración, introducir los goles de los jugadores y los puntajes de los partidos.

2.2.4.1.- Prueba de la aplicación

La aplicación funciona online y offline. Se crea el torneo introduciendo el nombre, eligiendo la modalidad del torneo (round-robin o playoff), dispone además de opciones avanzadas para personalizar las reglas de los partidos, como el número de veces que se enfrentarán los equipos, si se jugará sólo un tiempo o dos, etc.

Luego se elige el número de equipos, creando una lista con los nombres de los equipos provisionales. Una vez creado el torneo, la primera pantalla está formada por los encuentros creados automáticamente. Para registrar el resultado de cada partida, basta con elegir un encuentro de la tabla y añadir el resultado.

Existen ventanas para ver los resultados del torneo, el ranking o ver las reglas elegidas para el torneo, como se aprecia en la figura 9.

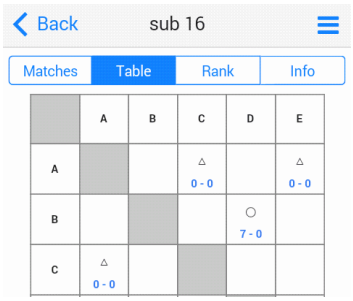


Figura 9: Football Tournament Maker tabla de resultados

Se puede compartir la tabla o el ranking del torneo eligiendo cualquiera de los medios que ofrece Android (email, hangouts, messaging, bluetooth, drive, skype, etc).

Se puede también crear una web de manera automática y gratuita, se comparte el link para poder ver la tabla de resultados y la tabla de ranking del torneo como se aprecia en la figura 10.

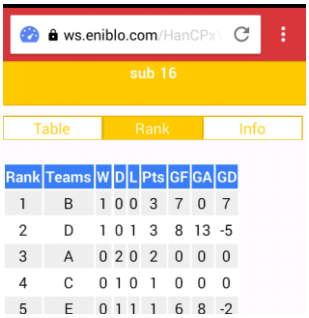


Figura 10: Football Tournament Maker web del torneo

2.2.5.- Fixture Maker

Según el autor la aplicación es ideal para partidos de football y partidas de la consola o pc, se puede crear un torneo con tantos equipos como uno quiera.

2.2.5.1.- Prueba de la aplicación

Se crea un torneo poniendo el nombre y eligiendo el tipo de torneo: liga o torneo que se juega a single match (un solo partido entre los participantes) o revenge (dos partidos entre los participantes). Los participantes se añaden manualmente escribiendo el nombre de cada equipo.

La primera pantalla que se ve una vez creado el torneo sirve para programar los partidos que se crea automáticamente, se modifica los resultados desde esa pantalla y se puede ver las estadísticas generadas por los resultados de los partidos.

Se puede añadir manualmente los nombres de los jugadores para indicar los goleadores del torneo. Se puede compartir los torneos a través de las redes sociales Facebook y Twitter. La aplicación está disponible online y offline. Se puede apreciar en la figura 11 la tabla de resultados del torneo creado y las opciones para compartir la información del torneo.

Figura 11: Football Tournament Maker opción para compartir torneo

2.3.- Resumen

A continuación se muestra la tabla 1 donde se ha hecho una comparativa con las ventajas y desventajas que se han ido encontrando durante las pruebas y análisis de las 5 aplicaciones:

Tabla 1: Ventajas y desventajas de las aplicaciones analizadas

NOMBRE DE LA APLICACIÓN	VENTAJAS	DESVENTAJAS

Best Tournament Manager	Alta personalización para crear equipos y torneos, desde el logo y todo tipo de descripciones necesarias, también se puede elegir torneos por defecto, asociado a torneos populares como competición por países, champions league, etc. Se crea automáticamente las partidas, según se elija el tipo de modalidad. Se puede compartir la información del torneo en forma de imagen a través de las opciones para compartir ofrecidas por Android. Funciona offline y online. No consume mucha memoria interna, 4.38MB de instalación.	La aplicación afirma que se puede compartir el Html del torneo para poder visualizarlo en la web, cosa que no se ha podido hacer, la aplicación no genera ningún código para compartir, por lo tanto es difícil difundir la información al público en general, ya que se tiene que elegir a quién compartir la información persona a persona. Se necesita ser usuario Pro para poder agregar amigos que puedan administrar la aplicación, a un coste de 1.80 €.
The Tournaments Manager	Alta facilidad para crear torneos y sus participantes, cuya creación no toma mucho tiempo. Fácil e intuitiva administración de partidas y torneos. Funciona offline (online no es necesario). No consume mucha memoria interna, 8.13MB de instalación. Es gratuito y no tiene costes extra.	No se puede compartir la información de los torneos desde la aplicación a otros usuarios.
Bracket Maker & Tournament App	La creación de torneos y sus participantes es fácil e intuitivo, su creación tampoco lleva mucho tiempo. Fácil e intuitiva administración de partidos y torneos. Permite compartir información de los torneos, instalando previamente otra aplicación "LEVERADE-Real Play".	El autor no ha especificado el coste extra que la aplicación tiene para personalizar los torneos, partidos y ver estadísticas. Si los torneos son grandes, aunque deja crear todo libremente después de 10 partidas se tiene que hacer un pago extra de 8€ mensuales, un costo demasiado elevado para lo que ofrece.

	No consume mucha memoria interna, 15.25MB de instalación.	
Football Tournament Maker	Es fácil la creación de torneos y sus participantes pero no muy intuitivo. Crear torneos lleva más tiempo que las otras aplicaciones. Fácil e intuitiva administración de partidas y torneos, es totalmente gratuita y no tiene límites. Funciona online y offline. Permite compartir información del torneo por los medios habituales de Android, pero también permite crear una web automáticamente con la información del torneo, basta con elegir la opción y se crea la web automáticamente, sólo se genera el link para ver la web, que se puede compartir con todo el mundo.	La aplicación tiene fallos de diseño, no sale nada en las listas para elegir el número de equipos y de vez en cuando desaparecen las opciones de las listas para elegir el nombre de los equipos. Poco personalizable. Consume mucha memoria, 69.48MB de instalación.
Fixture Maker	La creación de torneos y sus participantes no suele llevar mucho tiempo y es fácil e intuitivo. También es fácil e intuitiva la administración de partidas y torneos. Totalmente gratuita, sin limitaciones. Funciona offline, online no es necesario a menos que se quiera compartir la información del torneo por redes sociales. Consume poca memoria, 3.09MB de instalación.	Poco personalizable. No se puede compartir información de los torneos fácilmente al público en general, sin embargo se puede compartir por redes sociales (Facebook y twitter).

2.4.- Valoración

Tras ver y analizar el funcionamiento de las aplicaciones instaladas, una de las carencias más importantes es que no es fácil compartir información de los torneos creados al público en general.

Además, en la mayorías de estas aplicaciones, solo un usuario se encarga de toda la creación y administración de los torneos, siendo así complicado informar en tiempo real, todos los encuentros si se disputan al mismo tiempo en distintos sitios, por lo tanto es necesario la ayuda de varios usuarios para poder informar de los distintos incidentes en los partidos de los distintos torneos creados.

Estos tipos de carencias serán los puntos a solucionar con el prototipo final de la aplicación diseñada en este proyecto.

3.- HIPÓTESIS DE TRABAJO

En este capítulo se exponen las tecnologías elegidas para poder trabajar en este proyecto. Se propone una forma de trabajo basado en el modelo de capas aplicada a dos grandes grupos, un grupo es para los servicios web y el otro es para la aplicación Android. Por lo tanto, todas las tecnologías que se van a exponer representan las opciones más importantes que actualmente están disponibles.

Primero se analizarán las herramientas disponibles para poder aplicar el modelo de capas a las aplicaciones Android, luego se analizarán todas las herramientas disponibles para poder desarrollar los servicios web siguiendo el modelo de capas.

3.1.- ¿Por qué Android?

Una de las razones para elegir Android como sistema operativo sobre el cual desarrollar la aplicación, es la inmensa popularidad actual de la que goza.

Para indagar en ello el informe "La Sociedad de la información en España 2016" de Ariel y Fundación Telefónica, presenta datos estadísticos y análisis de las comunicaciones y servicios digitales en España, el informe se publica anualmente y está disponible en la web de la fundación telefónica [2], cuenta con información de las unidades de negocio de Telefónica y la contribución de las Comunidades Autónomas.

El informe presenta comparativas con respecto a otros años para analizar lo que está sucediendo y definir las tendencias de futuro de las tecnologías móviles. Es un informe muy variado con muchos puntos de análisis, se tendrá en cuenta lo más destacado para este proyecto sobre las nuevas tendencias de 2016, datos de conectividad, acceso y terminales.

En el informe se detalla que las ventas de Smartphones a nivel global alcanzaron los 334.9 millones en el primer trimestre de 2016, el crecimiento respecto al mismo periodo del 2015 fue del 0.2%. Samsung y Apple lideraron las ventas copando casi el 40% de la cuota de mercado.

Además, según el informe las personas mayores de sesenta y cinco años empezaron a adoptar el uso de Tablets, siendo así un segmento de población que crece en un 219%, por lo tanto la brecha digital se cierra cada vez más para los que hasta hace poco estaban rezagados en el uso de las tecnologías.

Motivo por el cual se va a optar por desarrollar para un dispositivo móvil, sobre todo para Smartphones, siendo una importante opción a tener en cuenta para llegar al mayor número de usuarios del público en general.

Según el informe Android fue la única plataforma con crecimiento interanual en junio de 2016, pasando de una cuota de mercado del 88.1% en junio de 2015 al 91,2% en el mismo mes de 2016, iOS pasó en el mismo periodo del 8.8% al 8.2% y Windows Phone pasó del 4% al 0.4%.

Viendo estos datos está claro que la plataforma preferida es el dispositivo móvil con el sistema operativo de Android instalado, que dispone en Google Play de una gran cantidad de aplicaciones gratuitas. El auge de aplicaciones a nivel mundial en 2015 creció un 58% con respecto al año anterior, demostrando que existen muchos desarrolladores para Android.

También en 2016 los wearables representan un sector que está en pleno crecimiento y se espera que siga así durante los próximos años. Android permite programar aplicaciones para todo tipo de dispositivos, también existe la posibilidad de programar para dispositivos wearables, por lo tanto será interesante tener en cuenta este punto de cara al futuro del proyecto.

Además, Android tiene muchas aplicaciones que solo funcionan si se dispone de una conexión permanente a internet, actualmente muchas compañías de telecomunicaciones ofrecen paquetes de acceso a internet fijo en casa permitiendo la conexión por Wi-fi de los dispositivos móviles, pero también ofrecen muchas facilidades para poder acceder a internet desde el dispositivo móvil fuera de casa y desde cualquier sitio (siempre y cuando estén dentro de la red de telecomunicaciones de la compañía contratada).

Un punto a destacar del informe de la sociedad de la información, es que el 92% de internautas accede a internet desde su propio terminal móvil, 20 puntos porcentuales más de quienes acceden desde un ordenador personal, además las conexiones de banda ancha móvil triplica a las conexiones de banda ancha fija.

Durante el 2016 el 91.7% de los internautas se han conectado a internet con el Smartphone, mientras que el 73.1% lo hizo con el ordenador, el 100% de los jóvenes entre catorce y diecinueve años se conectan actualmente utilizando el Smartphone, tendencia que continuará en los próximos años.

Por todo ello es conveniente tener en cuenta que existe un gran número de usuarios con conexión a internet, por lo tanto la aplicación se podrá desarrollar para ser administrada completamente desde internet, de esta forma la aplicación ocupará menos memoria en el dispositivo móvil, ya que analizará la información disponible en internet para poder mostrarlo adecuadamente sin la necesidad de almacenarlo en la memoria local.

Se debe destacar también que Android está optimizado para trabajar a baja potencia y poca memoria, aunque no todas las aplicaciones funcionan así, no optimizan muchos aspectos de la funcionalidad, pensando más en cómo mostrar la información, esto implica

que el dispositivo móvil utilice muchos más recursos sin ser necesarios, algo que se tendrá mucho en cuenta en este proyecto.

3.2.- Entorno de desarrollo integrado

Entorno de desarrollo integrado o IDE (Integrated Development Environment) es una plataforma de desarrollo que facilita las herramientas básicas que necesitan los desarrolladores para escribir y desarrollar software, normalmente una IDE contiene un editor de código, un compilador o interpretador y un debugger (depurador).

NetBeans, Eclipse y Android Studio son los IDEs más populares para desarrollar aplicaciones para Android, a continuación se analizará brevemente cada uno de ellos para ver al final qué herramienta de desarrollo es la más conveniente para programar aplicaciones en Android.

3.2.1.- IDE NetBeans

NetBeans es un IDE basado en Java, diseñado para limitar los errores de código y facilitar la corrección de errores con herramientas como FindBugs. En un principio fue diseñado específicamente para trabajar con Java, pero ahora soporta C/C++, Php, Groovy y Html5 junto a Java, JavaScript y JavaFX. Oracle es el principal patrocinador del proyecto NetBeans desde que fue adquirido por SUN en 2010, fuente de información [3] y [4].

Trabaja bajo sistemas operativos que soportan JVM (Java Virtual Machine) como Linux, Windows y OS X. En 2000 pasó a ser open source, actualmente está disponible bajo licencia Apache 2.0. La licencia Apache 2.0 es una licencia de software libre que requiere que se preserve un aviso con el copyright y el descargo de responsabilidades, escrita por Apache Software Foundation permitiendo utilizar el software libremente para cualquier propósito, distribuirlo y modificarlo bajo los términos y licencias sin conceder royalties.

NetBeans utiliza el plugin NBAndroid que sirve como puente de comunicación entre la IDE y la SDK de Android. NetBeans, junto al plugin NBAndroid cubre todas las necesidades que se pueda tener para trabajar cómodamente en Android, suelen liberar nuevas versiones de NetBeans frecuentemente, la última fue el 27 de Abril del 2017.

3.2.2.- IDE Eclipse

Eclipse es una plataforma que permite desarrollar software basado en Java, utiliza distintos plugins para personalizar el área de trabajo según el programa que se desee desarrollar, fuente de información [3].

Aparece en 2001 cuando IMB donó tres millones de líneas de código de sus herramientas Java, el objetivo principal era crear y fomentar una comunidad IDE open source. Está

escrito en Java aunque permite desarrollar y hacer test del código escrito en otros lenguajes de programación.

En el mundo empresarial, la mayor ventaja de tener una herramienta de desarrollo open source es que permite desarrollar un producto mezclando distintas herramientas de código libre para trabajar como un solo suite en el producto.

ADT (Android Development Tool) es un plugin para el IDE de Eclipse que permite desarrollar rápidamente projectos Android, crear la UI (interfaz de usuario), añadir paquetes basados en la "Android FrameWork API", depurar la aplicación usando las herramientas de Android SDK y exportar APKs firmadas o no firmadas.

Para poder comenzar a desarrollar con Eclipse es necesario descargar e instalar Android SDK desde la web oficial de Android, también es necesario descargar e instalar Eclipse IDE en el cual se podrá instalar el plugin "ADT Repository", la última versión de Eclipse IDE es la 4.6.3 de Marzo del 2017 que a diferencia de NetBeans utiliza la licencia "Eclipse Public Licence", reemplazando a "Common Public License" quitando algunos términos relacionados a litigios de patentes.

3.2.3.- IDE Android Studio

Android Studio es la IDE de Google que provee a los desarrolladores todas las herramientas necesarias para crear aplicaciones que funcionan sobre dispositivos móviles (Smartphones y Tablets), Pcs y cualquier otro tipo de tecnología emergente que funcione con Android (Android TV, Android Wear, Android Auto, Glass, etc.), fuente de información [3].

Su uso y descarga es gratuita, está disponible para Windows, Mac y Linux bajo licencia Freeware. La licencia Freeware posibilita utilizar el software propietario libremente sin realizar pagos, además el mismo propietario (Freeware publisher) es el que definirá y cambiará libremente las reglas Freeware de su producto. Las modificaciones, redistribuciones o ingeniería inversa (reverse-engineering) están prohibidas sin permiso del autor.

Android Studio tiene la intención de ser utilizado por equipos de desarrolladores grandes y pequeños. La IDE Android Studio se puede unir con GIT (sistema de control de versiones) o cualquier servicio de control similar para poder trabajar con grandes equipos de desarrolladores para repartir soluciones rápidas a los clientes.

El flujo de trabajo para Android Studio está construido en base al concepto de "Integración continua", este concepto permite realizar test a cada uno de los códigos de cada uno de los equipos de desarrollo cada vez que un equipo registre su trabajo. Los problemas pueden ser capturados y reportados al equipo inmediatamente, además el concepto de registro continuo de código provee feedback inmediato a los desarrolladores

con el objetivo de liberar rápidamente versiones de sus productos finales sobre Google Play App Store.

Bajo esos conceptos, hay un riguroso soporte para las herramientas LINT, Pro-Guard y App Signing. Es necesario tener una licencia de desarrollador que cuesta 25$ para publicar aplicaciones en Google Play App Store.

Las herramientas de performance de Android Studio permite ver cómo de bien está funcionando la APK, las herramientas de memoria permiten visualizar dónde y cuando la aplicación usará mucha memoria RAM y cuándo el colector de basura comenzará a funcionar, las herramientas de análisis de batería muestra cuánta batería está consumiendo la aplicación desarrollada en el dispositivo.

Android Studio soporta Google App Engine para una veloz integración de las nuevas APIs y nuevas características. Se podrá encontrar soporte para muchas APIs directamente en Android Studio como Google Play, Android Pay y Health.

Hay soporte para todas las plataformas de Android empezando con Android 1.6 y superiores, la versión actual para descargar es la 2.3.1 del mes de Abril 2017. Hay muchas variantes de Android que son significativamente diferentes a la versión Google Android (la más popular es Fire OS de Amazon), Android Studio puede ser utilizado también para crear APKs para Fire OS de Amazon siguiendo unas guías.

3.2.4.- Resumen de la IDE del proyecto

Se puede comenzar a trabajar con cualquiera de las tres IDEs analizadas. Todas estas herramientas permiten realizar el trabajo de manera parecida. Actualmente la interfaz de trabajo es casi la misma en todos, al igual que la organización del proyecto. NetBeans y Eclipse son las más antiguas y soportan muchos otros lenguajes de programación y Android Studio se ha creado para trabajar específicamente con Android.

Android Studio es la IDE oficial de Google, tiene menos bugs respecto al autocompletado y a la reutilización de código es la que más opciones inteligentes presenta, es la más fácil de instalar, la más fácil de utilizar y la que mejor compatibilidad tiene con distintos sistemas operativos, aunque Android Studio no siempre tiene el mejor rendimiento, cada nueva versión liberada tiende a mejorar mucho este aspecto.

Por todo ello Android Studio será el IDE que permitirá desarrollar la aplicación Android de este proyecto, es la más recomendable para todos los desarrolladores que quieren iniciarse en el desarrollo de aplicaciones Android, sin embargo para los desarrolladores más experimentados elegir una u otra herramienta dependerá con cual estén más cómodos o acostumbrados.

La versión que se ha instalado para trabajar en el desarrollo de la aplicación es Android Studio 2.1.2 del 26 de Mayo de 2016 con JRE 1.8.0_91-b15 amd64 y Java Virtual Machine HotSpot (TM) 64-bit Server VM de Oracle Corporation.

3.3.- Desarrollando en Android

Android es un sistema operativo similar a Symbian, iOS, Windows Mobile y otros. Fue inicialmente desarrollado por Android Inc. que luego fue adquirida por Google. Es open source y goza de gran popularidad, Google libera la mayoría del código de Android bajo licencia Apache, permitiendo poder crear extensiones de Android sin necesidad de registrarlo como open source, fuente de información [3].

Android utiliza un kernel modificado de Linux y permite desarrollar aplicaciones utilizando la tecnología de Java. Las aplicaciones son escritas en el lenguaje de Java, no hay una máquina virtual y el código Java no es ejecutado. Las clases de Java son recompilados para funcionar sobre una máquina virtual Dalvik. Dalvik es una máquina virtual modificada y optimizada para funcionar en dispositivos que tienen una batería como fuente de alimentación y CPUs de gama baja.

3.3.1.- La clase Activity

Una Activity representa la parte visual de cara al usuario, cada pantalla que el usuario ve en una aplicación puede representar distintas Activities, fuente de información [5].

Por ejemplo, una Activity puede tener una lista de algún menú con ítems que el usuario puede elegir o mostrar: fotografías, botones, etc. Una aplicación de mensajería de texto puede tener una Activity que muestre una lista de contactos para poder elegir y otra Activity que revise los mensajes antiguos o una Activity que tenga opciones que modifique el comportamiento de la aplicación.

Todas las Activities interactúan con el usuario, normalmente se representan como una ventana, sin embargo se puede representar de otras maneras: como ventana flotante con "windowsIsFloating" o embebida dentro de otra Activity usando "ActivityGroup".

Cuando una nueva Activity es iniciada, se pone a la cabeza de la lista y comienza a ser ejecutada. La figura 12 representa un diagrama del ciclo de vida de las Activities.

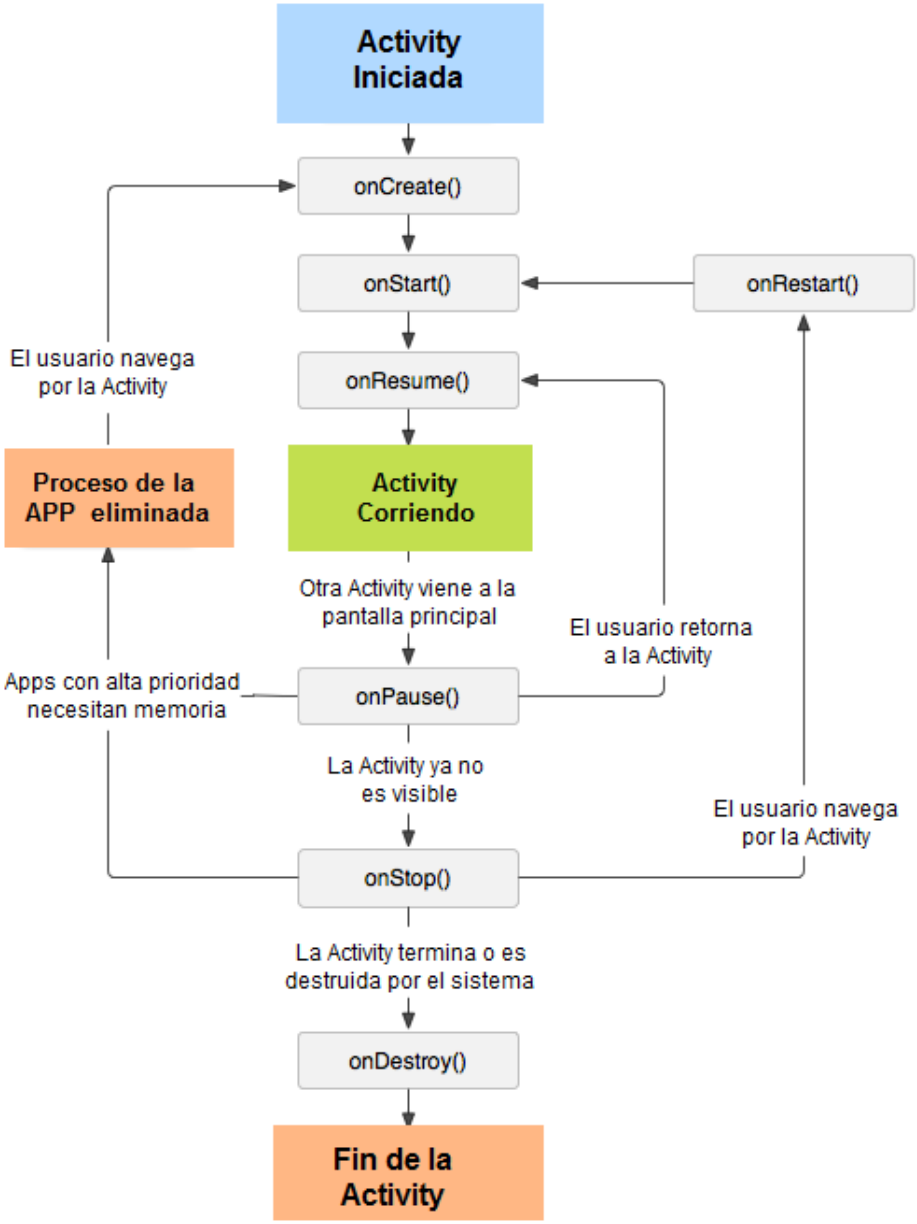


Figura 12: Ciclo de vida de la Activity

Existen tres estados de monitoreo en la Activity:

- El completo tiempo de vida, ocurre entre la primera llamada a "onCreate(Bundle)" hasta la llamada a "onDestroy()".
- Tiempo de vida visible, ocurre entre la llamada de "onStart()" hasta la correspondiente llamada de "onStop()", entre esos dos métodos se puede mantener los recursos necesarios para mostrar la Activity al usuario, que se pierden una vez que se haya detenido la Activity.
- Tiempo de vida en primer plano, ocurre cuando se llama a "onResume()" hasta la llamada a "onPause()", entre esos dos métodos la Activity está interactuando con el usuario y frecuentemente cambia a estado pausado y retomando su actividad.

El completo ciclo de vida de la Activity está definida por métodos que funcionan cuando la Activity cambia de estado. Todas las Activities implementan onCreate(Bundle) para

poder ser creados por primera vez, muchos implementarán onPause() para afianzar cambios de los datos o estar listo para detener la interacción con el usuario.

La lista de métodos de una Activity son: “protected void onStart();”, “protected void onRestart();”, “protected void onResume();”, “protected void onPause();”, “protected void onStop();” y “protected void onDestroy();”.

3.3.2.- El Content Provider

Un Content Provider es una interfaz estándar, crea un conjunto de datos específicos que son encapsulados y son provistos de mecanismos de seguridad para poder luego compartir esos datos con otras aplicaciones. Los datos pueden estar almacenados en la memoria interna, en una base de datos SQLite o en otro tipo de almacenamiento lógico, fuente de información [5].

El Content Provider es parte de una aplicación Android, implementarlo tienen muchas ventajas, el más importante es que se puede permitir a otra aplicación que llegue a modificar sus datos a través de una conexión segura, controlando los permisos de acceso de otras aplicaciones a esos datos.

3.3.3.- La clase Service

Un servicio no tiene un componente visual, pero funciona en background por un indefinido periodo de tiempo aunque el usuario cambie a otra aplicación. Un servicio puede manejar transacciones de red, reproducir música, realizar I/O de archivos o interactuar con el proveedor de contenido; todo en segundo plano, fuente de información [5].

Se puede adoptar de dos maneras:

- Como un servicio iniciado, ocurre cuando un componente de la aplicación (como una Activity) lo inicia llamando a “startService()”. Una vez iniciado el servicio se ejecuta en segundo plano de manera indefinida, por ejemplo si se descarga un archivo la operación terminará cuando el servicio se detenga por sí mismo.
- Como un servicio de enlace, ocurre cuando la aplicación se vincula al que le ha llamado con “bindService()”, este servicio ofrece una interfaz cliente-servidor para interactuar con los componentes del servicio para que envíen solicitudes, obtengan resultados, etc.

Para crear un servicio se debe crear una clase que derive de Service cuyos métodos más importantes son: “onStartCommand()”, “onBind()”, “onCreate()” y “onDestroy()”.

Se debe declarar el servicio en Manifest, el código en cursiva representa el contenido del archivo manifest y cómo se debería añadir el nombre de la clase service:

```
<manifest ... >
  ...
  <application ... >
      <service android:name=".NombreServicioEjemplo" />
      ...
  </application>
</manifest>
```

3.3.4.- La clase abstracta Broadcast receiver

Las aplicaciones Android pueden enviar o recibir mensajes de tipo Broadcast desde el sistema Android y otras aplicaciones. Esos mensajes Broadcast se envían cuando ocurre un evento de interés, fuente de información [5]. Por lo tanto la mayoría de mensajes broadcasts se originan en el sistema como cambio de zonas horarias, anuncios de baja batería, preferencia en idiomas, etc.

3.3.5.- La clase abstracta Intent

Un Intent puede ser utilizado junto a "startActivity" para cargar una "Activity" o "BroadcastIntent". Un Intent facilita la interacción y comunicación entre Activities, también se puede usar para poder comunicarse con servicios en segundo plano, fuente de información [5].

3.3.6.- Intercambio de datos

Los dispositivos móviles siempre están añadiendo, actualizando y mejorando características y funcionalidades con cada nueva versión, sobre todo características relacionadas a distintos modos de interacción. A la hora de programar es muy importante tener en cuenta la administración de datos que llegan constantemente a las aplicaciones.

Anteriormente XML era la única manera de compartir datos libremente, manejando datos como números, textos, ficheros, imágenes, audio, videos, etc. Las redes sociales son un claro ejemplo de cómo los datos deben ser tratados para poder acceder a su contenido desde dispositivos móviles, Twitter utiliza XML y Json para intercambiar datos.

3.3.6.1.- XML

Un documento puede tener una estructura fácil de reconocer. Cualquiera que esté trabajando programando webs, servidores virtuales, aplicaciones móviles y todo tipo de ambiente, se ha encontrado con XML (eXtensible Markup Language), fuente de información [3] y [6].

Por ejemplo, los paquetes de un ERP (Enterprise Resource Planning) usan XML para importar y exportar tareas. Varios sitios web utilizan XML en RSS (Really Simple Syndication) incluso Open Office o Microsoft Office usan XML.

XML fue diseñado para almacenar y transportar datos, está en varios sitios, pero a menudo ofrece pocas opciones a la hora de programar en Android ya que no se puede almacenar grandes cantidades de datos de XML, sin embargo se puede intercambiar datos en cualquier formato.

3.3.6.2.- Json

Json (JavaScript Object Notation) es una sintaxis que se utiliza para almacenar e intercambiar datos. Json presenta un formato ideal para el intercambio de datos, está limitado a valores textuales y numéricos, fuente de información [6].

Actualmente muchas APIs en internet ofrecen datos con formato Json, es muy utilizado en Ajax. Cuando una aplicación Ajax intercambia información con un servidor web, a menudo las peticiones van sin formato, esa situación se considera como una mala práctica sobre todo si la información representa a más de un elemento.

Si XML es conocido por su gramática, Json es conocido por ser difícil de ser leído, cada grupo de datos están separados por comas y van dentro de llaves "{}", a continuación se puede ver un ejemplo de datos de Json:

[{"firstname":"Troy","lastname":"Mott","age":"don't ask!"},
{"firstname":"Apple seed",
"lastname":"Mott's","age":"99"}]

3.3.6.3.- Comparativa XML vs Json

Json y XML utilizan el estándar Unicode, ambos son legibles para el ser humano. Ambos utilizan un sistema de conversión de datos (parsing) diferente para poder ser utilizado en una aplicación Android, fuente de información [7].

XML utiliza el método "examineXMLFile()" para recibir los datos XML y crear "SAXParser" asociado al handler "twitterFeedHandler" para realizar el trabajo de conversión, que normalmente almacena los datos en forma de string para que luego esos datos puedan ser utilizados por la aplicación.

Json invoca a una función para convertir los datos, no requiere ninguna clase handler adicional ya que todo el proceso de transformación y administración de los datos se realizan con librerías suplementarias de Android. Obtiene los datos de la fuente, es leída en memoria y convertida a "java.lang.String" para luego ser transformada a "JSONArray" que representará los datos como un string.

Json está limitado a almacenar datos como texto y números, XML almacena información en cualquier tipo de formato. Inicialmente en este proyecto solo se busca el intercambio de datos en forma de texto o número, Json almacena los datos en forma de vectores y

registros mientras que XML almacena los datos en árboles, por lo tanto este proyecto se beneficiará de Json más que de XML.

3.3.7.- La clase Application

La clase Application es una clase base que se mantiene como global, contiene componentes como Activities y Servicios como se ve en la figura 13, fuente de información [8] y [9].

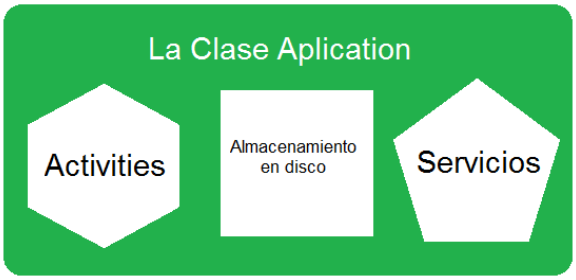


Figura 13: Componentes de la clase Application

Se implementa creando una clase que hereda de Application y se debe especificar el nombre de la clase sobre el atributo "android:name" de <application> en AndroidManifest.xml, como se ve a continuación:

```
<application
  android:name=".MiClaseApplication"
  android:icon="@drawable/icon"
  android:label="@string/app_name"
  ...>
```

Cuando la aplicación/paquete es creada, la clase Application es inicializada antes que cualquier otra clase, es utilizada principalmente para inicializar estados globales antes de que se muestre la primera Activity.

No se recomienda tener variables globales, ya que tiende siempre a generar errores. Es aconsejable utilizar el paso de variables de manera explícita entre Activities a través de "Intent" o almacenar los datos de alguna manera en el móvil. Solo se recomienda utilizar la clase Activity si es realmente necesario, por ejemplo en este proyecto se podrá crear una clase Application que haga de controladora global, según el Modelo Vista Controlador.

3.3.8.- Hilos de Ejecución

Cuando una aplicación es cargada, el sistema crea un hilo de ejecución, ese hilo es muy importante ya que está encargado de administrar eventos de la interfaz del usuario, además ese hilo interactúa con los componentes de Android UI (elementos de la pantalla del usuario), fuente de información [5].

El sistema no crea hilos de ejecución separados para cada instancia de cada componente. Todos los componentes funcionan sobre el mismo hilo de ejecución principal, por ejemplo, cuando el usuario toca un botón de la pantalla, el hilo de ejecución principal dispara el evento "onTouch" del botón, asignando su estado a presionado.

Android soporta el uso de hilos de ejecución utilizando la clase Thread, estos hilos necesitarán sincronizarse con el hilo principal, es importante dar al usuario un feedback del proceso de ejecución, por ejemplo, mostrando una barra para ver el estado del proceso. Cuando el hilo principal está bloqueado más de cinco segundos, saldrá un mensaje dejando al usuario elegir si se finaliza o no.

3.3.8.1.- El hilo de ejecución principal

Este hilo de ejecución modifica la interfaz del usuario y administra los eventos desde un solo hilo de ejecución, fuente de información [5]. Android colecta todos los eventos en el hilo principal sobre una cola, luego procesa esa cola con una clase Looper.

3.3.8.2.- Mensajes Handler

Un objeto Handler se registra con el hilo creado, provee un canal para enviar datos al hilo, se suele asociar a un hilo específico gestionando Mensajes y Runnables, fuente de información [5].

Cada hilo tiene una cola de mensajes (que guarda mensajes y Runnables) y un Looper para enviar el Mensaje y el Runnable a su hilo asociado. Los mensajes pueden contener el código del mensaje, un objeto o valores enteros.

3.8.3.3.- La clase AsyncTask

La clase AsyncTask permite realizar instrucciones asíncronas en background y se sincroniza con el hilo principal, realiza las operaciones de bloqueo y publica los resultados de su ejecución en el hilo principal. Además reporta el progreso de las tareas que se están ejecutando, fuente de información [5] y [10].

Para utilizarlo, la clase debe extender de AsyncTask y debe implementarse en el método callback "doInBackground()", que lo hace trabajar en una pila de hilos de ejecución en segundo plano, para actualizar el hilo principal se debería implementar "onPostExecute()", que repartirá el resultado desde “doInBackground()”, luego trabaja sobre el hilo de ejecución principal, permitiendo actualizar de manera segura la UI (interfaz de usuario).

AsyncTask usa las siguientes funciones: “onPreExecute()”, “doInBackground(Params…)”, “onProgressUpdate(Progress...)” y “onPostExecute(Result)”.

Se puede cancelar la ejecución de la tarea invocando "cancel(boolean)", se puede también verificar si está cancelado o no llamando a "isCancelled()" desde "doInBackground(Object[])". Cuando se invoca la cancelación, "onCancelled(object)" se invoca después de "doInBackground(Object[])".

Cuando el primer hilo se introduce, AsyncTask será ejecutado en serie sobre hilos de ejecución en segundo plano, en la versión de Android "Donut", los hilos se podían ejecutar paralelamente, pero desde la versión de Android "HoneyComb" se ejecutan en un solo hilo de ejecución para evitar que errores causados por la ejecución paralela. Aunque Android también permite la ejecución paralela.

3.4.- Comunicación con el servidor

Si se va a gestionar la difusión de datos en tiempo real en la aplicación Android desde internet, es necesario tener un servidor que contenga los datos y que la aplicación sea capaz de solicitar los datos más importantes para poder mostrar la información, para ello se analizará distintos métodos disponibles para poder realizar esa comunicación lo más eficiente posible.

3.4.1.- Comunicación mediante Sockets

Los sockets representan un punto de comunicación entre dos máquinas, una aplicación se puede conectar a un servidor vía internet (TCP/IP), el trabajo del socket es realizada por una instancia de la clase "SocketImpl". La aplicación se puede configurar y crear sockets apropiados para el firewall local, fuente de información [11] y [12].

Se necesita dos tipos de sockets para manejar la conexión, un socket cliente y un socket servidor. Tanto el cliente como el servidor pueden ser dispositivos móviles, uno de ellos tiene que ser el socket servidor que está siempre escuchando peticiones, debe tener una dirección IP y un puerto, también el servidor puede estar corriendo sobre tomcat con una IP específica por el puerto 8080, esperando y respondiendo peticiones del cliente.

El socket cliente es obviamente Android (aunque puede ser cualquier otra aplicación Java), utilizará "AsyncTask" para hacer conexiones en background, una interfaz "Callbacks" para administrar los mensajes que lleguen, "Handler" para actualizar la "GUI" y la clase "TCPClient" que representará al cliente.

Se utiliza AsyncTask para evitar errores de acceso a la red ("StrictMode"). "StricMode" simplemente prohíbe realizar acciones que afecten el hilo de procesamiento principal "UI Thread". Es necesario además que la aplicación tenga permisos de acceso a internet, en Manifest se añade: <uses-permission android:name="android.permission.INTERNET" />

3.4.2.- La clase HttpUrlConnection

HttpURLConnection es una clase pública abstracta del paquete “java.net.*” que deriva de "URLConnection", soporta una conexión HTTP. Su funcionamiento está condicionado a versiones superiores a GigerBread, para versiones anteriores se debe utilizar el cliente "HttpClient" de Apache, fuente de información [13].

Según la web [5], su uso debe seguir los siguientes patrones:

- Obtener una nueva HttpURLConnection llamando a "URL.openConnection()" y el resultado convertirlo con cast a "HttpURLConnection".
- Preparar la petición la cabecera de peticiones puede incluir metadata como credenciales, tipo de contenido preferido y sesiones con cookies.
- De manera opcional se puede subir el cuerpo de la petición, instanciada y configurada con "setDoOutput(true)", si se incluye el cuerpo de la petición se transmite datos escritos para ser retornados por "getOutputStream()".
- Para leer la respuesta las cabeceras suelen incluir metadata como el cuerpo de la respuesta que contiene el tipo y el tamaño, la fecha de comunicación y la sesión con cookies. El cuerpo de la respuesta puede ser leído por "getInputStream()", si la respuesta no tiene nada se devuelve un stream vacío.
- Desconectar una vez que el cuerpo de la respuesta ha sido leído, HttpURLConnection debería cerrarse llamando al método "disconnect()", desconectando y liberando los recursos de la conexión.

Se puede hacer comunicaciones seguras con HTTPS, llamando a "openConnection()" sobre una URL con "https", el esquema retornará un HttpsURLConnection, permitiendo sobreescribir los valores por defecto "HostnameVerifier" y "SSLSocketFactory".

Se puede subir datos a un servidor web, configurando la conexión para usar "setDoOutput(true)", para un mejor rendimiento, se debería llamar también a "setFixedlengthStreamingMode(int)", cuando el tamaño del cuerpo es conocido o "setChunkedStreamingMode(int)" si el tamaño del cuerpo no se conoce. De otra manera HttpURLConnection forzará el buffer para completar la petición en memoria antes de ser transmitido, desperdiciando recursos e incrementando la latencia.

Para reducir la latencia, esta clase puede reutilizar el mismo socket para múltiples peticiones/respuestas. Por defecto las peticiones HttpURLConnection requieren que los servidores utilicen el compresor gzip.

Para conectarse a la red es necesario tener permisos de conexión a Internet y permisos de acceso al estado de la red, añadiendo a Manifest lo siguiente:

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

Actualmente, todo “Apache http” en Android se ha quedado obsoleto. Cuando Android 1.0 fue liberado, HttpCliente 4.0 fue pausado prematuramente ya que muchas interfaces y estructuras internas no funcionaban, se esperaba que Google incorporara mejoras y opciones que permitan compatibilidad con Apache HttpClient, pero nunca ocurrió. Se había liberado Apache HttpClient 4.3 cuya intención era solucionar los problemas de compatibilidad con Android.

3.4.3.- La librería Volley

La librería Volley es una librería de red desarrollada por Google e introducida en 2013. Hasta ese momento existía una carencia en la SDK de Android, no había una clase para trabajar con operaciones en red. Antes de liberar Volley, las clases de Java "java.net.HttpURLConnection" y Apache "org.apache.http.client" en Android eran las herramientas más utilizadas por los desarrolladores para realizar operaciones en red. Además, si se quisiese caché de imágenes o priorizar peticiones, se debía desarrollar desde "scratch", fuente de información [14], [15] y [16].

Volley evita el uso de AsyncTask para operaciones en red, desde HoneyComb (API 11) las operaciones en red se debían realizar sobre hilos de ejecución separado del hilo principal. El principal problema de AsyncTask es la serialización, no se puede decidir cuál petición va primero o cuál debe esperar, todo funciona como FIFO (First In First Out).

Los problemas se pueden incrementar, por ejemplo, cuando se tiene que cargar una lista de ítems que tienen adjuntos thumbnails (miniaturas), cuando el scroll baja y espera nuevos resultados no se puede decir que Activity va a cargar primero el Json de la siguiente página o solo las imágenes previas.

Volley soluciona ese problema, es una poderosa API pudiendo incluso cancelar peticiones, por ejemplo, si la Activity ha sido destruida mientras se hacía una petición, no es necesario realizar comprobaciones sobre "onPostExecute" para saber en qué estado ha quedado la petición. De esta manera se evitan errores "NullPointerException".

Volley es mucho más rápido, realiza caché automáticamente de las peticiones, permitiendo además que el código de desarrollo sea más corto. Realiza pequeñas operaciones de metadata, como el uso de objetos Json, porciones de listas, detalles de los ítems elegidos, etc.

3.4.3.1.- ¿Cómo funciona Volley?

Por defecto realiza llamadas síncronas, trabajando en tres diferentes niveles de operación sobre el mismo hilo de ejecución.

3.4.3.1.1.- Hilo principal

Sobre el hilo principal, sólamente se realiza la petición y se gestionan las respuestas. Se ignora todo lo que ocurre en el método "doInBackground", Volley administra de manera automática todas las transacciones HTTP y la caché de los errores de red sin necesidad de que el desarrollador se preocupe de realizarlo manualmente con anterioridad.

3.4.3.1.2.- Cache y Networks threads

Ocurre muchas cosas cuando una petición se añade a la cola, la figura 14 representa un diagrama del funcionamiento de Volley.

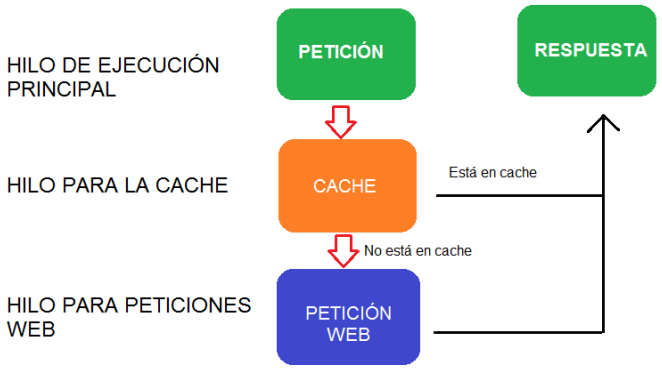


Figura 14: Hilos de trabajo de la librería Volley

Primero Volley verifica que la petición pueda ser servida desde la caché, si se puede, la respuesta es leída, convertida y repartida. De otra manera la petición pasa a un hilo de ejecución para realizar operaciones en red.

Los hilos de peticiones web realizan operaciones en red, trabajan constantemente junto a round-robin con una serie de hilos de ejecución. El primer hilo de ejecución disponible se quita de la cola de peticiones y realiza la petición HTTP, la respuesta debe ser transformada y se escribe sobre la cache.

Para terminar las respuestas convertidas se envían al hilo principal de ejecución donde unos listeners están esperando el resultado de la petición.

3.4.3.2.- Instalación

Primero se debe instalar la librería, una vez instalado se debe añadir a Manifest los permisos para poder acceder a internet "*android.permission.INTERNET*". La instalación de la librería Volley se puede hacer de dos maneras, una automática y otra manual.

Para instalar automáticamente, sencillamente se puede añadir la librería al módulo de dependencias gradle.build como se ve a continuación:

```
dependencies {
    ...
    compile 'com.android.volley:volley:1.0.0'
}
```

Es la más sencilla y recomendada si se quiere tener disponible la librería Volley, cuando la librería recibe una actualización, bastará con cambiar el nombre de la librería en compile.

Para la instalación manual, se debe descargar e instalar git y apache ant, luego se debe clonar la librería Volley para luego hacer make desde la consola de comando:

- *git clone https://github.com/google/volley*
- *cd volley*
- *android update project -p .*
- *ant jar*

El archivo generado se encuentra en volley.jar en la carpeta bin. se copia la librería a la carpeta de Android Studio y se añade una entrada en build.gradle como se ve a continuación:

```
dependencies {
    compile files('libs/volley.jar')
}
```

Estos pasos se deben repetir en caso de que se quiera añadir una versión nueva o actualizada de la librería Volley.

3.4.3.3.- Beneficios y usos de Volley

Beneficios de utilizar Volley:

- Programación automática de peticiones de red.
- Múltiple concurrencia de conexiones de red.
- Sigue el estándar "HTTP cache coherence".
- Soporte para priorización de peticiones.
- Cancelación de peticiones simples o por bloques.
- Fácil personalización.
- Fuertemente ordenado, haciendo fácil manejar datos en la UI .
- Existen herramientas para hacer debug.

Volley trabaja principalmente con dos clases, la clase “RequestQueue” y la clase “Request”. Primero se debe crear la clase RequestQueue, se construye la petición y se añade esa petición a RequestQueue, la cual administra los hilos de ejecución y entrega el resultado de una petición al hilo de ejecución principal.

El constructor de Request recibe como parámetros el tipo (Get, Post, etc.), la URL y los eventos listeners. Luego dependiendo del tipo de petición, puede solicitar algunas más variables, la figura 15 representa un diagrama de todo el proceso de comunicación de la aplicación Android con un servidor en Internet.

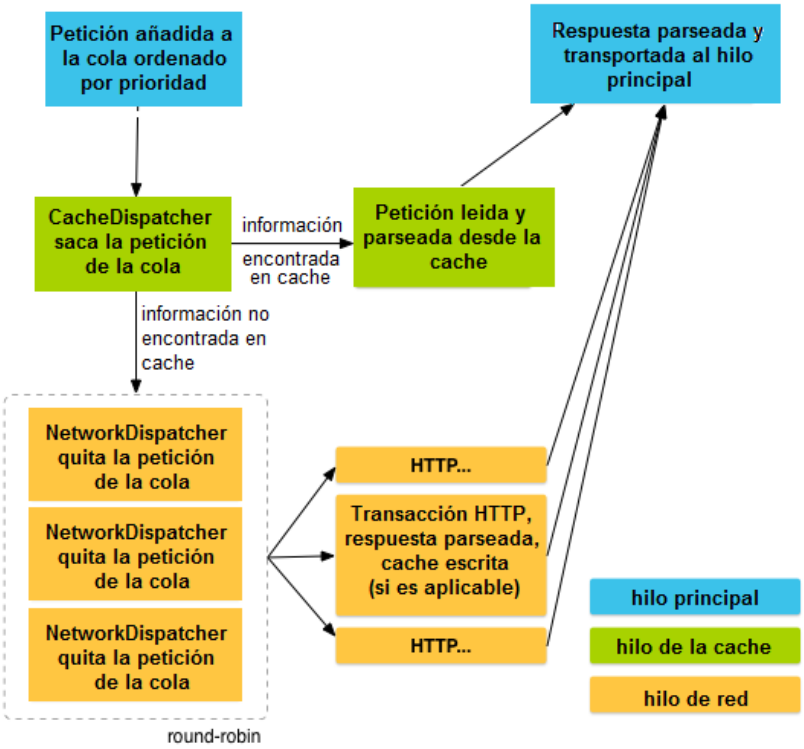


Figura 15: Funcionamiento de la librería Volley

Cuando se añade la petición a la cola, Volley hace trabajar un hilo de ejecución en la caché y otro hilo de ejecución para operaciones de red, primero se busca que la respuesta de la petición esté en caché, para poder responder desde la caché a menos que no esté ahí. Entonces la petición se enviará a la cola de red, el primer puesto disponible del hilo de ejecución de la red realizará la transacción HTTP y pasará la respuesta a la caché y lo traspasará al hilo de ejecución principal.

Para cancelar la petición se debe incluir el método "cancel()" sobre el objeto Request, también desde la Activity se puede llamar al método "onStop()". Una vez cancelado Volley se responsabiliza de que la respuesta nunca sea llamada.

Si se hace múltiples peticiones al mismo tiempo, se puede añadir priorizaciones como: baja, normal, intermedia o alta. También se puede consultar la prioridad, sobreescribiendo el método "getPriority" de la clase Request.

Volley no es adecuado para realizar operaciones de streaming con grandes cantidades de datos, ya que Volley mantiene todas las respuestas en memoria durante las operaciones de conversión. Sin embargo en este proyecto no se utilizarán grandes cantidades de datos y las operaciones de streaming serán gestionadas en el servidor web, dando al cliente solo los datos más actuales que no tenga (no se le dará toda la información).

Todo esto hace que la librería Volley sea la herramienta ideal en este proyecto, para administrar el intercambio de datos de manera eficiente entre el cliente y el servidor.

3.5.- El modelo de capas

El propósito de muchos sistemas y aplicaciones informáticas es mostrar información desde su fuente (donde está almacenada) hasta llegar al usuario, como el flujo de la información va desde su fuente al usuario, se puede pensar que solo intervienen esos dos aspectos, fuente de información [17].

Eso conlleva:

- La lógica de la interfaz de usuario cambia frecuentemente, más que la lógica de negocio (sobre todo en aplicaciones web), modificando constantemente el objeto que contiene la lógica de negocios cada vez que haya cambios en la interfaz de usuario.
- En algunos casos la aplicación muestra la misma información en diferentes formas, por ejemplo, analizando de manera diferente los mismos datos en diferentes vistas, por lo tanto si el usuario cambia datos en una vista el sistema debería ser capaz de actualizar todos los datos de todas las vistas.
- Es muy raro encontrar personas que son muy buenas analizando la complejidad de la lógica del negocio y al mismo tiempo ser experto en el diseño visual, es conveniente separar esos dos campos.
- La interfaz de usuario tiende a ser más dependiente del dispositivo que de la lógica de negocio, si se quiere migrar de una aplicación web a una aplicación adaptada a distintos dispositivos móviles, se debería reemplazar el código de la interfaz, el código de la lógica de negocio no debe sufrir cambios.

La solución está en el modelo de capas Modelo Vista Controlador (MVC) es el nombre de una metodología o patrón de diseño para una relación satisfactoria y eficiente de la interfaz de usuario con el modelo de datos, es ampliamente usado para el desarrollo de programas orientado a objetos con lenguajes como Java, Smalltalk, C o C++.

El modelo de capas es conocido por muchos desarrolladores como un patrón útil para la reutilización de objetos, permitiendo una significante reducción de tiempo a la hora de desarrollar aplicaciones con la interfaz de usuario.

Para el desarrollo de software, el modelo de capas propone principalmente tres componentes:

- Un Modelo que representa la estructura lógica de datos en la aplicación software y su asociación con las clases de más alto nivel. El Modelo no contiene ninguna información sobre la interfaz del usuario, sin embargo también representa a los datos que se transfieren entre la Vista y el Controlador.
- Una Vista es una colección de clases que representa a todos los elementos de la interfaz del usuario, incluye todas las cosas que el usuario puede ver y responder a eventos generados por la interacción entre el usuario y la interfaz a través de botones, cajas, etc.

- Un Controlador representa las clases que conectan el Modelo y la Vista, es usado para la comunicar el Modelo y la Vista. Procesa toda la lógica de negocio, manipula datos usando componentes del Modelo e interactúa con la Vista mostrando los datos finales al usuario.

Cada uno de esos componentes se ha creado para manejar aspectos específicos de la aplicación. Es importante tener en cuenta que el Modelo no depende ni de la Vista ni del Controlador, sin embargo la Vista y el Controlador dependen del Modelo, ese es uno de los beneficios de separar por capas, además el Modelo se puede crear y testear de manera independiente a la representación visual.

El modelo de capas MVC (Modelo Vista Controlador) es muy popular, sin embargo existe un problema a la hora de definir el Controlador, ya que se puede definir de diferentes formas según el contexto. Afortunadamente las aplicaciones web ayudan a solucionar esta ambigüedad, permitiendo la fácil separación entre la Vista y el Controlador.

Existen muchas variaciones de este modelo de capas, por ejemplo:
- El modelo pasivo, cuando un Controlador manipula exclusivamente el Modelo.
- El modelo activo, cuando el Modelo cambia de estado sin que el Controlador se involucre.

Beneficios:
- Soporta múltiples vistas, ya que la Vista está separada del Modelo, se puede mostrar varias vistas de los mismos datos en diferentes momentos.
- Adaptado al cambio, la interfaz de usuario tiende al cambio, mucho más que el modelo de negocio, los usuarios quieren siempre diferentes fuentes, colores, etc.

Responsabilidades:
- Complejidad, el MVC introduce nuevos niveles de complejidad, pudiendo incrementarse notablemente la complejidad.
- El coste de frecuentes actualizaciones, quitando el Modelo de la Vista no significa que los desarrolladores ignoren la naturaleza de la Vista, algunos gráficos requieren algo de tiempo para ser renderizados.

3.6.- Modelo de capas en android

A la hora de implantar el modelo de capas Modelo Vista Controlador (MVC) en Android no siempre es muy intuitivo ni fácil, existe una clara tendencia a programar todo en cada Activity cuando se está desarrollando la aplicación.

A continuación se explicará cómo y dónde utilizar las distintas capas del modelo en el desarrollo de aplicaciones Android para intentar reducir la complejidad de las Activities.

3.6.1.- La Vista

La Vista estará compuesta por todos los componentes visuales de las Activities, por lo tanto la Vista está compuesta por todas las variables locales dentro de la función "onCreate" de cada Activity que se utilice para representar los elementos en la pantalla como los botones, listas, textViews, etc.

3.6.2.- El Modelo

La aplicación Android se comunica con el servidor a través de unos archivos Php del servicio web. Esos archivos se tienen que conectar con la base de datos, obtener datos y procesarlos para que se pueda responder adecuadamente según la solicitud de datos que requiera la aplicación Android.

Entonces los archivos Php están alojados en el servidor y representarán el Modelo de datos de la aplicación Android, se pretende que la aplicación se comunique mediante peticiones Post y Get con el Modelo para obtener información que luego será procesada por el Controlador de la aplicación.

3.6.3.- El Controlador

La aplicación Android dispondrá de unas clases que conecten el Modelo y la Vista. Entonces esas clases controladoras obtendrán los datos del Modelo para ser validados y los resultados se podrán visualizar mediante la Vista.

El Controlador estará presente también en todas las Activities mediante los Listeners, por ejemplo, si se quiere iniciar sesión, se introducirán los datos del usuario en la Activity y se pulsará sobre el botón “login”, el Controlador (Listener) detectará el evento de pulsación enviando esos datos al Modelo y los procesarán para que puedan actualizar la Vista en la Activity.

3.7.- Html5

Html5 viene a ser una revisión de Html (Hypertext Markup Language), es el lenguaje de programación estándar para describir el contenido y la apariencia de las páginas web, fuente de información [18].

Fue desarrollado para solventar problemas de compatibilidad que afectan al actual estándar Html4. Una de las grandes diferencias entre Html5 y las versiones previas, es que el antiguo estándar Html requiere plugins propietarios y APIs, motivo por el cual las páginas web construidas y testeadas en un navegador web, puede no cargar correctamente en otro. Html5 provee una interfaz común para cargar elementos fácilmente, por ejemplo, no hay necesidad de tener instalado el plugin de Flash en Html5.

Uno de los objetivos de su diseño es el de poder soportar multimedia sobre todo tipo de dispositivos móviles. Se han introducido nuevas características semánticas que cambiar verdaderamente la manera en que los usuarios interactúan con los documentos, incluido:

- Nuevos elementos y atributos para mejorar la flexibilidad.
- Eliminación de elementos redundantes.
- Capaz de soportar Drag and Drop (arrastrar y soltar) de un documento Html5 a otro.
- Edición offline.
- Un estándar común para almacenar datos en bases de datos SQL (Web SQL).

Html5 fue adoptado por el nuevo grupo de trabajo del consorcio World Wide Web (W3c) en 2007. Este grupo publica el primer borrador público de Html5 en Enero del 2008. En 2014 se publicó la versión definitiva, actualmente se sigue trabajando en nuevas versiones 5.1 y 5.2. Está soportado por todos los navegadores web modernos. Introduce una impresionante lista de nuevos elementos como que se muestran en la tabla 2.

Tabla 2: Etiquetas Html5

Etiqueta	**Descripción**
<article>	Define un artículo de un documento
<aside>	Define el contenido a un lado de la página
<bdi>	Aislar el texto para darle una dirección u orientación diferente, por ejemplo los textos de idiomas que se lee de derecha a izquierda o de arriba hacia abajo.
<details>	Define detalles adicionales que el usuario puede ver o estar oculto.
<dialog>	Define una caja o ventana de diálogo.
<figcaption>	Define un subtítulo dentro de una etiqueta <figure>
<figure>	Define el contenido de la etiqueta, pudiendo ser imágenes, diagramas, ilustraciones y ejemplos de código.
<footer>	Define el pie de página de un documento o una sección
<header>	Define la cabecera de un documento o sección
<main>	Define el contenido principal de un documento

<mark>	Define el texto marcado
<menuitem>	Define un comando/menú contextual personalizado
<meter>	Define gráficamente mediante una barra, un rango de valores.
<nav>	Define los links de navegación
<progress>	Representa el progreso de una tarea
<rp>	Define la información de un texto en formato Ruby que se mostrará en navegadores que no soportan anotaciones Ruby
<rt>	Define una explicación/pronunciación de caracteres, normalmente tipografía del este asiático
<ruby>	Define una anotación Ruby, normalmente tipografía de idiomas de países del este asiático
<section>	Define la sección de un documento
<summary>	Define un sumario para ocultar algunos contenidos largos o mostrar solo un resumen, viene junto a la etiqueta <details>
<time>	Define la fecha/hora
<wbr>	Define un salto de línea la división de palabras que no entran en el navegador, evitando así el scroll horizontal

Hasta Html4 los desarrolladores utilizaban con un identificador nombrado según la conveniencia del desarrollador, para organizar las secciones de una web como la cabecera, el pie de página, el menú, el contenido, etc. El hecho de llamarlo como uno quiera hace que para los motores de búsqueda sea imposible identificar el contenido de la página web de manera correcta, ahora se mejora mucho esta situación con los nuevos elementos Html5.

Así una web semántica permite que los documentos sean compartidos y reutilizados en todo tipo de aplicaciones. Normalmente una página de inicio se puede dividir en secciones para la introducción, contenido e información de contacto.

Es interesante tener en cuenta que los desarrolladores utilizan Html5 junto a una serie de herramientas tipo Framework, estos Frameworks contienen librerías CSS y archivos JavaScript que permiten a los desarrolladores incluirlo en el archivo Html.

Una Framework puede solucionar muchos problemas que pueda existir a la hora de desarrollar aplicaciones Html5. Por ejemplo, una buena Framework puede manipular y reciclar los elementos de DOM (Document Object Model) de manera que ayuden a mejorar el rendimiento, también ayudan a implementar fácilmente comportamientos sobre JavaScript como la transición de pantallas, el scroll, etc.

La limitación de memoria y rendimiento son la clave para considerar desarrollar Html5, los desarrolladores no solo deben tener cuidado con el control del flujo de la página, también tienen cuidado con la correcta implementación de DOM, mientras más complejo sea más tiempo tomará navegar por la página web.

En este proyecto no hará falta el uso de frameworks, la aplicación no será de alta complejidad visual y se prestará más atención al aspecto funcional que solo requerirá de Html5 básico junto a otras herramientas como CSS, Ajax y Php.

3.8.- Ajax

Asynchronous JavaScript And XML (Ajax), es un método de creación interactiva de aplicaciones para la web que procesa la petición del usuario inmediatamente. Ajax combina varias herramientas de programación JavaScript, dynamic Html (DHTML), Extensible Markup Language (XML), cascading style sheets (CSS), Document Object model (DOM), Microsoft object y XMLHttpRequest. Fuente de información [3] y [19].

Permite actualizar el contenido de las páginas web cargadas inmediatamente, cuando el usuario realiza una acción. Por ejemplo, el usuario debe esperar una nueva página cada vez que solicita o actualiza datos, sin embargo se puede mostrar y actualizar datos en el mismo sitio, sin esperar por toda la página web, solamente se debería esperar los nuevos datos.

Google Maps es un gran ejemplo de aplicación que usa Ajax. La interfaz permite al usuario cambiar la vista y manipular los mapas en tiempo real. Las aplicaciones Ajax no requieren de instalación de plugins, trabajan directamente con el navegador web, actualmente muchos navegadores ya soportan Ajax, anteriormente funcionaba solo en el navegador web Internet Explorer de Microsoft.

Las aplicaciones creadas con Ajax utilizan un motor que actúa como un intermediario entre el navegador del usuario y el servidor del cual se pide información, en vez de cargar toda la página web como se hace normalmente, el navegador web carga el motor de Ajax que mostrará los datos que el usuario quiere ver.

El motor trabaja continuamente en segundo plano usando JavaScript para comunicarse con el navegador web. Cuando se produce un evento para solicitar o actualizar datos, la página envía la llamada de JavaScript al motor de Ajax que mostrará la respuesta inmediatamente, cuando el servidor responde con los nuevos datos, por ejemplo utilizando XML u otro tipo de texto formateado, actualiza la página web con los nuevos datos en los campos necesarios.

Ajax no es una tecnología o paquete propietario, los desarrolladores web lo han ido utilizando en combinación con JavaScript y XML por muchos años. En la figura 16 se puede observar un diagrama con el proceso de petición y respuesta de información en Ajax.

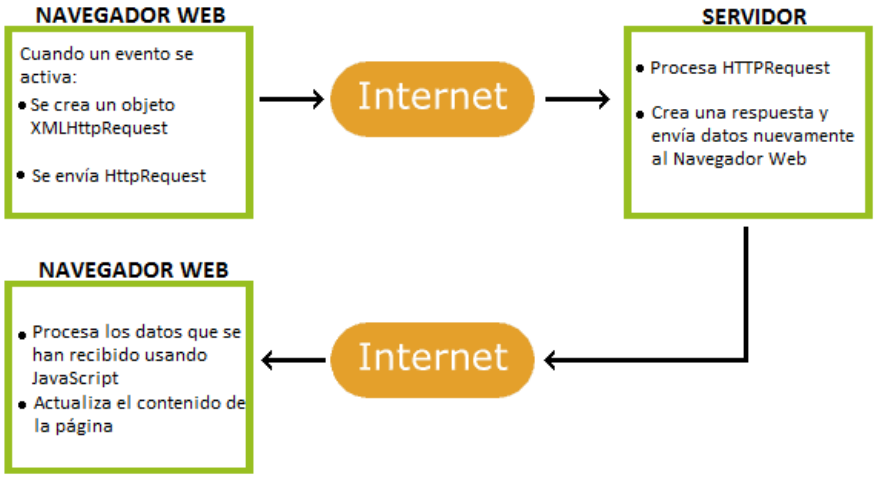


Figura 16: Petición y respuesta en Ajax

3.8.1.- El objeto XMLHttp

El objeto "XMLHttpRequest" es usado para intercambiar datos con el servidor en segundo plano. La sintaxis para crear el objeto XMLHttpRequest es "*variable = new XMLHttpRequest();"*. Para navegadores que no soportan el objeto "XMLHttpRequest" se crear un "ActiveXObject" como se ve a continuación:

```
var xhttp;
if (window.XMLHttpRequest) {
    // código para los navegadores modernos
    xmlhttp = new XMLHttpRequest();
 } else {
    // código para navegadores IE antiguos
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

3.8.2.- El objeto Request

Request de Ajax envía una petición al servidor, usando los métodos open() y send() del objeto XMLHttpRequest: "*xhttp.open("GET", "ajax_info.txt", true); xhttp.send();"*. Get

es el más simple, el más utilizado y es más rápido que Post, sin embargo se puede usar Post cuando:

- Se actualiza un fichero o la base de datos del servidor.
- Se envía grandes cantidades de datos al servidor, Post no tiene límites de tamaño.
- Se envía datos privados del usuario, Post es más robusto y seguro que Get.

3.8.3.- El objeto Response

El objeto Response tiene la propiedad "readyState" (0 petición no inicializada, 1 conectado al servidor, 2 petición recibida, 3 procesando la petición, 4 petición terminada y respuesta preparada) que mantiene el estado de XMLHttpRequest.

La propiedad "onreadystatechange" define la función que se ejecutará cuando "readyState" cambie. La propiedad "status" (200 "OK", 403 "Forbidden", 404 "Page not found", etc.) y "statusText" ("OK" o "Not Found") mantiene el estado del objeto XMLHttpRequest. Por ejemplo, cuando "readyState" es 4 y "status" es 200 la respuesta está preparada para ser evaluada como se ve a continuación:

```
function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
       document.getElementById("demo").innerHTML =
            this.responseText;
    }
    };
    xhttp.open("GET", "ajax_info.txt", true);
    xhttp.send();
}
```

3.9.- Php

Personal Home Page (Php) es un lenguaje e interpretador script similar a JavaScript y VBScript de Microsoft. Está disponible gratuitamente, es usado principalmente sobre servidores web Linux. Es una plataforma alternativa a la tecnología Active Server Page (ASP) de Microsoft (ASP solo funciona en sistemas operativos de Windows), fuente de información [3].

Php es gratuito y también está bajo licencia Open Source. Como ASP, el script de Php puede ser embebido junto a una página web sobre Html utilizando la etiqueta "<?php" para la apertura y "?>" para cerrar la utilización de código Php.

Antes de que la página web sea enviada al usuario que la ha solicitado, el servidor web llama al interpretador Php script y realiza las operaciones en Php, cuyo formato típico es

".php" ".php3," o ".phtml". Como ASP, Php puede trabajar con Html para funcionar como una página web dinámica, ya que permite modificar el resultado del contenido base.

Php tiene funciones para trabajar con arrays, calendarios, fechas, directorios, errores, archivos del sistema, filtros, ftp, http, libXML, mail, funciones matemáticas, MySqli, SimpleXML, string, XML y Zip.

Todas sus variables son case-sensitive, sin embargo todas las palabras clave, clases y funciones definidas por el usuario son Not case-sensitive. Las declaraciones terminan siempre con punto y coma, el código puede tener comentarios con "//", "#" o "/* comentario */".

3.9.1.- Php con bases de datos MySQL

Se puede conectar y manipular bases de datos MySql junto a Php, esa forma de trabajo se utilizará en este proyecto. La versión Php 5 y superiores trabajan con la base de datos utilizando la extensión "MySQLi" o "PDO" (Php Data Objects).

PDO trabaja sobre 12 diferentes sistemas de base de datos, MySQLi trabaja solo con bases de datos MySQL, en este proyecto se podrá utilizar cualquiera de las dos opciones. Es necesario conectarse con la base de datos para poder crear bases de datos, tablas, insertar uno o varios datos, obtener, borrar, actualizar y limitar datos.

3.9.2.- Php con Ajax

Ajax se utiliza para la mayoría de aplicaciones interactivas. Una vez que Ajax haya hecho una petición de datos al servidor, se puede comunicar con Php enviando parámetros para luego analizarlos y devolver datos específicos. Si Php requiere datos de una base de datos, es necesario que se abra una conexión con el servidor MySQL, se realiza la operación con la base de datos requerida y se devuelve esos datos en el formato que el usuario requiera.

3.10.- MySQL

Es necesario administrar múltiples datos y que esos datos sean accesibles concurrentemente por distintos usuarios, entonces es importante primero conocer el concepto de DBMS, para poder ver las distintas herramientas disponibles, conocer las ventajas y desventajas de cada uno y elegir el más adecuado para este proyecto.

3.10.1.- Sistemas administradores de bases de datos

Sistemas administradores de bases de datos (Data Base Management System o DBMS) se refiere a las tecnologías para administrar y crear bases de datos, haciendo posible que los usuarios finales puedan leer, actualizar y crear datos en la base de datos. Una DBMS sirve de interfaz entre la base de datos y los programas, aplicaciones o usuarios finales no

tienen por qué conocer dónde están almacenado físicamente esos datos, fuente de información [3] y [20].

Los distintos DBMS funcionan y trabajan utilizando distintos tipos de requerimientos y niveles de uso de los recursos del sistema donde esté implantado. Las ventajas y desventajas de los DMBS se pueden ver desde distintos puntos de vista, a continuación se indicará un resumen general de los más destacado y comentado de una DBMS.

Ventajas:

- Abstracción e independencia de los datos.
- Seguridad de los datos.
- Mecanismo para permitir el acceso concurrente a los datos entre varios usuarios.
- Balancea eficientemente las múltiples necesidades de las aplicaciones que compartan los mismos datos.
- Facilidad a la hora de hacer backups y recuperar datos.
- Robustez en la integración de datos.
- Logs que registran la actividad.
- Simple acceso para las APIs.
- Administración uniforme de datos.

Desventajas:

- Pueden llegar a ser muy complejos.
- Debido a la complejidad y funcionalidad utilizan mucha cantidad de memoria.
- La mayoría son sistemas centralizados que ante cualquier fallo, puede llegar a afectar a muchos usuarios.
- Las DBMS son sistemas para cualquier software en general, muchas veces la aplicación puede funcionar lento debido a que no está debidamente optimizado.
- El coste económico de la DBMS.

Los sistemas administradores de bases de datos más populares son:

- NoSQL DBMS
- IMDBMS (In-Memory DBMS)
- CDBMS
- Cloud Service
- RDBMS

3.10.1.1.- NoSQL DBMS

Está diseñado para administrar los problemas de escalabilidad y rendimiento que tienen las bases de datos relacionales. El sistema de administración de bases de datos NoSQL puede que no sea un candidato viable para transacciones financieras, pero sí puede ser una solución a las complejas matrices de ítems interrelacionados de las redes sociales o sistemas de localización usados para crear rutas de repartición y entrega.

Es importante saber que no hay una manera simple para trabajar con NoSQL DBMS, por ejemplo: Redis, MongoDB, Cassandra y Neo4j, son programas que pueden implantar NoSQL DBMS, pero todos son muy diferentes entre sí, cada uno con grandes ventajas y debilidades que se debe tener en cuenta. Sin embargo, muchas aplicaciones móviles y del Internet de las cosas (IoT) usan NoSQL DMBS para almacenar y administrar datos.

La mayoría de NoSQL DBMS son open source y son capaces de trabajar de manera flexible entre distintos esquemas ya que soporta infraestructuras distribuidas, escalabilidad horizontal y tolerancia a fallos que se puedan encontrar en Big Data.

3.10.1.2.-In-Memory DBMS

IMDBMS es conocido también como Main o In Memory DBMS, almacena datos fundamentalmente cargados en memoria principal en vez de almacenarlo en el disco duro, aunque soporta más de un solo tipo de almacenamiento.

Se usa principalmente para mejorar el rendimiento, ya que los datos están cargados en memoria principal y la velocidad de acceso a esta memoria es mucho más rápida que los discos duros, para asegurar la durabilidad de los datos, se mueve periódicamente los datos de la memoria principal a una memoria de almacenamiento no volátil.

Los sistemas IMDBMS son usados principalmente para aplicaciones que trabajan a tiempo real y que requieren un alto rendimiento como las telecomunicaciones, finanzas, defensa e inteligencia artificial, call center apps (CRM), aplicaciones de viajes y reserva y aplicaciones que trabajan en streaming.

Pueden administrar distintos tipos de bases de datos incluido NoSQL, bases de datos que almacenan datos en columnas en vez de filas, bases de datos gráficas (minería de datos) y bases de datos relacionales.

3.10.1.3.- DBMS basado en columnas

CDBMS es un sistema administrador de bases de datos que reorienta las filas a columnas. Se puede mejorar el rendimiento general de la base de datos, cuando se almacena grandes cantidades de datos en forma de columnas.

Es adecuado para los procesos de transacción de sistemas OLTP y procesos de análisis de los sistemas OLAP. No es adecuado para realizar queries variadas o soportar transacciones ACID (Atomicity, Consistency, Isolation and Durability).

El concepto de almacenamiento en columna no es nuevo, varias ideas de este tipo a sido implementadas en el pasado como parte de las bases de datos relacionales.

3.10.1.4.- Cloud Service

Un servicio de base de datos en la nube, almacena el contenido estructurado o no estructurado que reside en una plataforma computacional de manera privada, pública o en una nube híbrida. Existen dos modelos de bases de datos en la nube: el modelo tradicional y la base de datos que funciona como un servicio (DBaaS).

Los beneficios de la base de datos en la nube son:

- Eliminación de la infraestructura física, el proveedor de la nube provee los servidores, almacenamiento y otras infraestructuras y este proveedor es el responsable del mantenimiento y disponibilidad. La organización que contrata este servicio, es responsable de administrar el software y el contenido de la base de datos.
- Ahorro en costes, ya que se eliminar las infraestructuras físicas puede significar un importante ahorro de costes derivados del mantenimiento de dicha infraestructura, también se reduce el coste del personal y el consumo eléctrico que será considerablemente reducido.

Desde un punto de vista de software, diseño y estructura, el funcionamiento de la base de datos tradicional no es diferente al de la base de datos en la nube, aunque para acceder a la base de datos en la nube se requiere conexión permanente a internet. El acceso a través de queries o vía llamada a APIs debería ser también igual.

Sin embargo las diferencias son evidentes con los tiempos de respuesta, acceder a la base de datos en la nube requiere más tiempo de respuesta sin embargo esos tiempos actualmente se van reduciendo.

3.10.1.5.- DBMS Relacional

RDBMS es un sistema de base de datos basado en el modelo relacional que almacena datos en forma de tablas relacionadas entre sí. Ideal para poder llevar a cabo el almacenamiento de datos de este proyecto, ya que el formato de las tablas es simple, fácil de entender y utilizar, permite el acceso de varios usuarios de manera simultánea, controlando los permisos y privilegios que da el administrador a los demás usuarios.

La base de datos basada en el modelo relacional no es la estructura más rápida, pero se debe tener en cuenta que es mucho más simple para la mayoría de aplicaciones, provee de herramientas para un fácil mantenimiento, test, reparaciones y backups de las bases de datos.

Por supuesto, soporta el lenguaje genérico SQL (Structured Query Language), cuya sintaxis es estándar y fácil de aprender. Los programas para trabajar con RDBMS más importantes son: MS SQL Server, IMB DB2, Oracle y MySQL.

3.10.1.5.1.- MS SQL Server

Es un RDBMS (administrador de bases de datos relacional) que soporta una amplia variedad de procesos de transacción, inteligencia de negocios y análisis de aplicaciones en entornos corporativos.

Está fabricado sobre SQL, el código original fue desarrollado en 1980, entre 1995 y 2016 se han liberado 10 versiones de SQL Server, en la última versión de Junio del 2016, se ha desarrollado nuevas características como análisis operacionales en tiempo real, virtualización de datos, reportes sobre dispositivos móviles, soporte para hybrid cloud que permite trabajar con bases de datos combinando sistemas locales y servicios públicos en la nube para reducir gastos, además se ha incrementado el soporte para Big Data y otras aplicaciones para realizar poder realizar avanzados análisis a través de SQL Server R Services.

SQL Server 2014 añadió In-Memory OLTP, el cual permite a los usuarios realizar transacciones de aplicaciones OLTP sobre datos almacenados en memoria principal. SQL Server trabaja exclusivamente en sistemas Windows, pero en 2016 se anunció que habrá soporte para poder trabajar en sistemas Linux,

3.10.1.5.2.- IMB DB2

Es parte de la familia de RDBMS (administradores de bases de datos relacionales), está disponible para sistemas basados en UNIX, ofrece para mainframe OS/390, sistemas operativos VM y sistemas de rango medio As/400, fue el primer producto en utilizar el lenguaje SQL. Es capaz de lidiar con millones de transacciones y grandes cantidades de datos, administrado a través GUI de la multiplataforma de java.

3.10.1.5.3.- Oracle

Fue la primera compañía en comercializar bases de datos relacionales (RDBMS) a través de su producto Oracle Database y fue también el primero en dar soporte al estándar ANSI SQL y en comercializar RDBMS.

Las últimas versiones de la base de datos de Oracle funciona sobre plataformas Linux x86, Linux Itanium, Microsoft Windows (32 y 64 bits), Solaris x86, Solaris SPARC (64bits), AIX5L, HP-UX, PA-RISC, HP-UX Itanium, HP Tru64 UNIX, IMB z/OS y Mac OS X Server.

3.10.1.5.4.- MySql

Es un administrador de bases de datos relacionales liberado bajo licencia GNU (General Public License) y desarrollado por la compañía sueca MySQL AB. Está disponible en muchas plataformas incluido Microsoft Windows, Linux y Mac OS X. Es el más popular, dependiendo de su uso comercial o no comercial, puede ser gratuito o utilizar la versión de pago.

Es open source y gratuito (también dispone de una licencia comercial), la más popular para aplicaciones y web pequeñas o medianas, fácil de administrar y utilizar, es comprensible y tiene todo lo necesario para poder desarrollar una base de datos estable y adecuada a este proyecto, no se busca crear grandes data warehouses o que almacene terabytes de información.

El servidor de este proyecto tiene instalado phpMyAdmin, para poder administrar el servidor MySQL, es compatible con bases de datos de MySQL 5.5 o superior.

3.10.2.- Triggers

Un trigger en una base de datos es parte del lenguaje estructurado SQL, es un proceso especial que se dispara automáticamente cuando ocurre un evento específico a través de la manipulación del lenguaje de manipulación de datos (DML), como cuando se inserta, actualiza o elimina datos de una tabla o de una vista, fuente de información [21] y [22].

En este proyecto se busca principalmente preservar la integridad de datos y prevenir transacciones inválidas. El trigger es parte de una gran variedad de eventos del lenguaje de definición de datos (DDL). Un trigger se puede crear por distintos motivos como se ve a continuación:

- Para generar valores de una columna automáticamente.
- Reforzar la integridad referencial.
- Para almacenar información sobre la auditoría de tablas.
- Auditorias.
- Replicación síncrona de tablas.
- Mejorar la seguridad.
- Prevenir transacciones inválidas.

3.11.- Modelo de capas web

La implantación del modelo de capas Modelo Vista Controlador en el desarrollo de la aplicación web es mucho más fácil de ver que en el desarrollo de la aplicación Android, ya que se pueden separar los ficheros según la funcionalidad y cada funcionalidad tendrá las tres capas del modelo.

3.11.1.- La Vista

La Vista se encargará únicamente de mostrar la interfaz de trabajo al usuario. Un archivo con código Html estará preparado para mostrar los componentes necesarios para poder trabajar con la gestión de datos.

3.11.2.- El Modelo

Es un archivo Php que tendrá la lógica de los datos, la invocará el Controlador y se conectará con la base de datos para verificar si la petición es consistente y válida. La

respuesta se devolverá al Controlador en formato Json para que dé la orden de mostrar el resultado en la Vista.

3.11.3.- El Controlador

El Controlador se encargará de controlar toda la funcionalidad de la aplicación web junto al Modelo y la Vista. Por ejemplo, para la funcionalidad de inicio de sesión, el Controlador verifica que nadie haya iniciado sesión previamente, luego da la orden de actualizar la Vista para que el usuario introduzca sus datos de sesión, cuando se haga clic en el botón “login”, el Controlador estará atento al evento “onClick” que tiene asignado el botón de inicio de sesión.

El Controlador llamará mediante código Ajax al Modelo para verificar si el usuario introducido es el correcto. Entonces el Controlador actualizará la Vista con el resultado de la comprobación, si todo es correcto se dará la orden a la Vista de mostrar la interfaz con todos los componentes adecuados para que el usuario registrado pueda trabajar.

4.- METODOLOGÍA Y RESULTADOS

En este capítulo se va a explicar el desarrollo del prototipo basado en una arquitectura de diseño, planificando su ciclo de vida, descubriendo sus requisitos, diseñando, implementando y probando el prototipo.

4.1.- Planificación del proyecto

Es necesario planificar antes de comenzar a desarrollar, en los siguientes apartados se mostrará paso a paso las distintas etapas y metodologías del proceso de desarrollo del prototipo de este proyecto.

4.1.1.- Ciclo de vida

El ciclo de vida que este prototipo seguirá es el modelo en cascada, avanzando entre sus etapas de manera secuencial, sin que se solapen ni mezclen. Es un modelo sencillo, fácil de comprender, aprender y utilizar.

Se van a seguir e implementar las etapas de planificación, especificación, diseño y resultados, como se describe a continuación:

- Planificación: se verificarán los requerimientos del usuario, se creará un boceto manual para poder tener una primera visualización de la interfaz, luego se diseñará un boceto digital para tener una idea más fiel del prototipo y se creará un storyboard para ilustrar con ejemplos de uso las distintas actividades de los usuarios en las aplicaciones web y Android.
- Especificación: se determinarán las características del prototipo a través del diseño del diagrama de dependencias y el diagrama de carril de las aplicaciones web y Android.
- Diseño: el método de prototipado que se va a seguir es el método codificado. Se comenzará a preparar el código del prototipo con los mismos lenguajes utilizados en el desarrollo del producto final, pudiendo así reciclarlos.
- Resultados: se realizará una revisión interna para presentar el resultado del primer prototipo al usuario.

El modelo en cascada es uno de los más utilizados ya que ayuda a detectar los errores en las primeras etapas, permitiendo minimizar los gastos de la planificación. Sin embargo, cualquier error de diseño que se detecte en la etapa de prueba, induce al inmediato rediseño, alargando el tiempo de desarrollo. Como este proyecto es relativamente pequeño, los inconvenientes del rediseño pueden ser asumibles.

4.2.- Captura de requisitos

Se ha planificado el lugar de la reunión en el centro deportivo local, en una oficina que se suele utilizar para presentaciones y videoconferencias. A la captura de requisitos se ha

enviado al responsable del desarrollo de software, por parte del cliente asistieron un entrenador representante de los equipos y el encargado de diseñar los torneos. Mediante brainstorming (lluvia de ideas) se buscará captar ideas.

El organizador del torneo indica que es necesario que existan usuarios que desde una página web puedan crear torneos, partidos, equipos, jugadores y todo elemento necesario para poder tener un torneo en condiciones.

Además, el organizador propone que las autoridades encargadas de validar los partidos oficiales tengan un usuario con permisos necesarios para poder actualizar los eventos de los partidos desde un dispositivo móvil. También indica que los demás usuarios (usuarios públicos) podrán ver los resultados y todo evento generado en los partidos en tiempo real.

El entrenador sugiere que la aplicación pueda mostrar en la primera pantalla los torneos favoritos, en caso de no tener torneos favoritos se debe poder buscar torneos para elegirlos y añadirlos como favoritos. También quiere que se pueda elegir un torneo y mostrar en otra pantalla los partidos de ese torneo, además cada cambio o actualización de los partidos se debe notificar como se hace en muchas aplicaciones.

Finalmente, el responsable del desarrollo del programa sugiere que las autoridades y los usuarios públicos trabajen desde la misma aplicación Android y todos están de acuerdo en que el usuario organizador del torneo, pueda administrar torneos y usuarios desde una página web.

4.2.1.- Validación de Requisitos

Se ha preparado un documento (tabla 3) con la validación de los requisitos de la anterior reunión y sus resultados se muestran a continuación:

Tabla 3: Validación de requisitos

REQUISITO	TIPO	PRIORIDAD	VALIDADO SI/NO	RESULTADO	CAMBIOS
Deben existir usuarios con permisos para crear torneos y todas las características del torneo desde una página web.	Funcional, Diseño y Audiencia	Alta	Si	1º Reunión	

Deben existir usuarios con permisos para actualizar los eventos de un partido.	Funcional, Audiencia y Diseño	Alta	Si	1º Reunión	
El usuario que modifica partidos, puede iniciar sesión desde cualquier pantalla de la aplicación Android.	Funcional y Diseño	Alta	Si	1º Reunión	En el primer prototipo los usuarios con permisos podrán modificar los resultados de los partidos.
Deben existir usuarios públicos para ver los eventos de los partidos de cualquier torneo desde la aplicación Android.	Funcional, Audiencia y Diseño	Alta	Si	1º Reunión	
La aplicación Android debe mostrar como primera pantalla los torneos favoritos (con la opción de buscar y elegir torneos para añadirlos a favoritos).	Funcional y Diseño	Media	Si	1º Reunión	En el primer prototipo la primera pantalla mostrará todos los torneos disponibles.

Todos los cambios y nuevos eventos de los partidos se deben notificar (como lo hace Whatsapp) y actualizarse de manera automática.	Funcional y Diseño	Alta	Si	1º Reunión	

4.2.2.- Storyboard

Se han creado tres perfiles de usuarios que representan las distintas actividades que se pueden realizar dentro de las aplicaciones web y Android.

4.2.2.1.- Perfil 1

Los datos del primer perfil serían:

- Nombre: Learco Zavala Abrego
- Edad: 20 años
- Estado civil: Soltero
- Donde trabaja: Peluquero
- Nivel informático: Bajo
- Frecuencia de uso de aplicaciones Android: Muy frecuente

Escenario 1, trabaja en una peluquería por el día y tiene 2 hermanos. Por la noche entrena y juega football en un equipo regional, sus hermanos también juegan football en otros equipos de divisiones inferiores.

Está al tanto de los partidos que juegan sus hermanos, por lo tanto entra a la aplicación para ver los partidos de sus hermanos que están jugando en lugares diferentes en el mismo torneo.

Uno de sus hermanos juega en el “Equipo1”. Abre la aplicación Android, elige el torneo “Orihuela” categoría sub 15 y la aplicación le envía a la pantalla de los partidos de ese torneo. Él está pendiente de los partidos del “Equipo1” vs “Equipo2” que como se ve en la figura 17 de momento van 0 a 0.

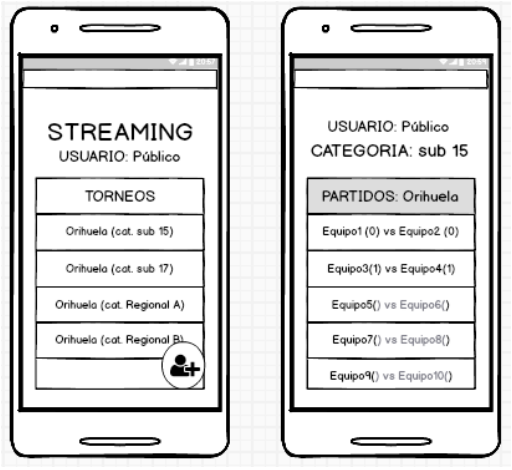


Figura 17: Torneos y partidos del prototipo

El "Equipo1" mete un gol, suena la notificación del teléfono, desplaza la zona de notificación para ver en detalle la notificación y se puede ver ahí que el "Equipo1" ha marcado un gol, ahora va ganando 1 a 0. Pulsa sobre la notificación y en la pantalla de partidos se puede ver el resultado actualizado y un pequeño mensaje indicando el evento ocurrido, como se ve en la figura 18.

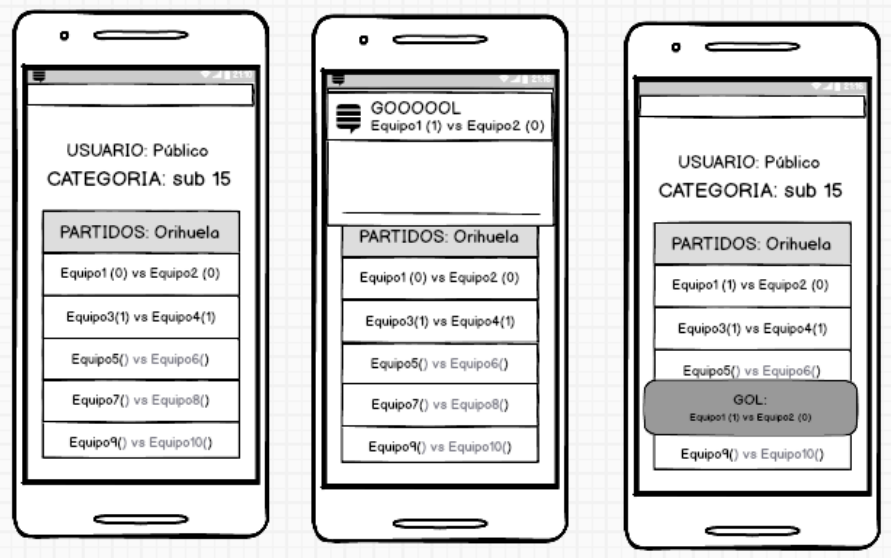


Figura 18: Notificación de un evento del partido en el prototipo

4.2.2.2.- Perfil 2

Los datos del segundo perfil serían:

- Nombre: Anastas Gonzales Romero
- Edad: 24 años
- Estado civil: Soltero
- Donde trabaja: Estudiante
- Nivel informático: Alto
- Frecuencia de uso de aplicaciones Android: Muy frecuente

Escenario 2: estudia un grado de formación profesional durante la semana, pero los fines de semana es una autoridad encargada de validar partidos oficiales de una liga sub 15.

Abre la aplicación, inicialmente se considera usuario público por lo tanto no puede modificar los resultados, sólo puede ver las actualizaciones. Previamente le han dado de alta como usuario administrador de la aplicación, por lo tanto puede iniciar sesión en la aplicación Android, como se ve en la figura 19.

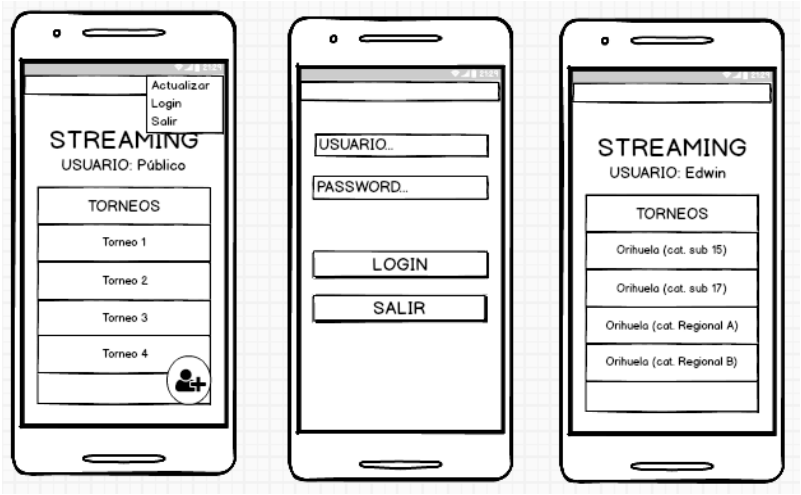


Figura 19: Inicio de sesión en el prototipo

Este usuario está encargado de modificar los resultados del partido del "Equipo1" vs "Equipo2". El "Equipo1" marca un gol, él se encarga de registrar el gol en los papeles oficiales y al mismo tiempo actualiza el resultado en la aplicación para que todos los demás usuarios se enteren del nuevo resultado, como se ve en la figura 20.

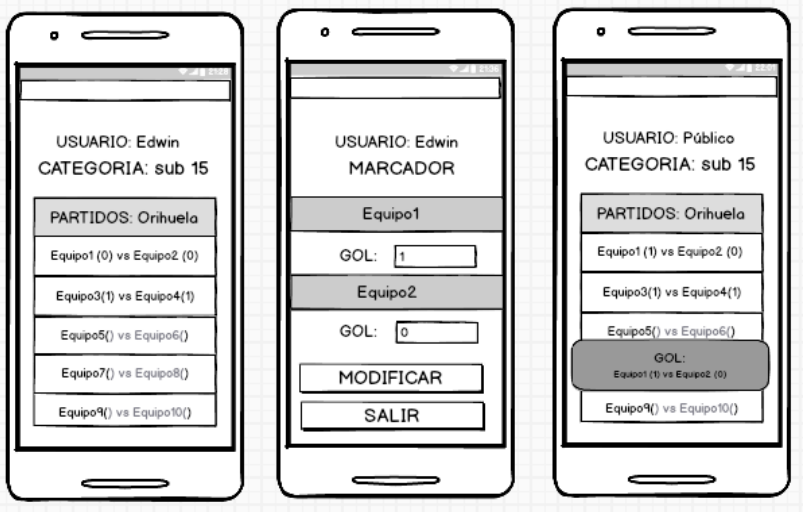


Figura 20: Modificando un evento en el prototipo

4.2.2.3.- Perfil 3

Los datos del tercer perfil serían:

- Nombre: Rinaldo Jaime Portillo
- Edad: 30 años
- Estado civil: Casado
- Donde trabaja: Ingeniero civil
- Nivel informático: Medio

- Frecuencia de uso de aplicaciones Android: Medio

Escenario 3: junto a unos amigos decide crear un torneo de football fugaz. A pesar de ser un partido amateur, considera que él mismo puede crear el torneo en la página web.

Inicia sesión como usuario con permisos web. Luego puede elegir entre gestionar los usuarios que gestionen los eventos del partido o gestionar los torneos, como se ve en la figura 21.

Figura 21: Gestión web del prototipo

Elige gestión de torneos y prepara la información del torneo, además desde la gestión de usuarios crea 2 usuarios que controlarán los partidos y actualizarán los resultados.

4.2.2.- Casos de uso

Los distintos roles de los usuarios se mostrarán en el diagrama de casos de uso y en sus complementos explicarán en detalle todas las actividades que pueden realizar.

4.2.2.1.- Diagrama de casos de uso

El diagrama de casos de uso se puede ver en la figura 22.

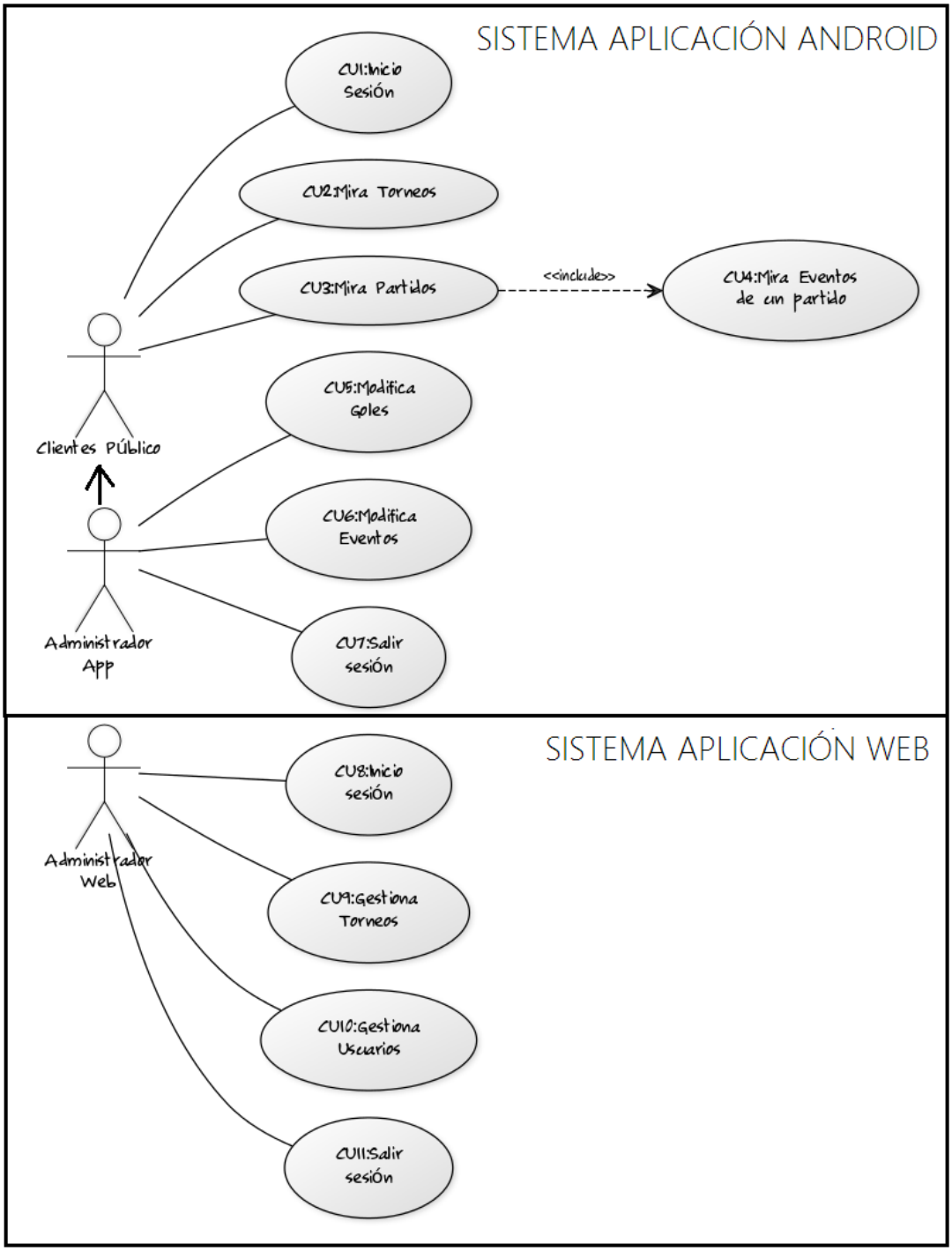


Figura 22: Diagrama de casos de uso

4.2.2.2.- Complementos del diagrama de casos de uso 1

En la tabla 4, tabla 5 y tabla 6 se definen los tres usuarios que forman parte de la aplicación.

Tabla 4: Definición del usuario público

ACTOR	Usuario público
DESCRIPCIÓN	Cualquier usuario (usuario público) que se haya descargado e instalado la aplicación Android para ver en streaming los partidos de distintos torneos.
CASOS DE USO RELACIONADO	CU1,CU2,CU3,CU4

Tabla 5: Definición del administrador de la aplicación

ACTOR	Administrador de la aplicación
DESCRIPCIÓN	Usuario que tiene la aplicación Android instalada y ha iniciado sesión con los permisos suficientes para poder modificar goles y añadir eventos de los partidos. Estos usuarios son los responsables de administrar los partidos físicamente (los controladores de los partidos de football) y al mismo tiempo actualizan marcadores desde la aplicación.
CASOS DE USO RELACIONADO	CU1,CU2,CU3,CU4,CU5,CU6

Tabla 6: Definición del administrador web

ACTOR	Administrador Web
DESCRIPCIÓN	Usuario que tiene permisos para poder gestionar usuarios y gestionar torneos. Accede desde la web, no puede realizar las actividades de los otros usuarios, y los otros usuarios tampoco pueden realizar las actividades del administrador web.
CASOS DE USO RELACIONADO	CU7,CU8,CU9,CU10

4.2.2.3.- Complementos del diagrama de casos de uso 2

En las tablas 7-17 se definen las distintas actividades (casos de uso) de los usuarios dentro del prototipo.

Tabla 7: Caso de uso 1, iniciar sesión en Android

C.U. 1	Iniciar sesión en la aplicación Android
Actores	Usuario público
Descripción	Se carga la pantalla de inicio de sesión desde el menú o icono de inicio sesión para escribir el usuario y la contraseña.
Dependencias	Ninguna
Precondición	Todos los usuarios administradores de la aplicación Android, deben ser dados de alta por el usuario administrador web. Tener instalado la aplicación Android. Haber elegido la opción “Inicio sesión”.
Secuencia Normal	Paso 1:Se carga la Vista para poder iniciar sesión. Paso 2:Escribe su usuario. Paso 3:Escribe su contraseña. Paso 4:Clic en Iniciar sesión. Paso 5:El Controlador actualiza la Vista para el usuario registrado.
Pos condición	
Excepciones	Si el usuario o la contraseña están mal escritos, se muestra un mensaje que describe el problema. Si no tiene los permisos necesarios, aparece otro mensaje describiendo el problema.
Rendimiento	
Frecuencia	media
Importancia	Alta

Urgencia	Alta
Estado	Fase prototipo terminada
Estabilidad	Buena
Comentarios	Iniciar sesión habilita la posibilidad de modificar marcadores y eventos del partido.

Tabla 8: Caso de uso 2, mirar torneo en Android

C.U. 2	Mirar torneos en la aplicación Android
Actores	Usuarios públicos y administrador de la aplicación
Descripción	Se crea una lista con los torneos que se estén disputando o que se disputarán, a partir de la fecha y hora en el que se encuentre.
Dependencias	
Precondición	Tener la aplicación Android instalada y tener acceso a internet.
Secuencia Normal	Paso 1:Se carga la Vista (mainActivity) y el Controlador(Listeners) de los torneos. Paso 2:El Controlador busca datos de los torneos en el Modelo. Paso 3:El Modelo responde con los datos de los torneos al Modelo en formato Json. Paso 4:El Controlador actualiza la Vista con los datos del torneo, creando una lista de torneos con sus respectivos listeners.
Pos condición	Tener acceso a internet.
Excepciones	Si no existen datos en internet, no se llena la lista de torneos y sale mensaje personalizado, indicando que no hay internet o no hay datos de torneos.
Rendimiento	
Frecuencia	Alta

Importancia	Alta
Urgencia	Alta
Estado	Fase prototipo terminada
Estabilidad	Buena
Comentarios	

Tabla 9: Caso de uso 3, mirar partidos en Android

C.U. 3	Mira partidos en la aplicación Android
Actores	Usuarios públicos y administrador de la aplicación
Descripción	Se crea una lista de partidos según el torneo que se haya elegido para poder ver los marcadores y eventos de los partidos. Si hay cambios, llega un mensaje indicando el cambio para poder actualizar los datos
Dependencias	CU2. Si no existen datos de torneos que se estén disputando, no se puede acceder a los partidos.
Precondición	Haber elegido en la Vista de torneos, previamente un torneo de la lista de torneos. Tener acceso a internet.
Secuencia Normal	Paso 1:se carga una nueva Vista (matchActivity) con su Controlador (Listeners), para los partidos. Paso 2:El Controlador busca datos de los partidos de un torneo, en el Modelo. Paso 3:El Modelo responde con los datos de los partidos de un torneo, al Modelo en formato json. Paso 4:El Controlador actualiza la Vista con los partidos, creando una lista de partidos con sus respectivos listeners.

Pos condición	Tener acceso a internet.
Excepciones	Si no existen datos en internet, no se llena la lista de partidos y sale mensaje personalizado, no hay datos de los partidos de un torneo específico. Si no existe internet, no se llena la lista de partidos y se muestra un mensaje personalizado, indicado que no tiene acceso a internet.
Rendimiento	El listener de cada partido creado por el Modelo, es diferente según el tipo de usuario se cargará una nueva Vista para los eventos del partido elegido: El usuario público sólo puede elegir el partido, cargar la nueva Vista para ver el marcador y los eventos del partido. El usuario administrador, elige un partido, carga la nueva Vista para poder modificar el marcador y los eventos del partido.
Frecuencia	Alta/media/baja
Importancia	Alta/media/baja
Urgencia	Alta/media/baja
Estado	Fase prototipo terminada
Estabilidad	Buena
Comentarios	Sólo está desarrollado la Vista para ver y modificar marcadores, no está desarrollada la Vista para ver y modificar otros eventos de un partido elegido.

Tabla 10: Caso de uso 4, mirar eventos del partido en Android

C.U. 4	Mira eventos de un partido en la aplicación Android
Actores	Usuarios públicos, administrador de la aplicación Android
Descripción	El usuario público y administrador de la aplicación Android, puede ver los marcadores y eventos de un partido, si hay cambios en el marcador, se le notifica mediante el servicio de mensajería, cuando responde el mensaje se actualizan sus datos.

Dependencias	CU3. Si no existen datos de partidos que se estén disputando, no se puede acceder a ver los eventos.
Precondición	Tener Acceso a internet. Haber elegido un partido.
Secuencia Normal	Paso 1:Se carga una nueva Vista y Controlador para poder ver *marcador y eventos* de un partido seleccionado. Paso 2:El Controlador pide al Modelo datos (marcador y eventos) y actualiza la Vista de *marcador y eventos*. Paso 3:El Controlador verifica si los datos de *marcador y eventos* ya existen y están almacenados localmente o no existen, busca mediante el Modelo los datos en base de datos mediante internet. Paso 4:El Controlador actualiza la Vista con los datos de los *marcadores y eventos* de un partido elegido. Paso 5:Un servicio Controlador en background verifica cambios en el *marcador y los eventos* en el servidor, si hay cambios, notifica dichos cambios mediante el servicio de mensajería y actualiza los datos almacenados localmente.
Pos condición	Tener acceso a internet.
Excepciones	
Rendimiento	Dependerá si es usuario público sólo puede ver los marcadores y eventos, si es usuario administrador web sólo puede modificar los marcadores y eventos.
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Fase prototipo en desarrollo no terminado
Estabilidad	Buena

Comentarios	Se está desarrollando la gestión de eventos.

Tabla 11: Caso de uso 5, modificar goles en Android

C.U. 5	Modifica goles en la aplicación Android
Actores	Administrador de la aplicación Android
Descripción	Una vez identificado como usuario administrador de la aplicación, modificar marcadores de los equipos y guardarlos.
Dependencias	CU1, CU3. Elegir un partido.
Precondición	Haber iniciado sesión como administrador de la aplicación. Haber elegido un partido para modificar. Tener acceso a internet.
Secuencia Normal	Paso 1:Modifica los goles de los equipos y guarda los marcadores. Paso 2:El Controlador actualiza los datos locales y la base de datos del servidor en internet, mediante el Modelo. Paso 3:Un servicio en background notifica a los demás usuarios el cambio realizado.
Pos condición	Tener acceso a internet.
Excepciones	Si no se modifica nada, sale un mensaje indicando que debe modificar algo para poder continuar. Si los datos a modificar son nulos o no se pone nada, sale un mensaje indicando que debe poner datos en el formato adecuado.
Rendimiento	Restricciones si existe
Frecuencia	Alta
Importancia	Alta

Urgencia	Alta
Estado	Fase prototipo terminada
Estabilidad	Buena
Comentarios	

Tabla 12: Caso de uso 6, modificar eventos en Android

C.U. 6	Modifica eventos en la aplicación Android
Actores	Administrador de la aplicación Android
Descripción	Una vez identificado como usuario administrador de la aplicación, puede modificar los eventos y guardarlos.
Dependencias	CU1, CU3. Elegir un partido.
Precondición	Haber iniciado sesión como administrador de la aplicación. Haber elegido un partido a modificar. Tener acceso a internet.
Secuencia Normal	Paso 1:Elige los eventos de una lista y guarda el evento. Paso 2:El Controlador actualiza los datos locales y la base de datos del servidor en internet, mediante el Modelo. Paso 3:Un servicio en background notifica a los demás usuarios el cambio realizado.
Pos condición	Tener acceso a internet.
Excepciones	Si no se modifica nada, sale un mensaje indicando que debe modificar algo para poder continuar. Si los datos a modificar son nulos o no se pone nada, sale un mensaje indicando que debe poner datos en el formato adecuado.
Rendimiento	

Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Fase desarrollo, no finalizado para el prototipo
Estabilidad	Ninguna
Comentarios	

Tabla 13: Caso de uso 7, cerrar sesión en Android

C.U. 7	Salir sesión de la aplicación Android
Actores	Administrador de la aplicación
Descripción	Después de haberse identificado como usuario administrador de la aplicación, desde cualquier parte de la aplicación se puede cerrar sesión y vuelve a ser usuario público.
Dependencias	CU1
Precondición	Haber iniciado sesión como usuario administrador de la aplicación.
Secuencia Normal	Paso 1:Elegir desde el menú cerrar sesión. Paso 2:El Controlador actualiza la Vista, para ver los datos como usuario público.
Pos condición	
Excepciones	
Rendimiento	
Frecuencia	Alta
Importancia	Media

Urgencia	Baja
Estado	Fase prototipo terminada
Estabilidad	Buena
Comentarios	

Tabla 14: Caso de uso 8, iniciar sesión en la web

C.U. 8	Inicio sesión en la aplicación web
Actores	Administrador web
Descripción	El administrador web, puede iniciar sesión desde la página web. Si es usuario administrador de la aplicación, no puede iniciar sesión en la página web.
Dependencias	Ninguna
Precondición	Ser administrador web para poder iniciar sesión. Abrir la página web. Tener acceso a internet.
Secuencia Normal	Paso 1:El Controlador detecta que ningún usuario se ha identificado, actualiza la Vista para iniciar sesión. Paso 2:Escribe su usuario. Paso 3:Escribe su contraseña. Paso 4:Clic en Iniciar sesión. Paso 5:El Controlador actualiza la Vista para el usuario identificado.
Pos condición	Tener acceso a internet.

Excepciones	Si el usuario que se identifica es administrador de la aplicación Android, no inicia sesión y muestra un mensaje personalizado, indicando que no tiene los permisos suficientes para trabajar en la web. Si no escribe usuario y contraseña, se muestra un mensaje indicando que debe escribir un usuario y contraseña. Si el usuario o contraseña es erróneo, se muestra un mensaje indicando el error y pidiendo que vuelva a escribir el usuario y contraseña.
Rendimiento	Bueno
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Fase prototipo terminada
Estabilidad	Buena
Comentarios	

Tabla 15: Caso de uso 9, gestión de torneos en la web

C.U. 9	Gestión torneos en la aplicación web
Actores	Administrador web
Descripción	Una vez que el usuario se haya identificado, se puede añadir, modificar y eliminar todos los datos relativos a los torneos de football.
Dependencias	CU8.
Precondición	Haberse identificado como usuario administrador web. Haber elegido la opción del menú “Gestión de torneos”. Tener acceso a internet.

Secuencia Normal	Paso 1:Elegir los datos de los torneos a modificar de la Vista. Paso 2:El Controlador de torneos actualiza los datos modificados mediante el Modelo. Paso 3:El Controlador actualiza la Vista con los nuevos datos.
Pos condición	Tener acceso a internet.
Excepciones	Si los datos son obligatorios y al modificar no están presentes, se indica mediante un mensaje que debe añadir cosas a la parte obligatoria. Si los datos no son consistentes o en algunos casos repetidos, un mensaje indica el problema.
Rendimiento	
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Fase prototipo no terminada.
Estabilidad	Buena
Comentarios	Todavía no se ha completado la modificación de todos los datos de los torneos.

Tabla 16: Caso de uso 10, gestión de usuarios en la web

C.U. 10	Gestión usuarios en la aplicación web
Actores	Administrador web
Descripción	Una vez que el usuario se haya identificado, se puede añadir, modificar y eliminar todos los datos relativos a usuarios administradores web y administradores de la aplicación Android.
Dependencias	CU8.

Precondición	Haberse identificado como usuario administrador web. Haber elegido la opción del menú “Gestión de usuarios”. Tener acceso a internet.
Secuencia Normal	Paso 1:Elegir los datos de los usuarios a modificar de la Vista. Paso 2:El Controlador de torneos actualiza los datos modificados mediante el Modelo. Paso 3:El Controlador actualiza la Vista con los nuevos datos.
Pos condición	Tener acceso a internet.
Excepciones	Si los datos son obligatorios y al modificar no están presentes, se indica un mensaje indicando añadir cosas a la parte obligatoria. Si los datos no son consistentes o en algunos casos repetidos, un mensaje indica el problema.
Rendimiento	
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Fase prototipo terminada
Estabilidad	Buena
Comentarios	

Tabla 17: Caso de uso 11, cerrar sesión en la web

C.U. 11	Salir sesión
Actores	Administrador web

Descripción	Una vez identificado como usuario administrador web, la opción cerrar sesión está disponible.
Dependencias	CU8
Precondición	Haber iniciado sesión como usuario administrador web. Tener acceso a internet.
Secuencia Normal	Paso 1:Elegir desde la Vista la opción *Cerrar sesión*. Paso 2:El Controlador se encarga de eliminar los datos el usuario que ha iniciado sesión. Paso 3:El Controlador actualiza la Vista para que se pueda iniciar sesión nuevamente.
Pos condición	Tener acceso a internet.
Excepciones	
Rendimiento	
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Estado	Fase prototipo terminada
Estabilidad	Buena
Comentarios	

4.3.- Diseño de la aplicación

Ahora se muestra el diseño de la interfaz gráfica, el diseño de diagramas para las actividades del prototipo, el diseño del diagrama de clases de la aplicación Android y el diseño de la base de datos.

4.3.1.- Interfaz gráfica

Se ha preparado distintos bocetos del diseño de las aplicacines Android y web, un primer boceto manual y posteriormente un boceto digital.

4.3.1.1 Boceto Manual

En las figuras 23 y 24 se muestran los bocetos manuales para la aplicación Android y web.

Figura 23: Boceto manual de la aplicación Android

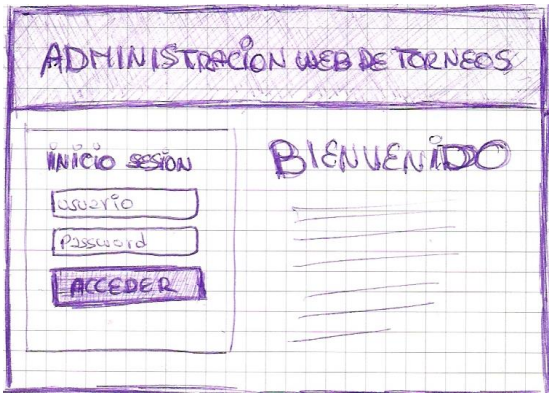


Figura 24: Boceto manual de la aplicación web

4.3.1.2 Boceto Digital

En las figuras 25 y 26 se muestran los bocetos digitales para las aplicaciones Android y web.

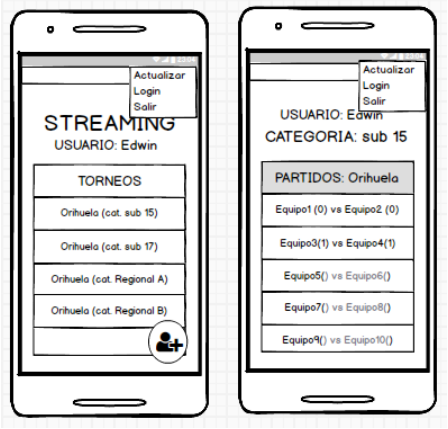


Figura 25: Boceto digital de la aplicación Android

Figura 26: Boceto digital de la aplicación web

4.3.2.- Diagrama de dependencias

Las dependencias de la aplicación web se muestran en la figura 27:

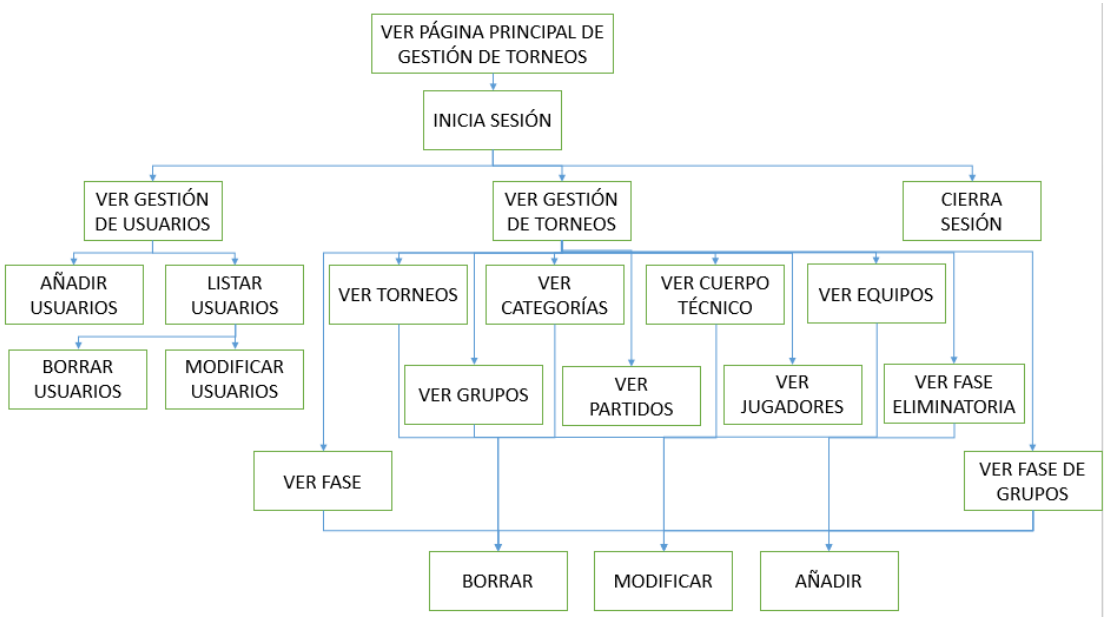


Figura 27: Diagrama de dependencias de la gestión web

Las dependencias de la aplicación Android se muestran en la figura 28:

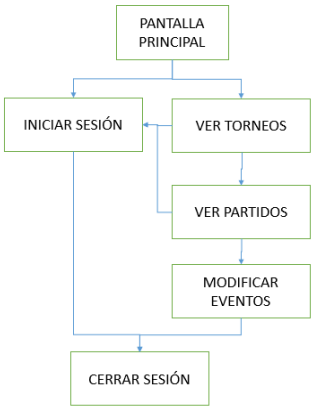


Figura 28: Diagrama de dependencias de la aplicación Android

4.3.3.- Diagrama de carril

A continuación se muestra en los siguientes subapartados los distintos diagramas de carril representando las distintas actividades de la aplicación Android y la aplicación web.

4.3.3.1.- Aplicación Android

En las figuras 29, 30, 31, 32 y 33 se muestran los diagramas de la aplicación Android.

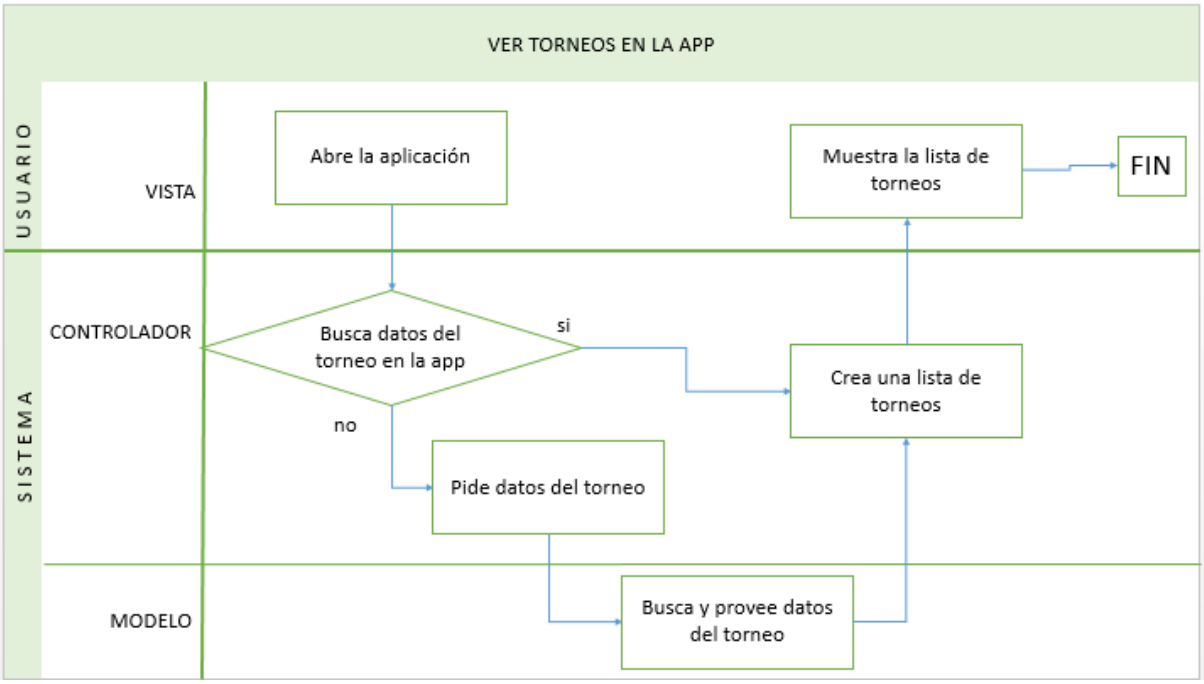


Figura 29: Diagrama de carril para ver torneos en la aplicación

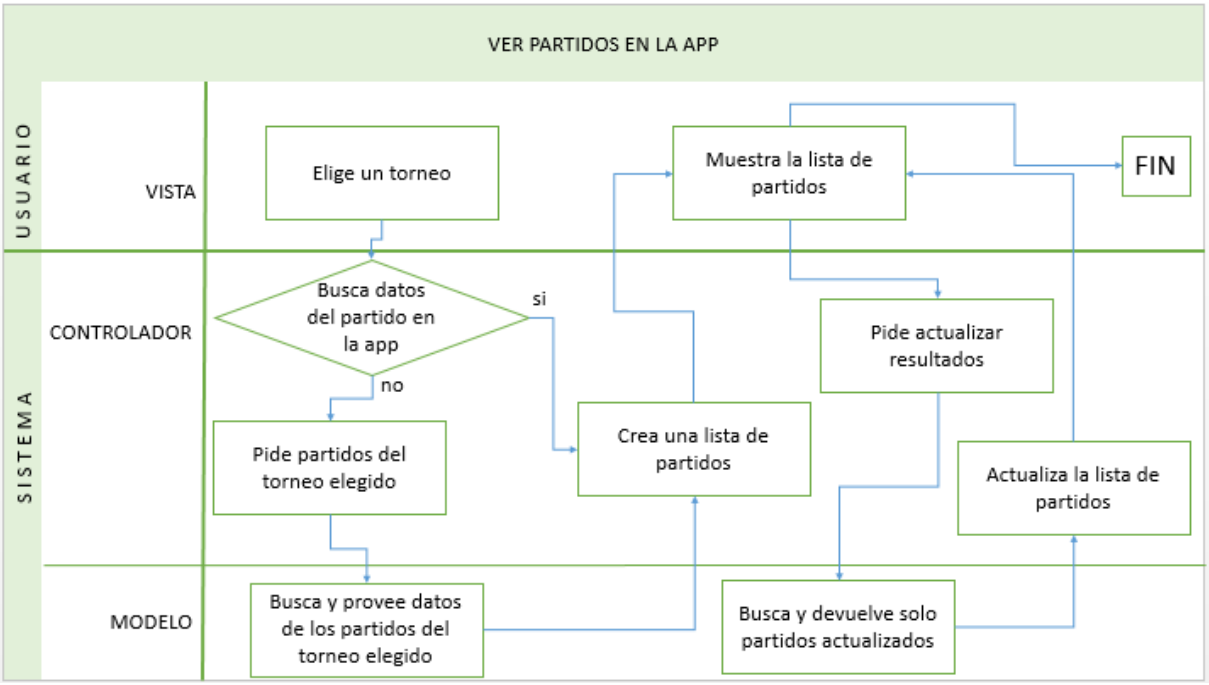


Figura 30: Diagrama de carril para ver partidos en la aplicación

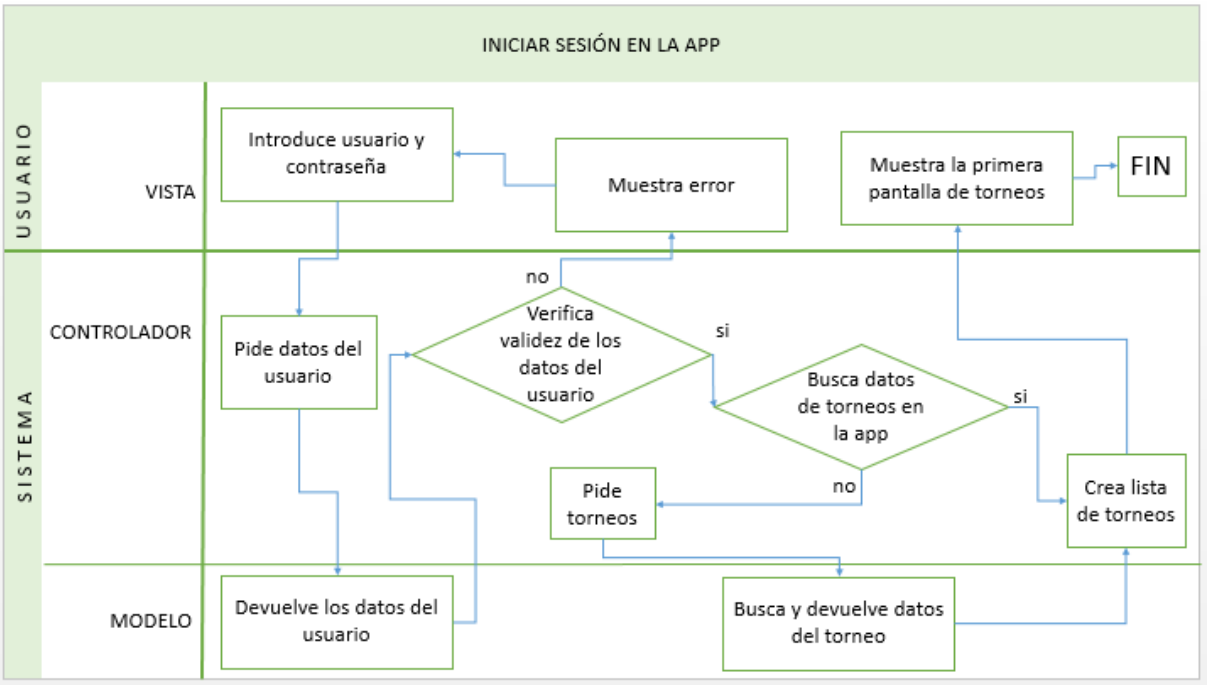


Figura 31: Diagrama de carril para el inicio de sesión en la aplicación

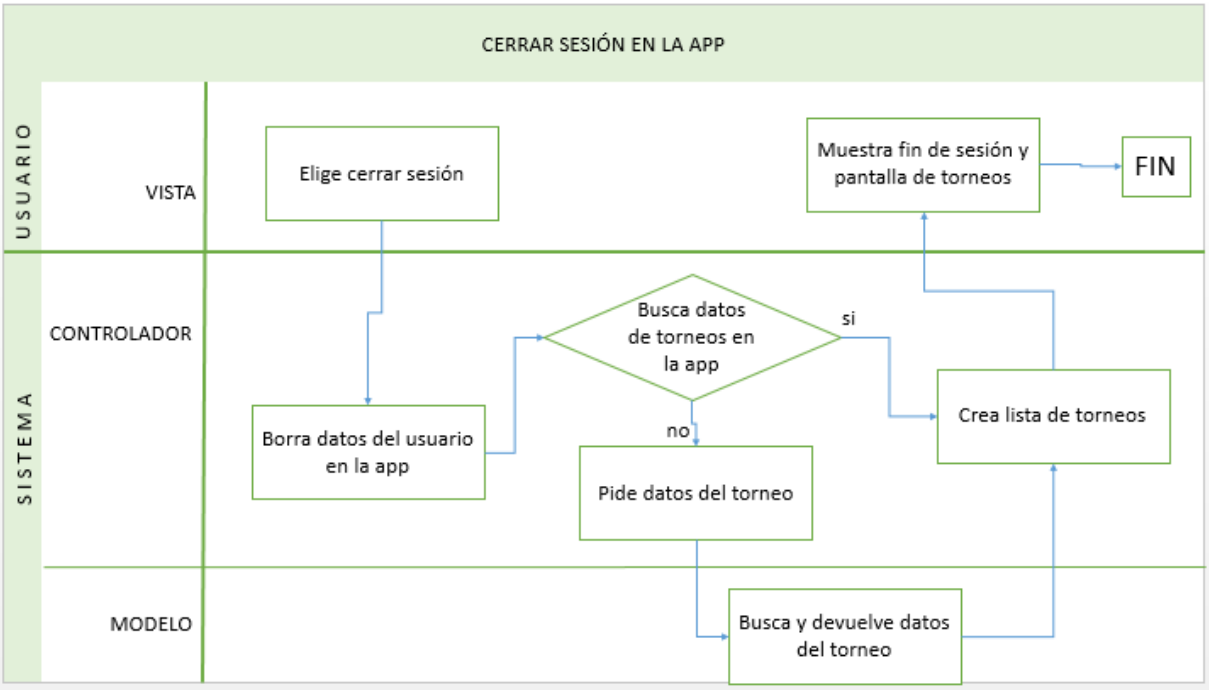


Figura 32: Diagrama de carril para cerrar sesión en la aplicación

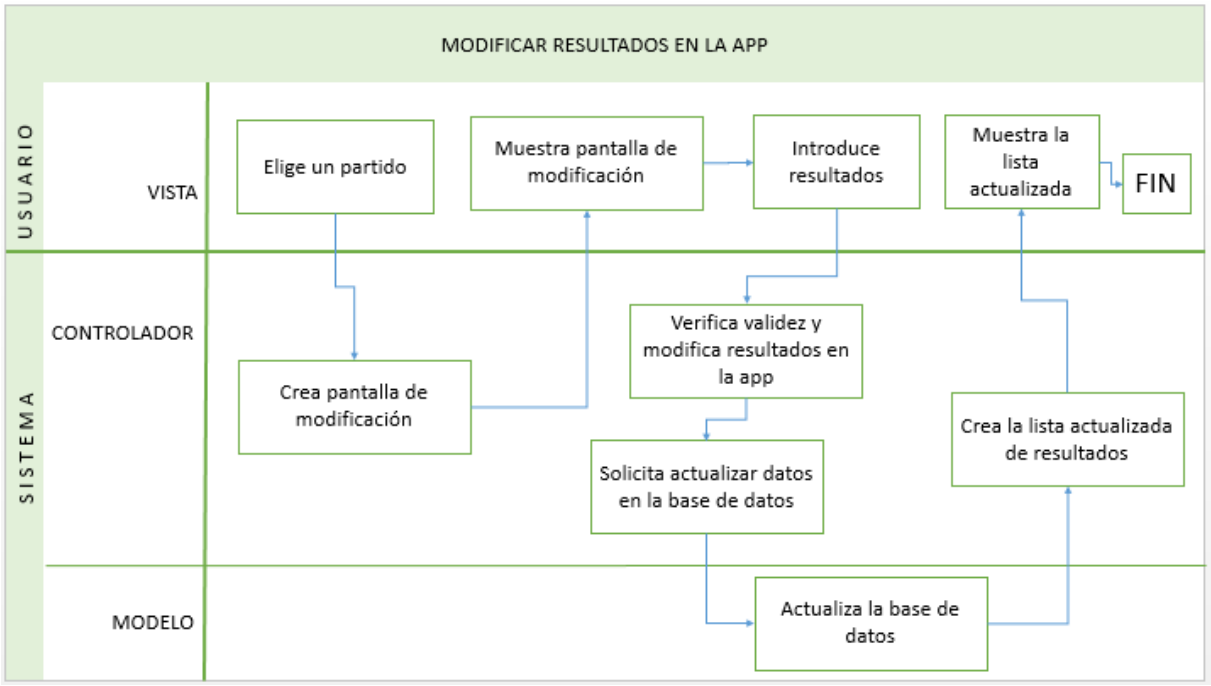


Figura 33: Diagrama de carril para modificar resultados en la aplicación

4.3.3.2.- Aplicación Web

En las figuras 34, 35, 36, 37 y 38 se muestran los diagramas de la aplicación web.

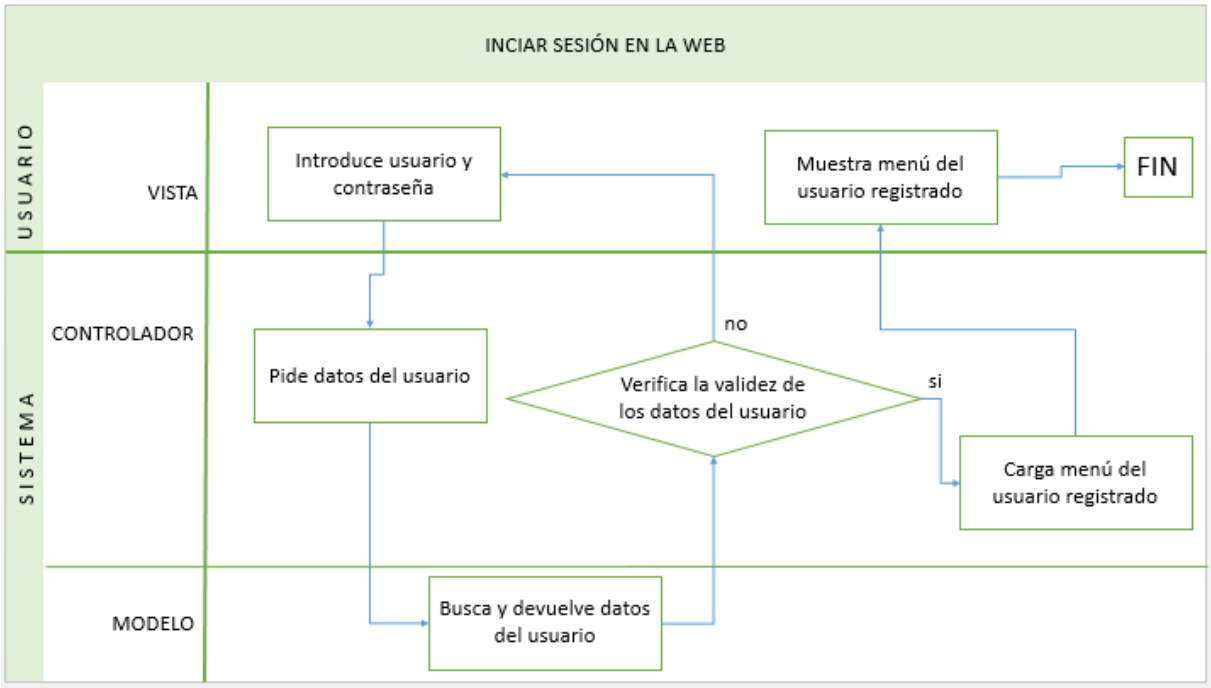


Figura 34: Diagrama de carril para iniciar sesión en la web

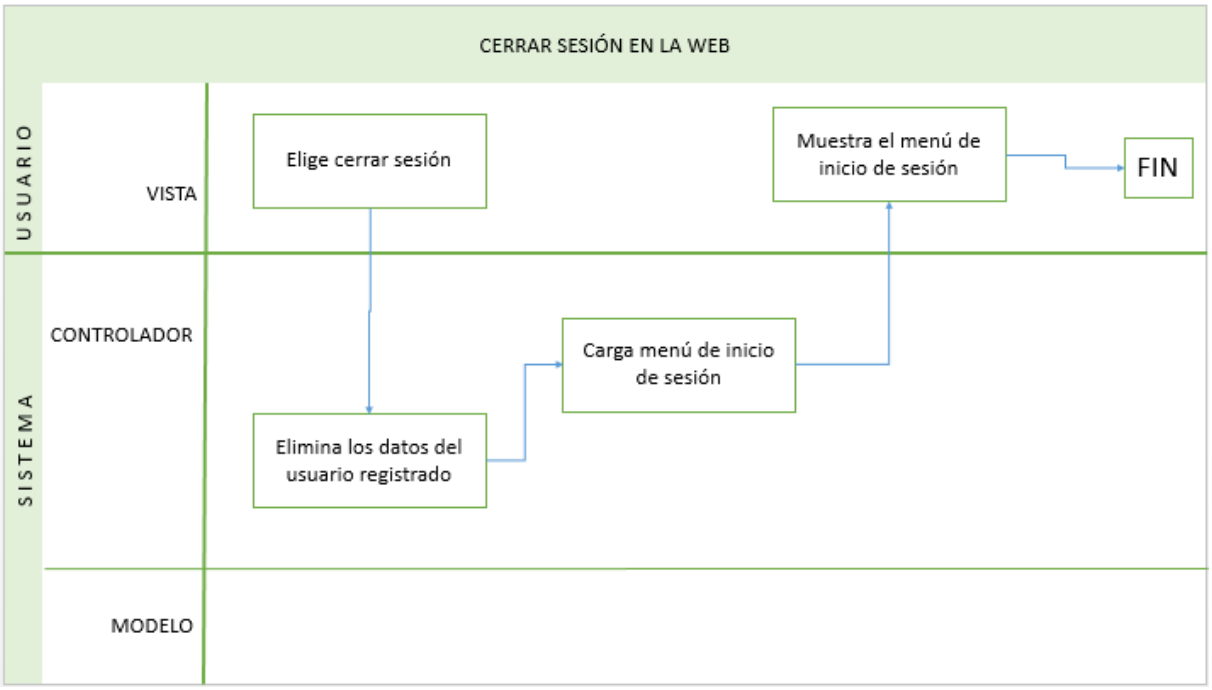


Figura 35: Diagrama de carril para cerrar sesión en la web

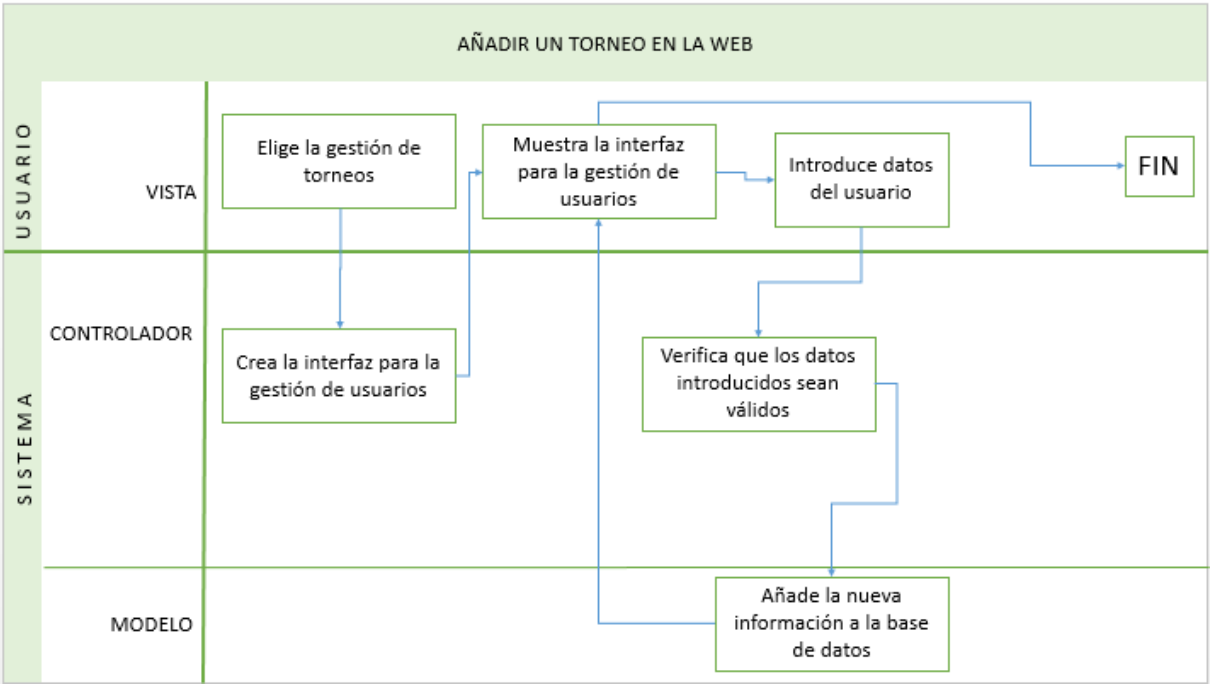


Figura 36: Diagrama de carril para añadir un torneo en la web

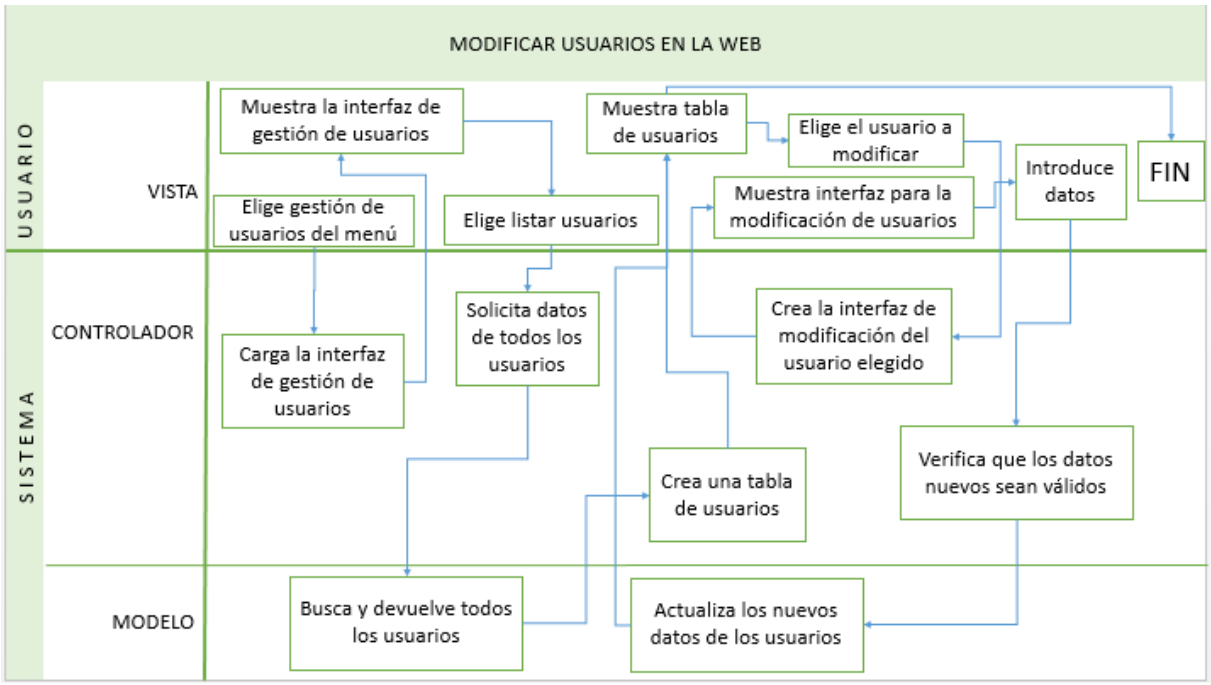


Figura 37: Diagrama de carril para modificar usuarios en la web

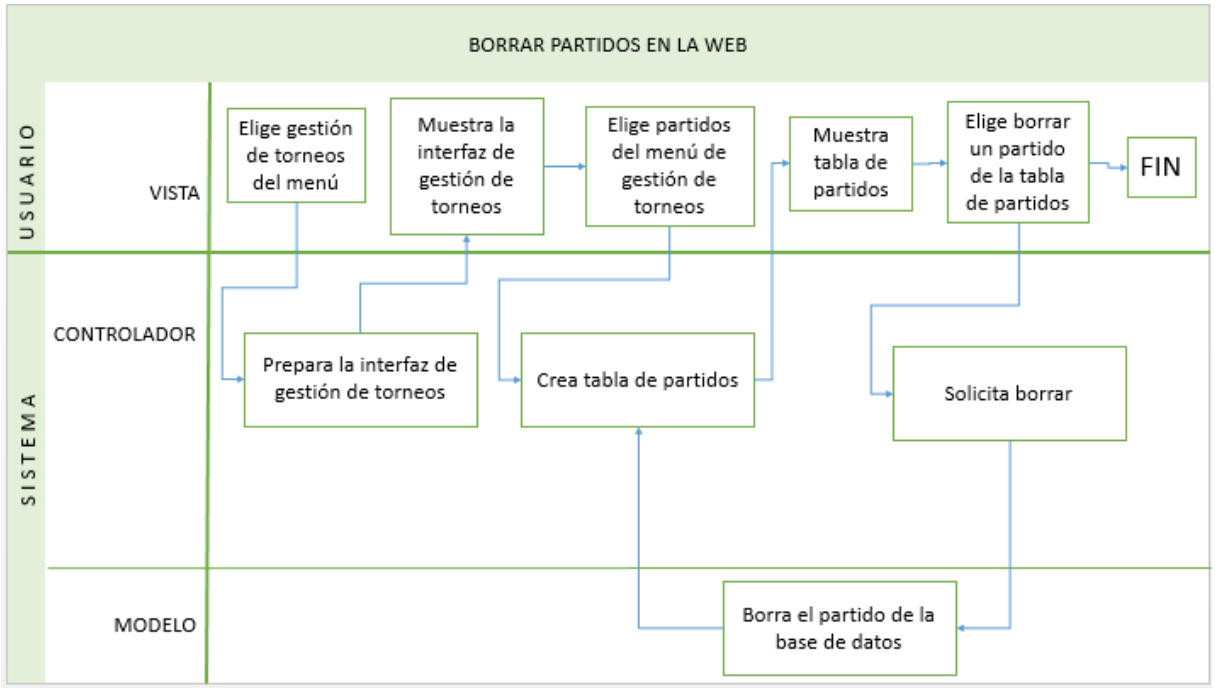


Figura 38: Diagrama de carril para borrar partidos en la web

4.3.4.- Diagrama de clases de la aplicación Android

La figuras 39, 40 y 41 representan a las clases con sus variables y funciones más importantes de la aplicación Android.

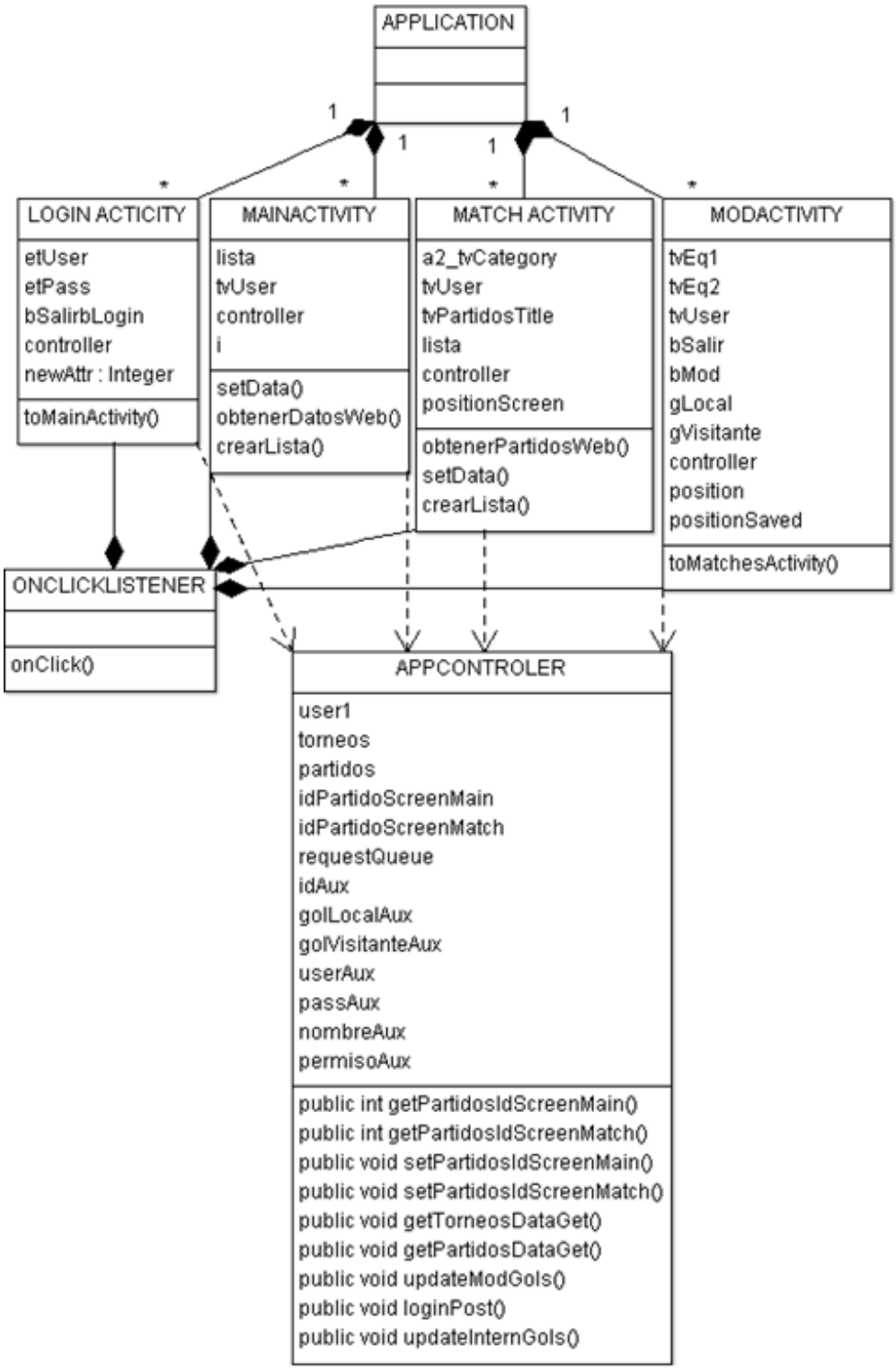


Figura 39: Diagrama de clases de la aplicación parte 1

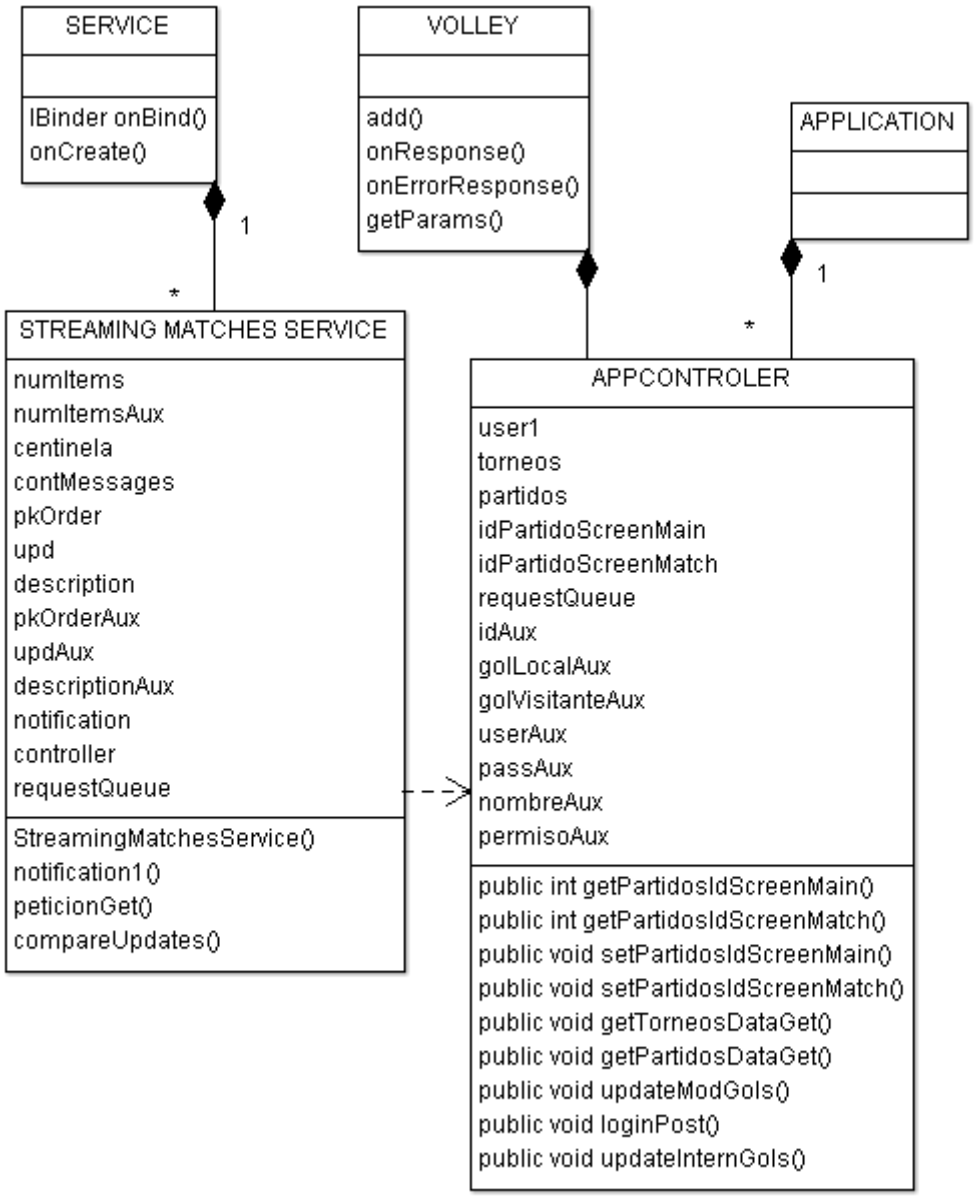


Figura 40: Diagrama de clases de la aplicación parte 2

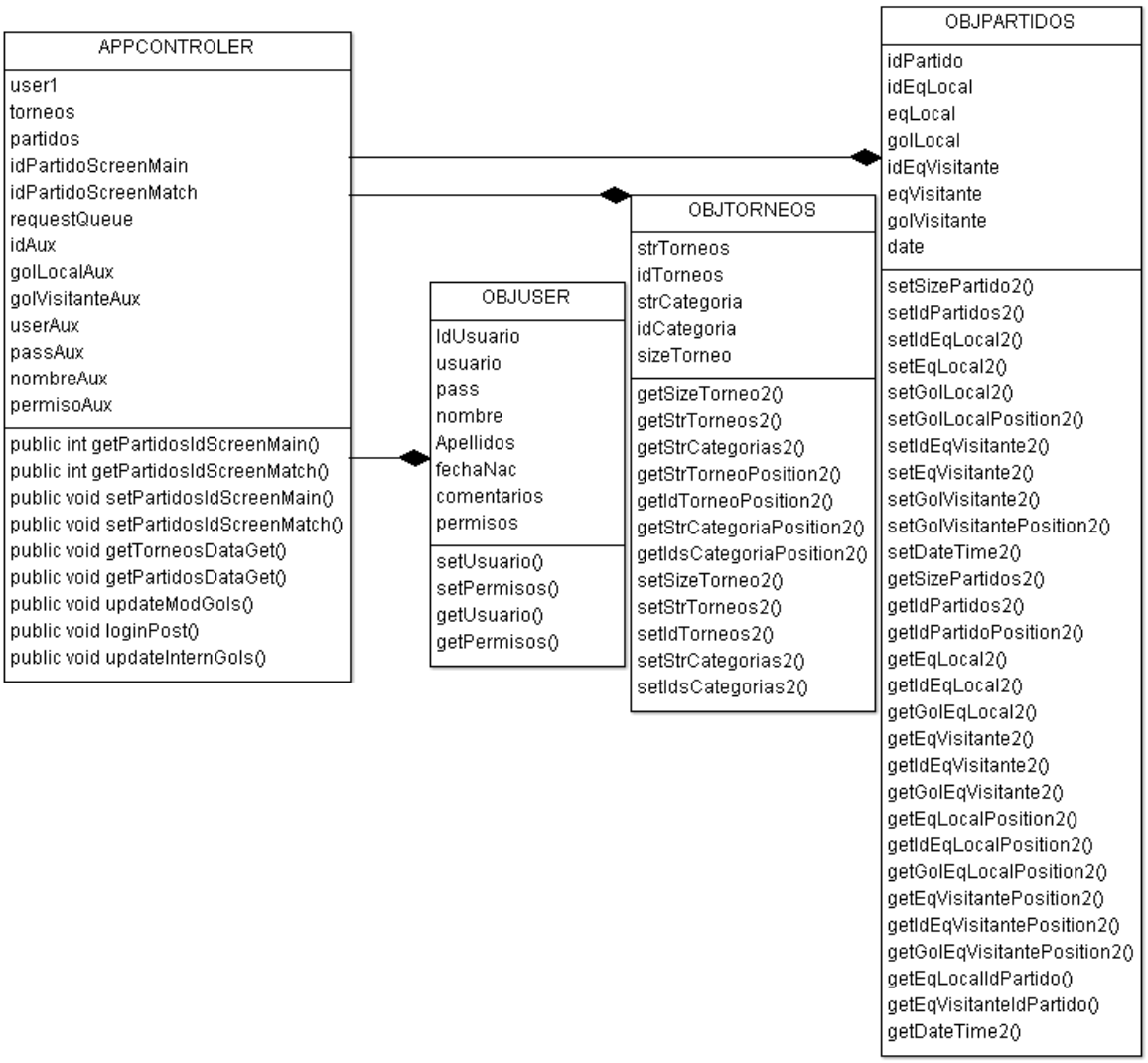


Figura 41: Diagrama de clases de la aplicación parte 3

4.3.5.- Diagrama Entidad Relación

En este apartado se mostrará el diseño de las dos bases de datos, una para la gestión de torneos y otra para la gestión de usuarios, mostrando el diseño de ambas bases de datos mediante el diagrama Entidad/Relación.

Luego se muestra la imágen que representa la implementación de todas las tablas y sus componentes creados en base al diagrama Entidad/Relación de cada una de las bases de datos junto a la descripción de cada una de sus tablas.

Finalmente se explicará el funcionamiento del trigger implementado para gestionar la introducción de nuevos eventos de la base de datos.

4.3.5.1.- Base de datos para torneos

La figura 42 representa el diagrama Entidad/Relación del diseño de la base de datos de la gestión de torneos.

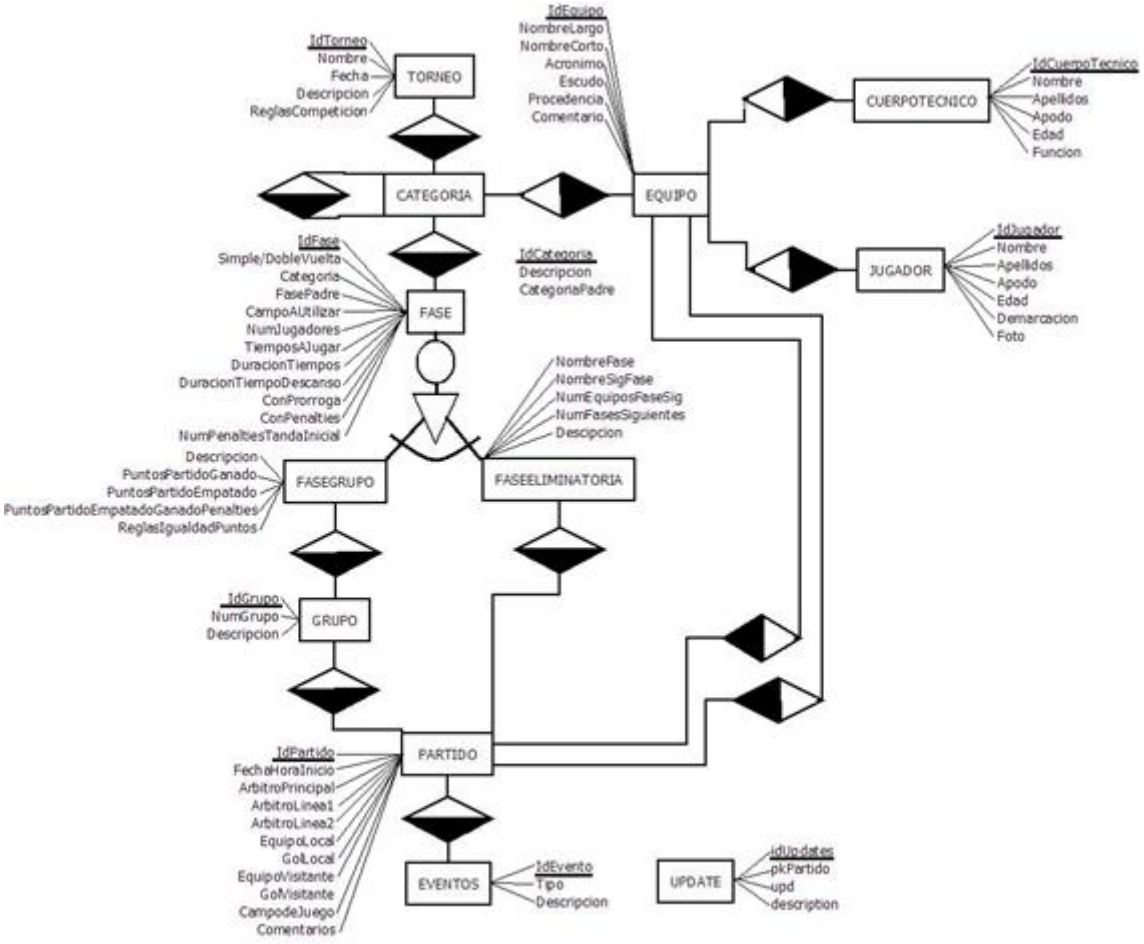


Figura 42: Diagrama Entidad/Relación de la gestión de torneos

En la figura 43 se puede observar la implementación de las distintas tablas de la gestión de torneos creadas en base al diagrama Entidad/Relación.

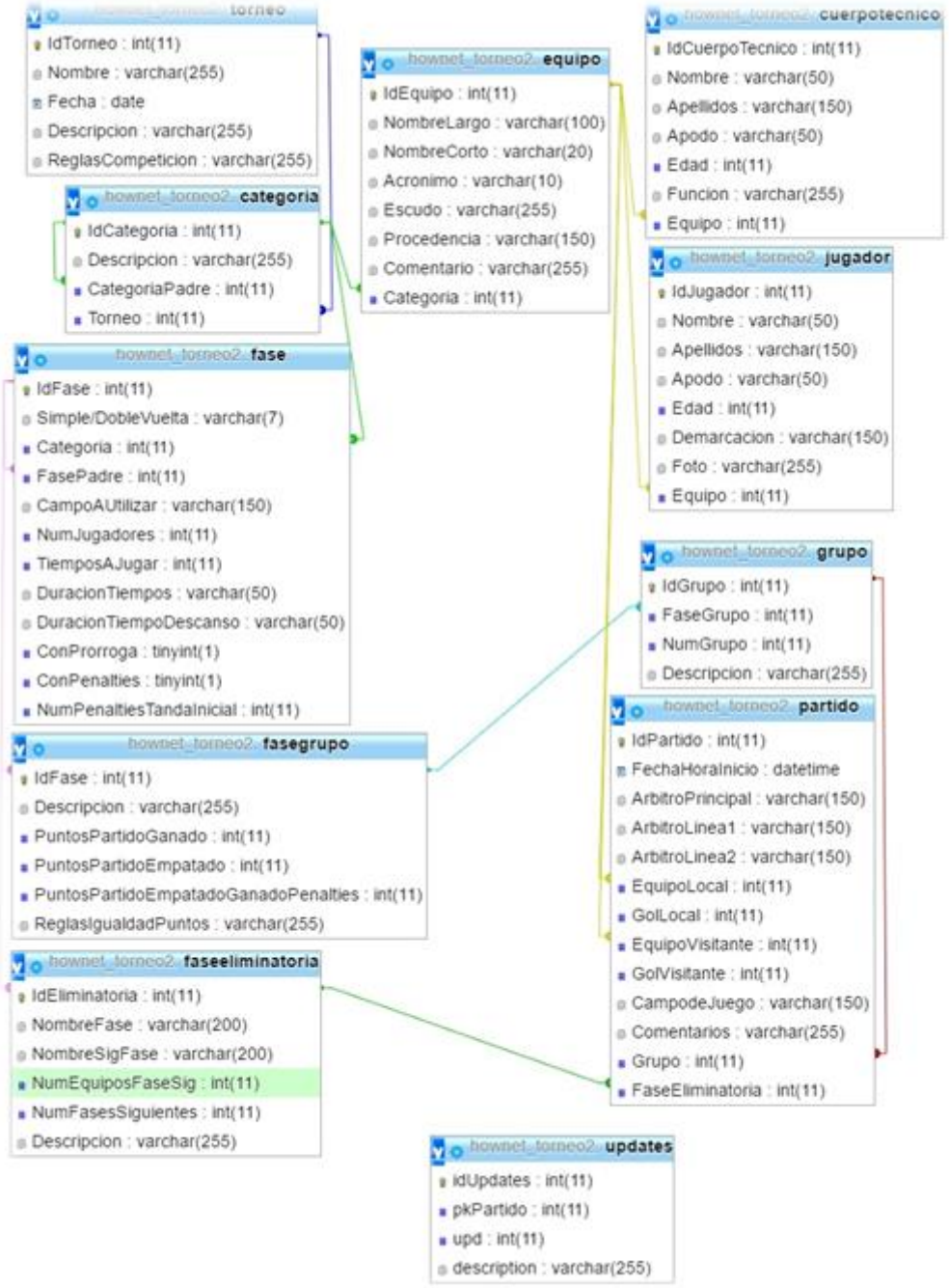


Figura 43: Tablas de la base de datos de la gestión de torneos

Descripción de las tablas implementadas:

- Tabla Torneos: en la que se puede crear torneos y sus características.
- Tabla Categorías: se puede crear distintas categorías bajo distintos nombres y asociarlos a un torneo específico, pudiendo incluso anidar categorías.
- Tabla Equipos: cada equipo se asocia a una categoría y la categoría está asociada a un torneo.
- Tabla Cuerpo Técnico: se añade datos del personal o cuerpo técnico de los equipos, cada una de estas personas se asocia a un equipo.

- Tabla Jugador: cada jugador se asocia a un equipo específico.
- Tabla de Fases: se da de alta las fases que puede tener un torneo, es una generalización total y disjunta. Generalización porque tendrá características comunes entre las entidades de fase grupo y fase eliminatoria. Es total porque todos los datos de la fase deben ser parte de las entidades de la fase de grupo y la fase eliminatoria y es disjunta porque la fase de grupos y la fase eliminatoria no pueden tener las mismas ocurrencias.
- Tabla Fase Grupos: se da de alta la fase de grupos con sus características.
- Tabla Grupos: se considera que un torneo puede tener una sola fase de grupos o tener dos grupos paralelos pertenecientes a una misma fase de grupos.
- Tabla Fase Eliminatoria: se da de alta la fase eliminatoria con sus correspondientes características.
- Tabla Partidos: para dar de alta partidos, se debe saber que cada partido está asociado una fase eliminatoria o una fase de grupos y está asociado tanto al equipo local como al equipo visitante.
- Tabla Update: está destinada a registrar los eventos actualizados que ocurren durante los partidos, un trigger se encarga de añadir datos esta tabla, siempre que se actualicen los eventos de un partido.

4.3.5.2.- Base de datos para usuarios

La figura 44 representa el diagrama Entidad/Relación del diseño de la base de datos de la gestión de usuarios.

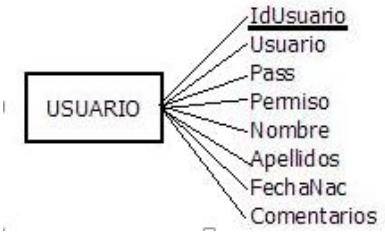


Figura 44: Diagrama Entidad/Relación de la gestión de usuarios

En la figura 45 se puede observar la implementación de la tabla de la gestión de usuarios creada en base al diagrama Entidad/Relación.

Figura 45: Tabla para la gestión de torneos

Descripción de la tabla creada:

- Tabla Usuario: dedicada a la gestión de usuarios, no dispone de ningún trigger, sin embargo se está utilizando la función PASSWORD() para codificar la

contraseña en la base de datos. Además el control de usuarios repetidos y datos obligatorios están siendo gestionados desde los programas Php.

4.3.5.3.- Trigger

Se tiene el inconveniente de que el hosting contratado no da permisos para poder crear funciones, procedimientos o triggers desde phpMyadmin, sin embargo se puede crear triggers mediante código SQL y poder utilizarlos. Funciones SQL se crean pero no se pueden utilizar por cuestiones de permisos. Los triggers por lo tanto no pueden contener funciones o procedimientos.

```
DELIMITER //
CREATE TRIGGER `updateEvent` AFTER UPDATE ON `partido`
 FOR EACH ROW BEGIN


   DECLARE cantidad_ocasiones INT;
 1 DECLARE eqLocal VARCHAR(100);
   DECLARE eqVisitante VARCHAR(100);

   SELECT NombreLargo INTO eqLocal
 2 FROM equipo,partido
   WHERE idEquipo=EquipoLocal and idPartido=new.idPartido;

   SELECT NombreLargo INTO eqVisitante
 3 FROM equipo,partido
   WHERE idEquipo=EquipoVisitante and idPartido=new.idPartido;



   SELECT COUNT(pkPartido) INTO cantidad_ocasiones
 4 FROM updates
   WHERE pkPartido = new.IdPartido;

   IF ( cantidad_ocasiones IS NULL) THEN
 5  SET cantidad_ocasiones = 0;
   END IF;

 6 SET @increment =cantidad_ocasiones;

   SET @increment=@increment+1;
   SET @eventoText1=(concat('GOL-',new.GolLocal));
   SET @eventoText2=(select(concat(@eventotext1,'-')));
   SET @eventoText3=(select(concat(@eventotext2,new.GolVisitante)));
 7 SET @eventoText4=(select(concat(@eventoText3,'-')));
   SET @eventoText5=(select(concat(@eventoText4,eqLocal)));
   SET @eventoText6=(select(concat(@eventoText5,'-')));
   SET @eventoText7=(select(concat(@eventoText6,eqVisitante)));

 8 INSERT INTO updates (pkPartido,upd,description) VALUES (new.IdPartido,@increment,@eventoText7);

END
//
DELIMITER ;
```

Figura 46: Trigger para la registrar un evento

En la figura 46 se puede observar que el trigger se llama “updateEvent”, se dispara después de actualizar un registro de la tabla partidos, sus características son:

- En el párrafo 1 se declaran las variables: “cantidad_ocaciones” (número de eventos), “eqLocal” (nombre equipo local) y “eqVisitante” (nombre equipo visitante).
- En el párrafo 2 y 3 se averigua el nombre largo del equipo local y visitante.
- En el párrafo 4 se averigua si ya existe algún evento asociado a algún partido.
- En el párrafo 5, si no existe ningún evento asociado a un partido específico (NULL), el contador inicia “cantidad_ocaciones” igual a 0.

- En el párrafo 6, “cantidad_ocaciones” se incrementa en 1 para indicar que se añadirá un nuevo evento.
- En el párrafo 7 se concatenan las características del evento, es importante indicar que este trigger está preparado para actuar junto el evento de actualización de goles de los equipos local y visitante, por lo tanto la concatenación busca el siguiente formato “Evento-Suceso_Equipo1-Suceso_Equipo2-Nombre_Equipo1-Nombre_Equipo2”.
- Finalmente, en el párrafo 8 se insertan las variables en la tabla “updates” según el ejemplo de la figura 46, la columna “pkPartido” será la id del partido, “upd” será 5 (número de eventos del partido) y “description” será “GOL-0-0-Equipo1-Equipo2”.

Cabe recalcar que la tabla “updates” se va a decodificar adecuadamente en la aplicación Android.

4.4.- Implementación del prototipo

La implementación del prototipo para las aplicaciones web y Android, estará basado en el modelo de capas Modelo Vista Controlador. En ambas aplicaciones la Vista únicamente muestra en pantalla la interfaz, el Modelo se encarga de proporcionar los datos y el Controlador gestiona todas las funciones de la aplicación.

4.4.1.- Aplicación Web

Se ha creado un único archivo "index.php" para gestionar todas las actividades de la web en una sóla página Html como se ve en la figura 47.

```
<?php
?>
<html lang="es">
<head>
    <title>ADMINISTRADOR WEB</title>
    <meta charset="UTF-8"/>
    <script type="text/javascript" src="javascript/interface.js"></script>
    <link rel="stylesheet" href="css/estilo.css">
</head>
<body onload="sessionInterface();">
<header id="cabecera">ADMINISTRACIÓN WEB DE TORNEOS</header>
<nav id="menu"></nav>
<main id="main"></main>
</body>
</html>
```

Figura 47: “Index.php”

Esta página carga el archivo JavaScript "interface.js" que cargará la Vista y el Controlador según la función que el usuario necesite como: "sessionInterface()" para el menú de inicio de sesión, "CargarGestUsuariosInterface()" para la gestión de usuarios y "CargarGestTorneosInterface()" para la gestión de torneos. La figura 48 muestra cómo se implementa la carga del menú para la función de inicio de sesión “sessionInterface()”.

```
function sessionInterface()
{
    // Cargar VISTA MENÚ
    try{REQ=new XMLHttpRequest();}
    catch(e){alert("No AJAX"); return;}
    REQ.open("POST","menu_v.php",false);
    REQ.setRequestHeader('Content-type','application/x-www-form-urlencoded');
    REQ.send("");
    $("menu").innerHTML=REQ.responseText;

    // Cargar CONTROLADOR MENÚ
    try{REQ2=new XMLHttpRequest();}
    catch(e){alert("No AJAX"); return;}
    REQ2.open("POST","menu_c.php",false);
    REQ2.setRequestHeader('Content-type','application/x-www-form-urlencoded');
    REQ2.send("");
    eval(REQ2.responseText);


}
function CargarGestUsuariosInterface(){
function CargarGestTorneosInterface(){
```

Figura 48: "Interface.js"

El menú se cargará sobre la etiqueta "<nav id='menu'></nav>" de "index.php" y sobre la etiqueta "<main id='main'></main>" se cargará todo lo necesario para poder gestionar las actividades (mostrar, modificar, añadir y borrar) en la gestión de usuarios y torneos.

4.4.1.1.- Implementación de la gestión del menú

La funcionalidad del menú implica crear tres archivos: un archivo será para la Vista, otro para el Modelo de datos y otro para el Controlador.

4.4.1.1.1.- La Vista

En el archivo "menu_v.php" se verificará la sesión para ver si el usuario con permisos web ha iniciado sesión o no.

```
<?php
session_start();

if(!isset($_SESSION["admin"]) || $_SESSION["admin"]!="admin")
{
    echo "<h1>INICIO SESIÓN</h1>"
        //."Id Usuario:<br/>"
        ."<input type='text' name='user' id='user' placeholder='USUARIO'/><br/>"
        //."Contraseña:<br/>"
        ."<input type='password' name='pwd' id='pwd' placeholder='CONTRASEÑA'/><br/><br/>"
        ."<button id='BotonAcceder'>Acceder</button>";
    exit();
}
?>

<h1>MENÚ</h1>
<button id='BotonGestUsuarios'>Gestión de usuarios</button><br/>
<button id='BotonGestTorneos'>Gestión de torneos</button><br/><br/>
<button id='BotonSalir'>Salir</button><br/>
```

Figura 49: "Menu_v.php"

Entonces, se creará la Vista del menú como se ve en la figura 49, según las siguientes características:

- Si no es administrador, se mostrará en la Vista un menú para que el usuario pueda iniciar sesión.
- Si es administrador, se mostrará un menú que permita elegir la gestión de usuarios, la gestión de torneos o cerrar sesión.

4.4.1.1.2.- El Modelo

El Modelo de datos del menú gestionará en el archivo "menu_m.php", los datos y las comprobaciones necesarias para iniciar sesión.

```php
<?php
session_start();
header("Content-Type: text/html; charset=iso-8859-1");

switch($_POST["opc"])
{
    case "ENTRAR":
        $u=$_POST["u"];
        $p=$_POST["p"];

        include("db_config_user.php");
        ////verifying user and password
        $sentenciaSQL="SELECT Usuario,Pass,Permiso FROM Usuario WHERE Usuario='$u' AND Pass=PASSWORD('$p')";
        $result = mysql_query($sentenciaSQL) or die(mysql_error());
        if (mysql_num_rows($result) > 0) {
        else{
        //verifying permiso
        if(strcmp($permisoUsu,"app")==0){
        else if(strcmp($permisoUsu,"web")==0){
        mysql_close($conn);
        break;


    case "SALIR":
        $_SESSION["admin"]="";
        $_SESSION = array();
        session_destroy();
        break;

    default:
        echo "menu - 'default'";
        break;
}
?>
```

Figura 50: "Menu_m.php"

Como se ve en la figura 50, se recibirá una petición para el inicio de sesión y cierre de sesión, en esa petición se le enviará una variable "opc" que podrá ser "ENTRAR" o "SALIR":

- La operación "ENTRAR", realizará las operaciones necesarias para verificar si la información del usuario es válido, devolverá dos mensajes. El primer mensaje es "True" y su valor es "ok" para indicar que todo ha ido bien. El segundo mensaje es "False" y su valor podrá ser "Usuario y/o contraseña incorrectos" o "No tiene suficientes permisos para acceder".
- La operación "SALIR", borrará todos los datos de la sesión del administrador sin devolver nada al usuario.

4.4.1.1.3.- El Controlador

Se gestionará desde el archivo "menu_c.php", se verificará si el usuario que acceda a este archivo ha iniciado sesión como usuario administrador web, como se ve en la figura 51.

```
if(!isset($_SESSION["admin"]) || $_SESSION["admin"]!="admin")
{
?>
/*Actualiza vista*/
$("main").innerHTML="<h1>BIENVENIDO</h1><p>Web para la gestión

$("BotonAcceder").onclick=function()
{
    var u=$("user").value;
    var p=$("pwd").value;

    if(!u || u.length<1)
    {
    if(!p || p.length<1)
    {

    try{REQ=new XMLHttpRequest();}
    catch(e){alert("No AJAX"); return;}
    REQ.open("POST","menu_m.php",false);
    REQ.setRequestHeader('Content-type','application/x-www-form
    REQ.send("opc=ENTRAR&u="+u+"&p="+p);

    try{REQ=eval(REQ.responseText);}
    catch(e){alert("Error:\n\n" + REQ.responseText); return;}

    if(!REQ[0])
        alert(REQ[1]);
    else
        sessionInterface();

    /*Actualiza vista*/
    $("main").innerHTML="<h1>BIENVENIDO</h1><p>Web para la gest
```

Figura 51: "Menu_c.php" para usuario que no inicia sesión

Si no se ha iniciado sesión, se dará la orden de actualizar la Vista para crear la interfaz de inicio de sesión. También se cargará el evento para el botón "BotonAcceder" que enviará el usuario y la contraseña al Modelo "menu_m.php" para verificar si es correcto o no. El Modelo devuelve datos de la operación de inicio de sesión y el resultado se mostrará sobre un mensaje alert para indicar si el inicio de sesión ha ido correcto o no.

```
//actualiza vista para usuarios que hayan inicia
$("main").innerHTML="<h1>BIENVENIDO, SE HA IDENT

//botones
$("BotonGestUsuarios").onclick=function()
{
    $("main").innerHTML="";
    CargarGestUsuariosInterface();
}

$("BotonGestTorneos").onclick=function()
{
    //carga Vista y controlador de gestión de to
    $("main").innerHTML="";
    CargarGestTorneosInterface();
}

$("BotonSalir").onclick=function()
{
    try{REQ=new XMLHttpRequest();}
    catch(e){alert("No AJAX"); return;}
    REQ.open("POST","menu_m.php",false);
    REQ.setRequestHeader('Content-type','applica
    REQ.send("opc=SALIR");
    //carga nuevamente la interfaz de inicio de
    sessionInterface();
}
```

Figura 52: "Menu_c.php" para usuario que inicia sesión

Como se ve en la figura 52, si se ha iniciado sesión como usuario administrador, se dará la orden de actualizar la Vista mostrando y dando la funcionalidad correspondiente a los botones para la gestión de usuarios, torneos y salir ("BotonGestUsuarios","BotonGestTorneos" y "BotonSalir").

4.4.1.2.- Implementación de la gestión de usuarios

Para dar toda la funcionalidad necesaria a la gestión de usuarios, se crearán 3 archivos, uno para la Vista, otro para el Modelo de datos y otro para el Controlador.

```php
<?php
session_start();

if(!isset($_SESSION["admin"]) || $_SESSION["admin"]!="admin")
{
    echo "<p>Acceso restringido, debe iniciar sesión</p>";
    exit();
}
?>
```

Figura 53: Comprobación de usuario registrado en la web

Como se puede apreciar en la figura 53, si no se ha iniciado sesión un mensaje tipo "echo" muestra al usuario que el acceso es restringido e inmediatamente se sale de la página.

4.4.1.2.1.- La Vista

Desde el archivo "usuario_v.php" se cargará en pantalla la Vista con los componentes para la gestión de usuarios en pantalla. Como se ve en la figura 54, se prepara un formulario para poder añadir usuarios con los permisos pertinentes. A continuación se mostrará un botón que permitirá mostrar la lista con todas las características de todos los usuarios de la base de datos.

```html
<h1>GESTIÓN DE USUARIOS</h1>
<h2>Añadir usuario</h2>
<div id="tablaUsers">
<!--<form method="POST" action="usuario_m.php">-->
<fieldset>
    <legend>Usuario y Contraseña:</legend>
    Usuario*:<input type="text" id="Usuario" name="Usuario" required><br/>
    Contraseña*:<input type="password" id="Password" name="Pass"required><br/>
</fieldset>
<fieldset>
    <legend>Datos del usuario:</legend>
    Permiso*:
    <select id="Permiso" name="Permiso">
        <option value="app" selected>Gestión de la aplicación Android</option>
        <option value="web">Gestión web</option>
    </select><br/>
        Nombre*:<input type="text" id="Nombre" name="Nombre" required><br/>
        Apellidos*:<input type="text" id="Apellidos" name="Apellidos" required><br/>
        FechaNac*:<input type="date" id="FechaNac" name="FechaNac" required><br/>
        Comentarios:<input type="text" id="Comentarios" name="Comentarios"><br/>
</fieldset>
<!--<input type="hidden" name="opc" value="addUser"/>-->
<p>Los campos marcados con * son obligatorios.</p>
<button type="submit" id="buttonAddUser">AÑADIR USUARIO</button><br/><br/>
<!--</form>-->
</div>
<h2>Lista de Usuarios</h2>
<button id="ActualizarVistaUsuarios">MOSTRAR LISTA DE USUARIOS</button>
<div id="ListaUsuarios">
```

Figura 54: "Usuario_v.php"

4.4.1.2.2.- El Modelo

El Modelo está en el archivo "usuario_m.php" que recibirá peticiones desde el Controlador y devolverá información en formato Json. La variable "opc" puede tener los siguientes valores:

- "showUser" devolverá los datos de todos los usuarios con todas sus características, como se ve en la figura 55.

```php
switch($_POST["opc"])
{
    case "showUser":
        include("db_config_mysqli.php");
        include("mysql2json.php");
        echo get_Data_Table($tb_cds, "", array(), $mysqli);
        break;
```

Figura 55: "Usuario_m.php" petición "showUser"

- "addUser" recibirá las variables necesarias para poder añadir el nuevo usuario. Se verificará que todos los campos obligatorios existan caso contrario se devolverá un mensaje "false" indicando "Debe llenar los campos obligatorios marcados con *". Se conectará a la base de datos, creará la sentencia necesaria para añadir usuarios y se desconectará de la base de datos, ejemplo ilustrado en la figura 56.

```
case "addUser":

    include("db_config_user.php");
    //$action=$_POST['action'];
      //ADD parameters
      $usuario=$_POST['Usuario'];
      $pass=$_POST['Pass'];
      $permiso=$_POST['Permiso'];
      $nombre=$_POST['Nombre'];
      $apellidos=$_POST['Apellidos'];
      $fechaNac=$_POST['FechaNac'];
      if($usuario=="" || $pass=="" || $nombre=="" || $apellidos=="" || $fechaNac==""){
      else{

     mysql_close($conn);

    break;
```

Figura 56: "Usuario_m.php" petición "modUser"

- "modUserShow" se conecta a la base de datos para poder mostrar un usuario específico, siendo obligatorio recibir el id desde el Controlador, como se ve en la figura 57.

```
case "modUserShow":
    include("db_config_mysqli.php");
    include("mysql2json.php");
    $sentencia="SELECT * FROM Usuario WHERE IdUsuario=".$_POST['id'];
    echo get_Data_SQL($sentencia, array(), $mysqli);
    mysql_close($conn);

    break;
```

Figura 57: "Usuario_m.php" petición "modUserShow"

- "modUser" se conectará a la base de datos para poder modificar las características de los usuarios de las aplicaciones, es obligatorio recibir las variables que representan las características que se van a modificar. Se realizarán comprobaciones para recibir al menos los datos obligatorios desde el Controlador, como se ve en la figura 58. Se pasará la sentencia necesaria para actualizar los datos del usuario elegido. Se devolverá "falso" en caso de que existan problemas. Se devuelve "true" en caso de haber realizado la operación correctamente.

```
case "modUser":
    include("db_config_user.php");
      //ADD parameters
      $id=$_POST['id'];
      $usuario=$_POST['Usuario'];
      $pass=$_POST['Pass'];
      $permiso=$_POST['Permiso'];
      $nombre=$_POST['Nombre'];
      $apellidos=$_POST['Apellidos'];
      $fechaNac=$_POST['FechaNac'];
      if($usuario=="" || $nombre=="" || $apellidos=="" || $fechaNac==""){
          echo '([false, "Debe llenar los campos obligatorios marcados con *"])';
      }
      else{
     mysql_close($conn);
    break;
```

Figura 58: "Usuario_m.php" petición "modUser"

- "delUser" recibirá desde el Controlador la "id" del usuario a eliminar, se conecta con la base de datos y elimina el usuario, devuelve "True" con el valor "Se ha eliminado el usuario Correctamente", ejemplo ilustrado en la figura 59.

```php
case "delUser":
    include("db_config_user.php");
    $idUsuario=$_POST['id'];
    //getting values
    $sentenciaSQL="DELETE FROM Usuario WHERE IdUsuario=$idUsuario";
    $result = mysql_query($sentenciaSQL) or die(mysql_error());
    echo '([true, "Se ha eliminado el usuario Correctamente"])';
    mysql_close($conn);
    break;
```

Figura 59: "Usuario_m.php" petición "delUser"

4.4.1.2.3.- El Controlador

Desde el archivo "usuario_c.php" se ordena actualizar la Vista según las opciones elegidas, mostrando las opciones necesarias para poder gestionar usuarios.

Como se ve en la figura 60, se cargarán todos los eventos "onclick" para: "buttonAddUser", "ActualizarVistaUsuarios", "ModUser", "botonModificar" y "BorUser".

```javascript
//EVENTO PARA AÑADIR UN NUEVO USUARIO
$("buttonAddUser").onclick=function(){
//evento para actualizar vista usuarios
$("ActualizarVistaUsuarios").onclick=function()
{
    try{REQ=new XMLHttpRequest();}
    catch(e){alert("No AJAX"); return;}
    REQ.open("POST","usuario_m.php",false);
    REQ.setRequestHeader('Content-type','application/x-www-form-urlencoded');
    REQ.send("opc=showUser");

    try{resp = eval(REQ.responseText);}
    catch(e){$("ListaUsuarios").innerHTML = "<b><u>Error</u>:</b><br/><br/><pr

    //cabecera de formulario
    var cad="";
    cad+="<table>";
    //datos auxiliares para la captura del id de una fila
    cad+="<input  id='idUser' style='display:none'  type='text' />";
    cad+="<button id='ModUser' style='display:none'>MOD</button>";
    cad+="<button id='BorUser' style='display:none'>BOR</button>";
    //cabecera de tabla
    cad+="<tr><th>USUARIO</th><th>PERMISO</th><th>NOMBRE</th><th>APELLIDOS</th
    //a?adiendo datos de la base de datos en la tabla
    for(i=0;i<resp.length;i++)
    {
    //texto oculto con el id del usuario
    cad+="<input type='hidden' id='ID_Usuario'>";
    cad+="</table>";

    $("ListaUsuarios").innerHTML = cad;

    //MODIFICAR USUARIO, MOSTRAR DATOS DE USUARIO A MODIFICAR
    $('ModUser').onclick=function()
    {

    //BORRAR USUARIO
    $('BorUser').onclick=function()
    {
```

Figura 60: "Usuario_c.php"

4.4.1.3.- Implementación de la gestión de torneos

Para dar la funcionalidad necesaria para la gestión de torneos, se van a crear tres archivos, uno para la Vista, otro para el Modelo de datos y otro para el Controlador. Como se aprecia en la figura 61, si el usuario no tiene permisos un mensaje tipo "echo" indica que el sitio es de acceso restringido.

```php
<?php
session_start();

if(!isset($_SESSION["admin"]) || $_SESSION["admin"]!="admin")
{
    echo "<p>Acceso restringido, debe iniciar sesión</p>";
    exit();
}
?>
```

Figura 61: Controlando usuario registrado para la gestión de torneos

4.4.1.3.1.- La Vista

Mostrará todas las opciones necesarias para gestionar los torneos, las categorías, las fases, la fase eliminatoria, la fase de grupo, grupos, partidos, equipos, cuerpo técnico, jugadores desde el archivo "torneos_v.php".

```
<h1>GESTIÓN DE TORNEOS</h1>
<p>Eliga una opción:</p><hr/>
<button id="torneo">TORNEOS</button>
<button id="categoria">CATEGORÍAS</button>
<button id="fase">FASES</button>
<button id="faseEliminatoria">FASE ELIMINATORIA</button>
<button id="faseGrupo">FASE DE GRUPOS</button>
<button id="grupo">GRUPOS</button><br/>
<button id="partido">PARTIDOS</button>
<button id="equipo">EQUIPOS</button>
<button id="cuerpoTecnico">CUERPO TÉCNICO</button>
<button id="jugador">JUGADORES</button><br/><hr/>
<div id="contenidoTorneos">
</div>
```

Figura 62: "Torneos_v.php" de usuario registrado

4.4.1.3.2.- El Modelo

Desde el archivo "torneos_m.php" se gestionará el Modelo de datos. Para listar tablas con todas las características de los componentes del torneo: "showTorneos", "showCategorias", "showCuerpoTecnico", "showEquipo", "showFase", "showFaseEliminatoria", "showFaseGrupo", "showGrupo", "showJugador" y "showPartido", como se ve en la figura 63.

```
switch($_POST["opc"])
{
/*SHOW TABLES*/
    case "showTorneos":
        /*include("db_config_torneos_mysqli.php");
    case "showCategorias":
        /*include("db_config_torneos_mysqli.php");
    case "showCuerpoTecnico":
        /*include("db_config_torneos_mysqli.php");
    case "showEquipo":
        /*include("db_config_torneos_mysqli.php");
    case "showFase":
        /*include("db_config_torneos_mysqli.php");
    case "showFaseEliminatoria":
        /*include("db_config_torneos_mysqli.php");
    case "showFaseGrupo":
        /*include("db_config_torneos_mysqli.php");
    case "showGrupo":
        /*include("db_config_torneos_mysqli.php");
    case "showJugador":
        /*include("db_config_torneos_mysqli.php");
    case "showPartido":
        /*include("db_config_torneos_mysqli.php");
```

Figura 63: "Torneo_m.php" y con sus peticiónes

Se puede borrar datos con las órdenes: "delTorneo", "delCategorias", "delCuerpoTecnico", "delEquipo", "delFase", "delFaseEliminatoria", "delFaseGrupo", "delGrupo", "delJugador" y "delPartido". Se puede modificar datos, en este caso se listará sólo el dato a modificar. Finalmente se puede añadir datos en cada una de las categorías de la gestión de torneos.

4.4.1.3.3.- El Controlador

El archivo "torneos_c.php" controla la Vista y el Modelo de datos de la gestión de torneos. También añade todos los eventos "onClick" y actualiza la Vista mostrando un listado de opciones.

Se cargan eventos "onclick" para los torneos, el cuerpo técnico, los equipos, fases, fases eliminatorias, fases de grupos, grupos, jugadores y partidos (figura 64). Cada uno de esos

elementos tendrá eventos asociados a botones para poder actualizar la Vista, borrar datos y modificar los datos del elemento elegido.

```
$("torneo").onclick=function()
{
$("categoria").onclick=function()
{
$("fase").onclick=function()
{
$("faseEliminatoria").onclick=function()
{
$("faseGrupo").onclick=function()
{
$("grupo").onclick=function()
{
$("partido").onclick=function()
{
$("equipo").onclick=function()
{
$("cuerpoTecnico").onclick=function()
{
$("jugador").onclick=function()
{
```

Figura 64: "Torneos_c.php" con sus eventos onclick

4.4.2.- Aplicación Android

Al igual que la aplicación web, se va a seguir el patrón de diseño del modelo de capas Modelo Vista Controlador. Se irá explicando el desarrollo de cada capa del modelo indicando en todo momento dónde y cómo se integra con los componentes de la aplicación Android.

4.4.2.1.- Implementando la Vista

La Vista serán todos los elementos de la Activity que se muestren en pantalla, a continuación se expondrán las características de cada una de esas clases con sus componentes visuales:

- "MainActivity.java" es la pantalla principal y está formada por una ListView para listar los torneos, un TextView para conocer la identidad del usuario que ha iniciado sesión, un menú contextual y las notificaciones Toast. La figura 65 muestra su implementación.

```
public class MainActivity extends AppCompatActivity {
    /*VARIABLES*/
    //listview
    ListView lista;
    //textView
    TextView tvUser;
    //Appcontroller
    AppController controller;
    //variable intent, for describe service's things, depending on the message got it
    Intent i;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        /*VARIABLES LOCALES*/
        //listview
        lista=(ListView)findViewById(R.id.main_listView);
        //textView
        tvUser=(TextView)findViewById(R.id.main_tvUser);
        // Calling AppController (see application tag in AndroidManifest.xml)
        controller=(AppController) getApplicationContext();
        //variable for swiperefreshlayout
        final SwipeRefreshLayout swipeView = (SwipeRefreshLayout) findViewById(R.id.swipe);
```

Figura 65: Vista de la MainActivity

- "match4Activity.java" se compone de tres TextViews, en el que se mostrará información sobre el usuario que ha iniciado sesión, la categoría en que se esté

desarrollando la partida y el título de la Activity. También tiene un ListView, para ver la lista actualizada de los resultados de los partidos que se estén disputando en ese momento. Su implementación se puede ver en la figura 66.

```
public class match4Activity extends AppCompatActivity {
    /*VARIABLES*/
    //textView
    TextView a2_tvCategory,tvUser,tvPartidosTitle;
    //listView
    ListView lista;
    //for AppController
    AppController controller;
    //for variables got it from main Activity
    int positionScreen;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_match4);
        /*VARIABLES LOCALES*/
        //textView
        a2_tvCategory=(TextView)findViewById(R.id.a2_tvCategory);
        tvPartidosTitle=(TextView)findViewById(R.id.a2_tvPartidosTitle);
        tvUser=(TextView)findViewById(R.id.a2_tvUser);
        //listView
        lista=(ListView)findViewById(R.id.a2_lvMatches);
        // Calling AppController (see application tag in AndroidManifest.xml)
        controller=(AppController) getApplicationContext();
```

Figura 66: Vista de la match4Activity

- "modActivity.java" tiene los elementos visuales para modificar el resultado de la partida. Está compuesta por tres TextViews para indicar el nombre de los equipos y el nombre del usuario, dos botones uno para poder salir y otro para hacer efectivo la modificación de los resultados del partido, dos EditTexts para poder modificar los los goles de los equipos, como se ve en la figura 67.

```
public class modActivity extends AppCompatActivity implements View.OnClickListener{
    /*VARIABLES*/
    //for textView
    TextView tvEq1,tvEq2,tvUser;
    //buttons
    Button bSalir,bMod;
    //editText
    EditText gLocal,gVisitante;
    //for AppController
    AppController controller;
    //for variables got it from main Activity
    int position,positionSaved;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mod);
        /*VARIABLES LOCALES*/
        //textView
        tvEq1=(TextView)findViewById(R.id.am_tvTitleEq1);
        tvEq2=(TextView)findViewById(R.id.am_tvTitleEq2);
        tvUser=(TextView)findViewById(R.id.ma_tvUser);
        //editText
        gLocal=(EditText)findViewById(R.id.ma_etGolEq1);
        gVisitante=(EditText)findViewById(R.id.ma_etGolEq2);
        //buttons
        bMod=(Button)findViewById(R.id.ma_botonModificar);
        bMod.setOnClickListener(this);
        bSalir=(Button)findViewById(R.id.ma_botonSalir);
        bSalir.setOnClickListener(this);
```

Figura 67: Vista de la modActivity

- "LoginActivity.java" está compuesto por dos EditTexts para introducir el nombre y la contraseña del usuario, también tiene dos botones y una barra de progreso para poder mostrar el estado del proceso de inicio de sesión, se puede ver en la figura 68 su implementación.

```
public class LoginActivity extends AppCompatActivity implements View.OnClickListener{
    /*VARIABLES*/
    public  EditText etUser,etPass;
    Button bSalir,bLogin;
    ProgressBar pb;
    TextView tvPB;
    //Appcontroller
    AppController controller;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        /*VARIABLES LOCALES*/
        // Calling AppController (see application tag in AndroidManifest.xml)
        controller=(AppController) getApplicationContext();
        //EditText
        etUser=(EditText)findViewById(R.id.la_etUser);
        etPass=(EditText)findViewById(R.id.la_etPass);
        //progressbar
        pb=(ProgressBar) findViewById(R.id.progressBar);
        //TextViewProgressBar
        tvPB=(TextView) findViewById(R.id.textViewPB);
        //Button
        bLogin=(Button)findViewById(R.id.la_bLogin);
        bLogin.setOnClickListener(this);
        bSalir=(Button)findViewById(R.id.la_bSalir);
        bSalir.setOnClickListener(this);
```

Figura 68: Vista de LoginActivity

4.4.2.2.- Implementando el Modelo

El Modelo de datos son todos aquellos archivos Php alojados en el servidor, que permiten la comunicación con la base de datos para obtener toda la información que la aplicación le solicite, los archivos Php son: "db_config.php", "db_config_user.php", "login.php", "partidos.php", "seeUpdate.php", "torneo.php" y "db_config.php"

4.4.2.2.1.- Archivo Db_config.php

Este archivo se conecta con el servidor para acceder a la base de datos de los torneos, la figura 69 muestra su implementación.

```
<?php
$mysql_hostname = "tfg01.hownet.es";
$mysql_user = "...";
$mysql_password = "...";
$mysql_database = "hownet_torneo2";
$db = mysql_connect($mysql_hostname, $mysql_user, $
mysql_select_db($mysql_database, $db) or die("Opps
?>
```

Figura 69: “Db_config.php” para conectar con la base de datos de los torneos

4.4.2.2.2.- Archivo Db_config_user.php

Este archivo se conecta con el servidor para acceder a la base de datos de los usuarios, la figura 70 muestra su implementación.

```
<?php
$mysql_hostname = "tfg01.hownet.es";
$mysql_user = "...";
$mysql_password = "...";
$mysql_database = "hownet_user";
$db = mysql_connect($mysql_hostname, $mysql_user, $
mysql_select_db($mysql_database, $db) or die("Opps
?>
```

Figura 70: “DB_config_user.php” para conectar con la base de datos de usuarios

4.4.2.2.3.- Archivo Login.php

Como se ve en la figura 71, mediante Post se recibe el nombre del usuario y la contraseña, se verifica que esos datos sean válidos. Se ha utilizado Post para asegurar los datos introducidos por el usuario no sean vistos en texto plano de manera pública.

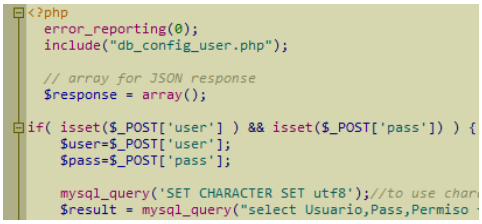


Figura 71: Modelo "login.php"

Si los datos existen se devuelve el texto "[NombreUsuario]#[Permiso]", indicando el nombre del usuario y los permisos que tiene, como se ve en la figura 72.

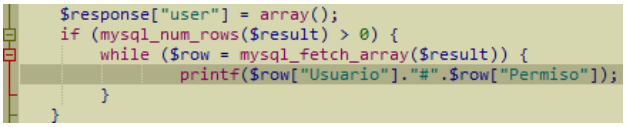


Figura 72: "Login.php" devolviendo el usuario con su permiso

Si los datos no son correctos, se devuelve el texto "-1#-1" para indicar que no existe un usuario con los datos proporcionados ni existen permisos válidos, como se ve en la figura 73.

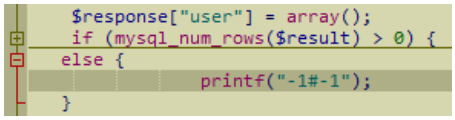


Figura 73: "Login.php" devolviendo usuario no válido

4.4.2.2.4.- Archivo Partidos.php

Se recibe el valor de la categoría por Get, ya que esos datos son públicos. Se verifica que el dato recibido sea válido, se conecta con la base de datos para recibir información de los partidos de la categoría elegida.

El primer valor del array se llama "partido" y es otro array compuesto por: el id del partido, la fecha de inicio del partido, nombre del equipo local, id del equipo local, goles locales, nombre del equipo visitante, el id del equipo visitante, el gol del equipo visitante, el nombre de la fase de grupo y el nombre de la fase eliminatoria. En caso de que algún dato no exista, se pone "NN" para evitar que existan nulos, se puede ver un ejemplo de la obtención del nombre del equipo en la figura 74.

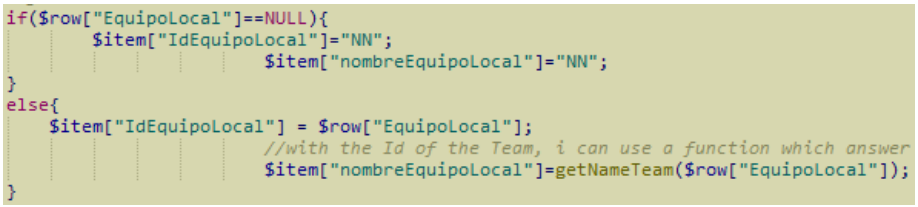


Figura 74: "Partidos.php" devolviendo el nombre de los equipos

El segundo valor del array se llama "success" y será "1" si la consulta se ha realizado correctamente, será "0" si existe algún fallo, además si existe algún fallo se envía otro valor llamado "mensaje" cuyo contenido es "No matches Found", como se ve en la figura 75.

```php
if (mysql_num_rows($result) > 0) {

    $response["partido"] = array();
    while ($row = mysql_fetch_array($result)) {
      // success
     $response["success"] = 1;
}
else {
    // order is empty
      $response["success"] = 0;
      $response["message"] = "No matches Found";
```

Figura 75: "Partidos.php" devolviendo una consulta fallida

4.4.2.2.5.- Archivo seeUpdate.php

No se pasa ninguna variable, como se ve en la figura 76, este archivo obtiene datos de los eventos más recientes de los partidos en la tabla "updates" en la base de datos.

```php
<?php
error_reporting(0);
include("db_config.php");

// array for JSON response
$response = array();

// get all items from myorder table
$result = mysql_query("
SELECT pkPartido,max(upd) as upd
FROM updates
GROUP BY pkPartido
") or die(mysql_error());

if (mysql_num_rows($result) > 0) {

    $response["updates"] = array();

    while ($row = mysql_fetch_array($result)) {
            // temp user array
            $item = array();
            $item["pkPartido"] = $row["pkPartido"];
            $item["upd"] = $row["upd"];
            $item["description"] = lookDescription($row["upd"]);

            // push ordered items into response array
            array_push($response["updates"], $item);
           }
     // success
     $response["success"] = 1;
}
else {
    // order is empty
      $response["success"] = 0;
      $response["message"] = "No Items Found";
}
// echoing JSON response
echo json_encode($response);

/*funciones*/
function lookDescription($updaux){
?>
```

Figura 76: "SeeUpdate.php"

Si existen valores, se crea y compone un array con dos valores:

- El primer valor se llama "updates" y es también un array compuesto por la clave primaria del partido, el token que indica el número máximo de la cantidad de veces que se han actualizado los resultados del partido y la descripción del evento que ha generado la actualización, obviamente se obtiene y devuelve siempre la más reciente actualización del partido (max(upd)).
- Cuando no existe ningún problema y se realiza la obtención de datos correctamente, el segundo valor del array principal llamado "success" será "0".

- En caso de que ocurra cualquier problema y no se realice correctamente la obtención de datos, el valor llamado "success" será "1", además se envía otro valor llamado "mensaje" cuyo contenido es "No matches Found".

4.4.2.2.6.- Archivo Torneo.php

Como se ve en la figura 77, se conecta con la base de datos para verificar cuantos torneos existen, se compone un array con dos valores:

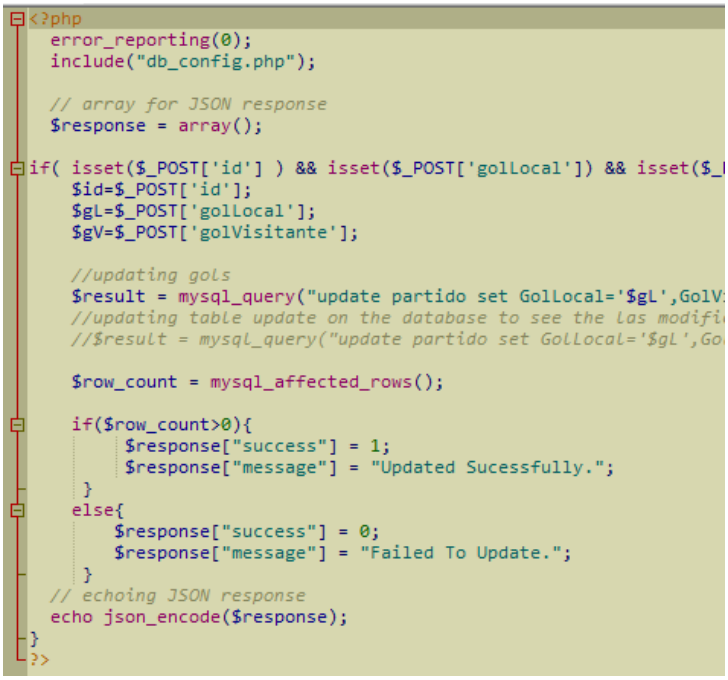
- El primer valor es un array llamado "torneo" que es otro array compuesto por el id del torneo, el nombre del torneo, la id de la categoría y el nombre de la categoría.
- El segundo valor se llama "success" y será "1" si la consulta se ha realizado correctamente, "0" si existe algún fallo y si existe algún fallo se envía otro valor llamado "mensaje" cuyo contenido es "No tournaments Found".

```php
<?php
error_reporting(0);
include("db_config.php");

// array for JSON response
$response = array();

// get all items from torneo table
mysql_query('SET CHARACTER SET utf8');//to use characters u
$result = mysql_query("SELECT Torneo,Nombre,IdCategoria,cate

if (mysql_num_rows($result) > 0) {

    //saving infor of tournaments on response
    $response["torneo"] = array();

    //getting the info of tournaments
    while ($row = mysql_fetch_array($result)) {

            //getting torneo data
            $item = array();
            $item["idTorneo"] = $row["Torneo"];
            $item["nombreTorneo"] = $row["Nombre"];
            $item["idCategoria"] = $row["IdCategoria"];
            $item["nombreCategoria"] = $row["Descripcion"];
            // push to add information on torneo of response
            array_push($response["torneo"], $item);
    }

    // success
    $response["success"] = 1;
}
else {
    //if there is no data of tournaments or category
      $response["success"] = 0;
      $response["message"] = "No tournaments Found";
}
// echoing JSON response
echo json_encode($response);

?>
```

Figura 77: "Torneo.php"

4.4.2.2.7.- Archivo Update.php

La figura 78 muestra cómo se recibe el "id" del partido, "golLocal" y "golVisitante", son valores importantes para actualizar el resultado de un partido en la base de datos, por lo tanto se recibirán mediante Post.



```php
<?php
  error_reporting(0);
  include("db_config.php");

  // array for JSON response
  $response = array();

if( isset($_POST['id'] ) && isset($_POST['golLocal']) && isset($_
    $id=$_POST['id'];
    $gL=$_POST['golLocal'];
    $gV=$_POST['golVisitante'];

    //updating gols
    $result = mysql_query("update partido set GolLocal='$gL',GolV
    //updating table update on the database to see the las modifi
    //$result = mysql_query("update partido set GolLocal='$gL',Go

    $row_count = mysql_affected_rows();

    if($row_count>0){
        $response["success"] = 1;
        $response["message"] = "Updated Sucessfully.";
    }
    else{
        $response["success"] = 0;
        $response["message"] = "Failed To Update.";
    }
  // echoing JSON response
  echo json_encode($response);
}
?>
```

Figura 78: "Update.php"

Se comunica con la base de datos y con los valores recibidos se compone la query necesaria para actualizar el resultado de un partido. Se devuelve un array con dos valores dependiendo del resultado de la ejecución de la query:

- Si la operación de actualización se ha realizado correctamente, se devuelve el valor llamado "success" con "1" y otro valor llamado "message" con "Updated Sucessfully."
- Si la operación de actualización no se ha realizado correctamente, se decuelve el valor llamado "success" con "0" y otro valor llamado "message" con "Failed To Update."

4.4.2.3.- Implementando el Controlador

El Controlador de la aplicación Android está presente en varios sitios, se irá indicando dónde está el Controlador y qué tareas está realizando dentro de la aplicación Android.

4.4.2.3.1.- Archivo "AppController.java"

Esta clase representa a la clase controladora general (deriva de Application) y es donde se van a inicializar y gestionar los objetos ObjUser, ObjTorneos, ObjPartidos; también inicializa y gestiona la clase "RequestQueue" de la librería Volley, como se ve en la figura 79.

```
public class AppController extends Application {
/*DECLARACIÓN DE VARIABLES*/
    //for users
    ObjUser user1=new ObjUser();
    //for torneos
    ObjTorneos torneos=new ObjTorneos();
    //for matches
    ObjPartidos partidos=new ObjPartidos();
    int idPartidoScreenMain=-1,idPartidoScreenMatch=-1;
    //for Volley API
    RequestQueue requestQueue;
    //helpers for POST variables
    String idAux,golLocalAux,golVisitanteAux;
    String userAux,passAux,nombreAux,permisoAux;
```

Figura 79: "AppController.java" inicialización de variables

Como se ve en la figura 80, existen funciones para obtener y asignar los valores de los usuarios, torneos y partidos desde la aplicación.

```
/*FUNCIONES*/
//GET VARIABLES
    //for Torneos
    //for Partidos
    public int getPartidosIdScreenMain(){return idPartidoScreenMain;}
    public int getPartidosIdScreenMatch(){return idPartidoScreenMatch;}
//SET VARIABLES
    //for Torneos
    //for partidos as a control variable
    public void setPartidosIdScreenMain(int xxx){idPartidoScreenMain=xxx;}
    public void setPartidosIdScreenMatch(int xxx){idPartidoScreenMatch=xxx;}

//VOLLEY
    //for Torneos
    public void getTorneosDataGet(Context context,Response.Listener<JSONObject> listener,
                                  Response.ErrorListener errlsn) {
    //for Partidos
    public void getPartidosDataGet(String idCategoria,Context context,Response.Listener<JSONObject> listener,
                                   Response.ErrorListener errlsn) {
    //for update Gols
    public void updateModGols(String idAux1,String golLocalAux1,String golVisitanteAux1){
    //for login
    public void loginPost(String user, String pass){
//UPDATE FUNCTIONS
//update idPartido, gollocal and golvisitante from intern storage
    public void updateInternGols(String idPartidoAux,String golLocalAux,String golVisitanteAux){
}
```

Figura 80: "AppController.java" funciones

Además gracias a las funciones de la librería Volley se puede realizar peticiones web Post y Get, para poder:

- Obtener los torneos, pidiendo al Modelo de datos ("torneo.php") la información de los torneos.
- Obtener los partidos de un torneo elegido, pidiendo información al Modelo ("partidos.php") indicando la categoría del partido.
- Actualizar los goles que el usuario haya modificado desde la aplicación, indicando al Modelo de datos ("update.php") mediante Post junto a las variables id del partido, el gol del equipo local y el gol del equipo visitante.
- Realizar operaciones de inicio de sesión, enviando el nombre del usuario y la contraseña mediante Post al Modelo de datos ("login.php") para verificar si el usuario es el correcto y para ver qué permisos tiene.

También tiene la función "updateInternGols(...)" para actualizar los resultados de los partidos que han sufrido modificaciones en memoria local, de esta forma se evita eliminar todos los datos y cargarlos nuevamente para obtener los datos actualizados.

4.4.2.3.2.- Archivo "StreamingMatchesService.java"

Este es el servicio en segundo plano que tiene la aplicación, esta clase controladora se utiliza para poder verificar cada cierto tiempo si se han actualizado los resultados de los partidos en la base de datos.

Con ayuda de Volley cada 5000 milisegundos se realiza la verificación, se pregunta al Modelo ("seeupdate.php") el partido con los datos de los eventos más recientes en la base de datos, el Modelo verifica y solicita datos a la tabla “updates” de la base de datos, la figura 81 muestra su implementación.

```
//espera 5 segundos
final Handler h = new Handler();
final int delay = 5000; //milliseconds
//una vez pasa 5 segundos verifica datos nuevos en la base de datos
controller=(AppController) getApplicationContext();
//Toast.makeText(getApplicationContext(), "Service created", Toast.LENGT

h.postDelayed(new Runnable() {
    public void run() {
        //do something
        h.postDelayed(this, delay);
        //solicita una peticion para ver datos actualizados del servidor
        requestQueue = Volley.newRequestQueue(getApplicationContext());
        peticionGet();
    }
}, delay);
```

Figura 81: “StreamingMatchesServices.java” verificando cada 5 segundos

Como se ve en la figura 82, cuando se recibe la información del partido con los eventos más recientes, la función "compareUpdates(...)" verifica que los nuevos datos sean realmente nuevos. Si los datos son nuevos se actualiza el resultado del partido en la memoria interna y se hace una llamada a una notificación Android para alertar al usuario que se ha actualizado el resultado de la partida.

```
    public void notification1(int id,int iconId,String titulo,String contenido,String pkPartido,String gLocal,String gVisitante){
    //FUNCIÓN para realizar la petición Get al servidor
    //PETICION GET PARA VER SI SE HA ACTUALIZADO DATOS EN EL SERVIDOR
    public void peticionGet(){
    /*Función para comparar tabla de ventos*/
    public void compareUpdates(String pkOrderAux,String updAux,String descriptionAux){
}
```

Figura 82: “StreamingMatchesServices.java” funciones

De esta forma gracias al Modelo de datos y las operaciones del Controlador, sólo se obtiene el valor más reciente y no todos los datos de los partidos del torneo, ahorrando espacio en memoria, evitando el rápido consumo de la batería (ya que consume menos recursos) y evitando el elevado consumo de datos (Internet) en el dispositivo móvil y el servidor web.

4.4.2.3.3.- Las actividades controladoras de cada Activity

La Activity "MainActivity.java" primero verifica qué usuario está activo. Si existe un usuario se verifica que permisos tiene. Si el permiso obtenido es "1" se considera que tiene permisos para modificar y actualizar partidos en la aplicación, si es "2" indica que tiene permisos sólo para administrar la web de la gestión de torneos y si es "-1" indica que es un usuario público ya que no ha iniciado sesión, como se ve en la figura 83. Si es un usuario válido, la Vista escribirá el nombre del usuario en la aplicación.

```
// permisos=1 user manager of app to upgrade gols, permisos=2 user manager of web, permisos=-1 public user
if(!controller.user1.getUsuario().equals("-1") && controller.user1.getPermisos().equals("app")){
    tvUser.setText("USUARIO:"+controller.user1.getUsuario());
}
else{
    tvUser.setText("USUARIO:público");
}


//getting data and creating list
if(controller.torneos.getSizeTorneo2()<0){//si no existen datos los obtiene desde el servidor
    obtenerDatosWeb();
}
else{
    //Toast.makeText(MainActivity.this,"lista sin internet", Toast.LENGTH_LONG).show();
    crearLista(controller.torneos.getStrTorneos2(),controller.torneos.getStrCategorias2());//si ya los tien
}
```

Figura 83: Controlador de MainActivity

Luego obtiene información de los torneos desde el Modelo de datos para crear una lista, cada elemento de la lista está formado por el nombre y la categoría del torneo para luego mostrarlo en la Vista.

En caso de haber ya cargado los torneos previamente se verifica el tamaño del torneo, si es mayor a 0 lo único que hace es reutilizar esos datos ya cargados previamente para luego mostrarlo, sin la necesidad de gastar recursos al hacer una nueva solicitud de nuevos datos.

A la Activity "match4Activity.java" se puede llegar desde la pantalla de los torneos, modificación de resultados y también desde la notificación. Si se ha producido una actualización, se envía un mensaje en las zona de notificaciones del móvil con la información del resultado actualizado, cuando se pulsa sobre la notificación se redirige automáticamente a esta pantalla para ver los partidos actualizados.

```
//control of screens, to not lose data already created
if(controller.getPartidosIdScreenMatch()!=controller.getPartidosIdScreenMain()){
    //idScreenMatch will be equal to idScreenMain, if idScreenMatch have -1 value
    controller.setPartidosIdScreenMatch(controller.getPartidosIdScreenMain());
    //data is obtained from Internet
    positionScreen=controller.getPartidosIdScreenMatch();
    obtenerPartidosWeb();

}
else{
    //when idScreenMatch is iqual to idScreenMain, means that data is stored already in memory, so it must be used
    positionScreen=controller.getPartidosIdScreenMatch();
    crearLista(controller.partidos.eqLocal,controller.partidos.golLocal,controller.partidos.eqVisitante,
    controller.partidos.golVisitante,controller.partidos.getDateTime2());
}

//search if there is a user already logged or not
//and if this user has the necesary grants to loging,
// permisos=1 user manager of app to upgrade gols, permisos=2 user manager of web, permisos=-1 public user
if(!controller.user1.getUsuario().equals("-1") && controller.user1.getPermisos().equals("app")){
    tvUser.setText("USUARIO:"+controller.user1.getUsuario());
}
else{
    tvUser.setText("USUARIO:público");
}
//set a name to category
a2_tvCategory.setText("CATEGORIA:" + controller.torneos.getStrCategoriaPosition2(positionScreen));
//change the title to know in which torneo I am
tvPartidosTitle.setText("PARTIDOS:" + controller.torneos.getStrTorneoPosition2(positionScreen));

//inicio del servicio en background
startService(new Intent(this, StreamingMatchesService.class));
```

Figura 84: Controlador de match4Activity

Entonces, se debe controlar al menos si se viene desde las notificaciones, la figura 84 muestra su implementación. Si se viene desde la notificación, se debe traducir el mensaje que contiene los datos del partido actualizado para poder mostrar en pantalla mediante un Toast con la actualización producida por el nuevo resultado.

Luego, para cargar los datos de los partidos de un torneo desde el Modelo, se hace un

control para verificar que la pantalla de partidos no haya perdido los datos de los partidos creados del torneo elegido. Esto se produce principalmente cuando se ha actualizado el resultado de un partido en un torneo, se envía la notificación, mientras se puede ver otros torneos y si se pulsa en la notificación se debe poder volver a la pantalla correspondiente al torneo del resultado actualizado.

Para controlar esa situación la clase "AppController" añade un identificador a la pantalla principal y a la pantalla actual de partidos. El identificador de la pantalla principal de partidos representa la pantalla de partidos que el usuario haya elegido. El identificador de pantalla actual de partidos representa la pantalla a la cual se necesita mover por cualquier motivo, sin haberlo elegido desde la pantalla de torneos.

Cuando no se entra a la pantalla de partidos, la clase controladora almacena el valor "-1" como id de la pantalla de partidos principal y la pantalla actual. La primera vez que se entre a la pantalla de partidos desde los torneos, la id de la pantalla actual es igual un número que corresponde a la posición de la lista del torneo elegido (asignado en la pantalla de torneos).

La pantalla actual de partidos ahora es diferente a la pantalla principal de partidos, por lo tanto se carga la lista pidiendo datos del partido al Modelo y se actualiza la id de la pantalla principal de partidos, la misma operación ocurre cada vez que se elige otro torneo.

Si el identificador de la nueva pantalla es el mismo que el identificador de la pantalla principal, no hace falta una nueva petición de datos al Modelo, se utilizan los datos de los partidos que ya se habían almacenado en los objetos. Una vez cargada la lista de partidos del torneo elegido, se añade los títulos de la categoría del torneo y el nombre del torneo para que la Vista los muestre en pantalla.

La Activity "modActivity.java" primero verifica que tipo de usuario está en la aplicación, luego se obtiene los datos del partido para mostrarlos en pantalla y cambiar los resultados, se puede ver su implementación en la figura 85.

```
if(!controller.user1.getUsuario().equals("-1") && controller.user1.getPermisos().equals("app")){
    tvUser.setText("USUARIO:"+controller.user1.getUsuario());
}
else{
    tvUser.setText("USUARIO:público");
}
//giving values to teams
tvEq1.setText(controller.partidos.getEqLocalPosition2(position));
tvEq2.setText(controller.partidos.getEqVisitantePosition2(position));
//giving values to gols
gLocal.setText(controller.partidos.getGolEqLocalPosition2(position));
gVisitante.setText(controller.partidos.getGolEqVisitantePosition2(position));
```

Figura 85: Controlador de “modActivity.java”

La Activity "LoginActivity.java" no realiza ningún tipo de control en el funcionamiento lógico de la pantalla, sólo los listeners están atentos a la pulsación de los botones de salir y iniciar sesión.

4.4.2.3.4.- Listeners de cada Activity

La Activity "MainActivity.java" tiene un listener para caputar el evento para hacer "swap down" (pulsar y desplazar el dedo hacia abajo) y enviar la orden de actualizar la lista de torneos, como se ve en la figura 86.

```
swipeView.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
    @Override
    public void onRefresh() {
        swipeView.setRefreshing(true);
        (new Handler()).postDelayed(new Runnable() {
            @Override
            public void run() {
                swipeView.setRefreshing(false);
                obtenerDatosWeb();
                Toast.makeText(getApplicationContext(), "Actualizado", Toast.LENGTH_SHORT).show();
            }
        }, 3000);
    }
});
```

Figura 86: SwipeView MainActivity

Los listeners están atentos a la pulsación de los elementos del menú contextual para ordenar actualizar la lista de torneos, iniciar o cerrar sesión. En caso de haber iniciado sesión, esa lista cambia y las opciones serán actualizar la lista de torneos, cerrar sesión y salir.

Existe un listener para cada uno de los elementos de la lista de torneos. Si se pulsa sobre algún elemento de la lista, se enviará el "id" del torneo a la pantalla de partidos, para poder ver los partidos de ese torneo.

Un listener está atento a la pulsación sobre un icono flotante, que está en la parte inferior de la pantalla y se utiliza para iniciar sesión. Cuando se pulsa sobre ese icono, se cambia la pantalla a otra para poder iniciar sesión. En caso de haber iniciado sesión, este botón no enviará nuevamente a la pantalla de inicio de sesión, simplemente mostrará un mensaje en la parte inferior de la pantalla (snackbar) indicando que ya se ha iniciado sesión.

La Activity "match4Activity.java" contiene listeners asociados a la pulsación de un elemento de la lista de partidos del torneo, dependiendo de los permisos que se tenga su comportamiento variará:

- Si el usuario tiene permisos para poder modificar el resultado de los partidos desde la aplicación Android, podrá elegir un item que lo enviará a otra pantalla para modificar el resultado.
- Si el usuario no tiene permisos para modificar los resultados, cuando se pulse el elemento de la lista de partidos se podrá ver un mensaje indicando el resultado del partido elegido.

También están activos los listeners del menú contextual que tendrán distintos valores según se haya iniciado sesión o no.

La Activity "modActivity.java" contiene listeners asociados a la pulsación de los botones salir (para salir de la pantalla de modificación de resultados) y "Modificar" (hace efectivo el proceso de modificación).

Cuando el listener del botón modificación se activa, se verifica que los valores introducidos sean los correctos y si han cambiado, en caso de haber cambiado se actualiza los resultados en memoria interna y en la base de datos del servidor web (mediante el Modelo de datos). Los listeners del menú contextual están atentos y se puede elegir cerrar sesión y salir.

La Activity "LoginActivity.java" tiene listeners asociados a la pulsación de los botones de salir (sale de la pantalla de login) y login (hace efectivo el proceso de inicio de sesión). Cuando se pulsa sobre el botón de inicio de sesión, se comprueba que los datos introducidos no estén vacíos como se ve en la figura 87.

```
public void onClick(View v) {
    //boton salir
    if (v.getId() == R.id.la_bSalir) {
        toMainActivity();
    }
    //boton cambiar
    else if (v.getId() == R.id.la_bLogin) {
        if(etUser.getText().toString().equals("") || etPass.getText().toString().equals("") ){
            Toast.makeText(getApplicationContext(), "Debe introducir Usuario y Contraseña", To
        }
        else {

            new loginTread().execute etUser.getText().toString(),etPass.getText().toString());
        }
    }
}
```

Figura 87: Ejecución en segundo plano del inicio de sesión

Se crea un nuevo hilo de procesamiento mediante Asynctask para que el proceso de inicio de sesión se haga por otro hilo de procesamiento, el nuevo hilo recibe los datos introducidos de nombre y contraseña y los envía al Modelo de datos para realizar las operaciones necesarias para el inicio de sesión.

4.5.- Pruebas/Implantación

A continuación se muestra un breve manual para utilizar la aplicación Android y la aplicación web, las interfaces de ambas son sencillas y fáciles de manejar.

4.5.1.- Aplicación Web

Si el usuario tiene permisos web, podrá iniciar sesión y comenzar a trabajar en la gestión de torneos desde la página web, la figura 88 representa la pantalla principal.

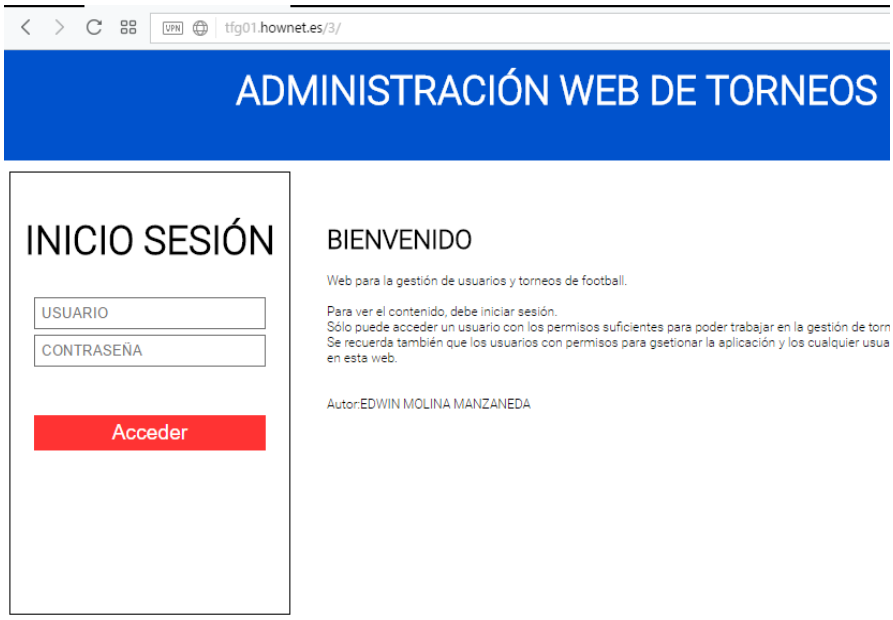


Figura 88: Página de inicio de la aplicación web

En la figura 89 se observa que un usuario ha iniciado sesión correctamente, el menú ha cambiado para que el usuario pueda administrar la gestión de usuarios y torneos desde la web.

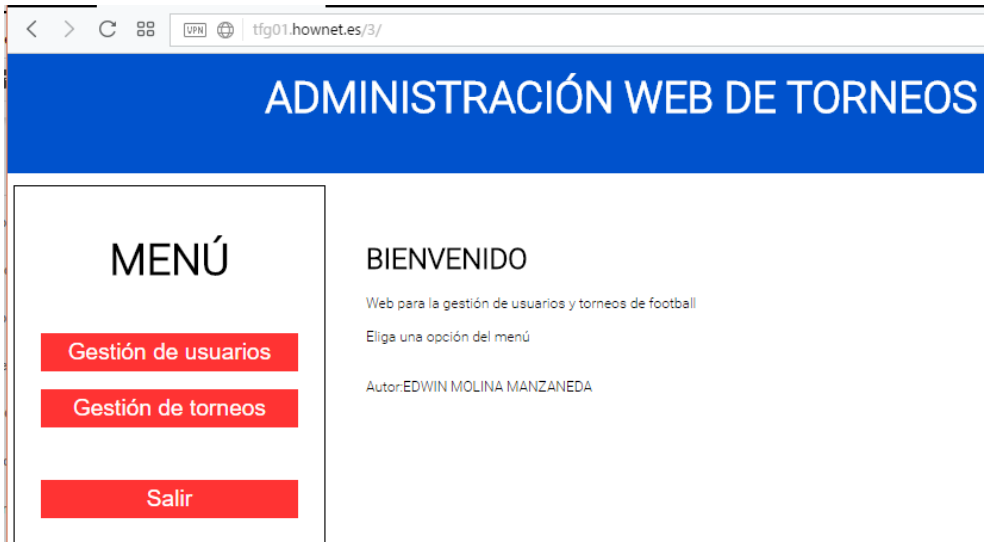


Figura 89: Usuario registrado en la aplicación web

Ahora si se desea administrar a los distintos usuarios que van a formar parte de la gestión de los torneos en las aplicaciones Android y web, se debe hacer clic en la opción “Gestión de usuarios” del menú, donde se podrá añadir usuarios con sus respectivas características, como se ve en la figura 90.

Figura 90: Añadiendo usuario en la aplicación web

En la figura 91, se puede apreciar un botón para listar usuarios, desde esa la lista se podrá borrar y modificar datos de cualquier usuario que se elija.

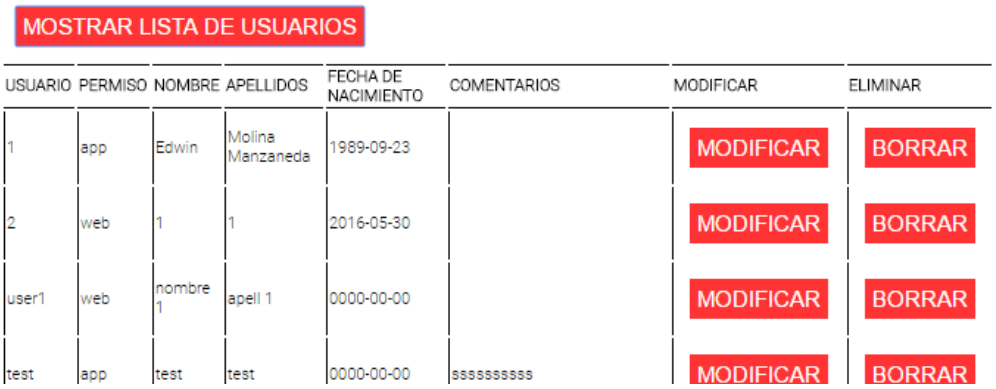


Figura 91: Listando usuarios en la aplicación web

Para borrar un usuario de la lista bastará con pulsar sobre el botón “borrar”. Si se desea cambiar los datos de cualquier usuario de la lista, se puede pulsar sobre el botón “modificar” que mostrará los datos del usuario listos para ser modificados, como se ve en la figura 92.

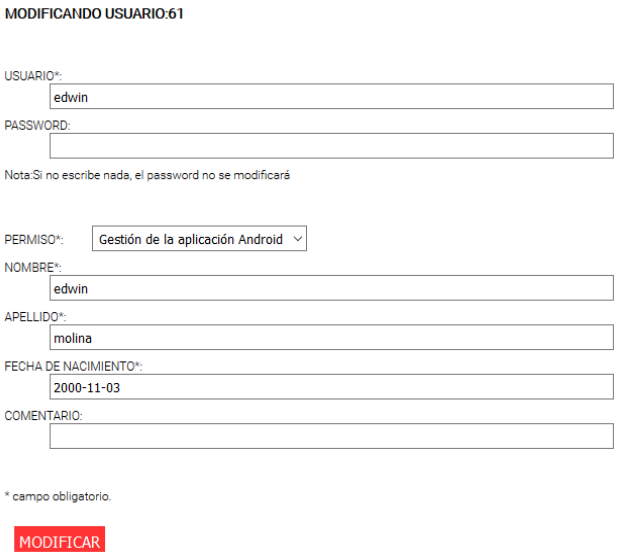


Figura 92: Modificando un usuario en la aplicación web

Cuando se elige la opción de "gestión de torneos" del menú principal, como se ve en la figura 93 se carga un submenú con todas las opciones para poder administrar los torneos.

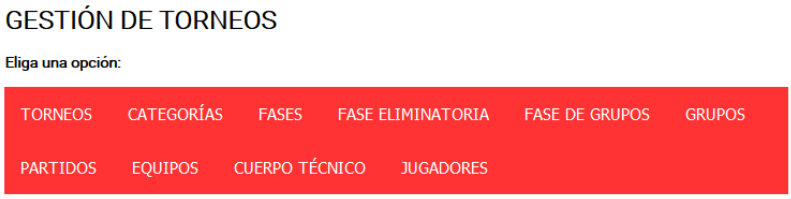


Figura 93: Menú de la gestión de torneos en la aplicación web

Bastará con elegir cualquier opción para comenzar a administrar las características de cada apartado. Por ejemplo, la figura 94 muestra la opción "cuerpo técnico" del submenú de la gestión de torneos, se carga la lista del personal del cuerpo técnico y se puede elegir el botón "modificar" o "borrar".

Figura 94: Listando el cuerpo técnico en la aplicación web

El botón “borrar” borrará los datos del cuerpo técnico elegido, si se elige el botón “modificar”, se cargará el formulario en el mismo sitio con las distintas opciones para modificar el cuerpo técnico elegido como se ve en la figura 95.

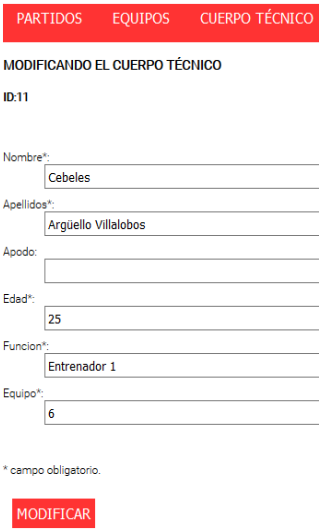


Figura 95: Modificando el cuerpo técnico de la aplicación web

El funcionamiento de todas las opciones de la gestión de torneos es el mismo que en la gestión de usuarios. Para salir de la aplicación web bastará con hacer clic en el botón “salir” del menú principal.

4.5.2.- Aplicación Android

Se debe instalar la aplicación Android en el dispositivo móvil, se puede observar en la figura 96 el icono de la aplicación instalada desde el cual se podrá iniciar la aplicación.

Figura 96: Icono de la aplicación Android

La pantalla principal cargará los torneos automáticamente, como se ve en la figura 97 se puede actualizar manualmente desde el menú contextual o utilizando la acción swipe down de la lista de torneos.

Figura 97: Pantalla principal de la aplicación Android

El usuario público puede elegir un torneo y se carga la pantalla con los partidos del torneo elegido como se ve en la figura 98. Si se pulsa sobre cualquier partido de la lista sólo se ve un mensaje con el resultado del partido elegido.

Figura 98: Pantalla de partidos de la aplicación Android

En la figura 99 se puede ver que un mensaje aparece en la barra de notificaciones del móvil, eso indica que se ha modificado un evento de algún partido. Cuando se pulse sobre la notificación, se dirigirá a la pantalla del partido modificado.

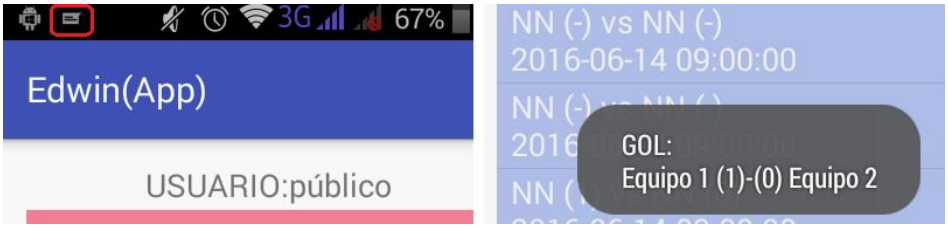


Figura 99: Notificación de un evento en la aplicación web

Si el usuario tiene permisos para modificar un partido, podrá iniciar sesión desde cualquier pantalla a través del menú contextual como se ve en la figura 100, en la pantalla principal existe el botón flotante con el icono del usuario que también envía a la pantalla de inicio de sesión.

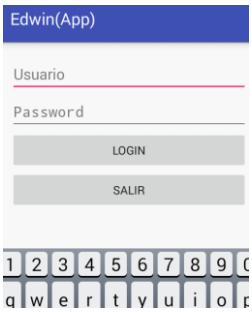


Figura 100: Login del usuario en la aplicación Android

Cuando el inicio de sesión sea satisfactorio, se dirige a la pantalla principal y en todas las pantallas se podrá ver el nombre del usuario que ha iniciado sesión, en la figura 101 se puede observar que el usuario llamado “1” ha iniciado sesión.

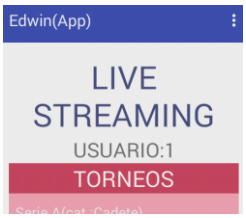


Figura 101: Usuario registrado en la aplicación Android

Ahora cuando se pulse sobre un partido se podrá ver la pantalla de modificación, la figura 102 muestra la opción de modificar el marcador del partido.

Figura 102: Modificando un partido en la aplicación Android

Cuando se modifique la partida un mensaje se activa en la barra de notificaciones del dispositivo móvil, indicando el evento ocurrido y si se pulsa sobre esa notificación se podrá ver un mensaje indicando el partido actualizado como se ve en la figura 103.

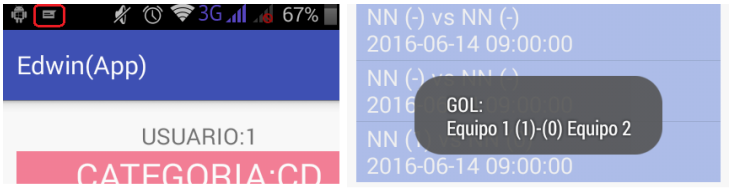


Figura 103: Notificación del evento modificado en la aplicación Android

Desde cualquier pantalla de la aplicación se puede cerrar sesión, la figura 104 muestra la opción para cerrar sesión desde el menú contextual.

Figura 104: Opción de cerrar sesión en la aplicación Android.

5.- CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se comentará las conclusiones finales del proyecto junto a los futuros desarrollos que se pueden llevar a cabo después de haber desarrollado el primer prototipo siguiendo la arquitectura de diseño propuesta.

5.1.- Conclusiones

En un principio se ha desarrollado una aplicación Android y Web inicial, en el que se buscaba poder ver y modificar datos de una única tabla de la base de datos de un servidor y de esta forma conocer las tecnologías que se van a utilizar en el desarrollo del prototipo.

La aplicación web trabajaba solo con html, php y sin ningún patrón de diseño, por lo tanto en un principio se había generado páginas web con código repetido y una mezcla importante de código php y html para gestionar y modificar una tabla de prueba. Ahora gracias al modelo de capas es posible gestionar toda la complejidad de su funcionamiento en una única página web que irá cambiando según las necesidades del usuario.

La aplicación Android tenía muchísimo código en la Activity principal y para poder gestionar la modificación de una tabla de prueba desde otra Activity, se ha repetido mucho el código de la Activity principal, generando una importante complejidad en cada Activity y a pesar de trabajar de manera modular, ya era complicado comprender su funcionamiento. Ahora las Activities siguen el modelo de capas y tienen un código más claro y su complejidad se ha reducido considerablemente.

Si el desarrollo de la aplicación Android y Web no iban a seguir el modelo de capas el primer prototipo iba a ser muy complejo. Por lo tanto, el Modelo Vista Controlador ha permitido poder trabajar de manera ordenada y eficiente, permitiendo reducir de manera importante la complejidad en el desarrollo de las aplicaciones. Por distintos motivos se ha dejado de desarrollar el prototipo por un tiempo, retomar su desarrollo no ha costado mucho tiempo ni ha sido complejo, además ahora su mantenimiento y ampliación será más fácil.

Gracias a la librería Volley se ha podido gestionar los recursos hardware y software del dispositivo móvil de manera eficiente a la hora de gestionar las peticiones y respuestas de datos de un servidor web, ya que es una tarea que si no se gestiona bien, puede llegar a consumir mucha batería, datos y memoria de manera inútil causando muchos inconvenientes que el usuario al final lo llega a notar.

Es complicado poder ajustar la manera convencional de trabajar de Volley para poder trabajar según el modelo de capas, ya que existe una clase controladora que utiliza la librería Volley según la necesidad de las Activities.

La gestión de hilos de procesamiento para las peticiones get y post de la librería Volley permite conocer cuando es necesario utilizarlos en Android, gracias a ello la gestión de envío y recepción de datos, como el login del usuario se realiza en segundo plano (utilizando distintos hilos de procesamiento), el controlador es el encargado de gestionar los hilos, envía datos al modelo para verificar su validez y gestiona el resultado devuelto por el modelo para que la vista enseñe al usuario el resultado.

En conclusión se ha conseguido alcanzar todos los objetivos que se habían planteado.

5.2.- Posibles desarrollos futuros

Ahora el resultado del primer prototipo Android y web es sólo para demostrar su correcto desarrollo siguiendo el Modelo Vista Controlador, su funcionalidad es muy básica, es necesario que los usuarios realicen pruebas para luego poder ampliar el funcionamiento según el feedback que se reciba.

Por ejemplo, el primer diseño de la aplicación web está pensado para administrar su contenido conociendo la base de datos, debería adaptarse a cualquier tipo de cliente y ser más sencillo. La aplicación Android requiere ampliar la publicación y modificación de eventos que ocurran en un partido.

Será necesario continuar con el ciclo de vida del prototipo hasta conseguir un programa en condiciones aceptables para poder ofrecerlo a los clientes y explotarlo económicamente.

En todo caso siguiendo este modelo de trabajo se podrá ampliar el desarrollo de este programa a distintas plataformas como los dispositivos de Apple o Windows.

6.- BIBLIOGRAFÍA

[1] Título: Google Play
https://play.google.com/store?hl=en
Fecha de consulta: 09/04/2017

[2] Título: La sociedad de la información en España 2016
http://www.fundaciontelefonica.com/arte_cultura/sociedad-de-la-informacion/informe-sie-espana-2016/
Fecha de consulta: 19/04/2017

[3] Título: Definitions
http://whatis.techtarget.com
Fecha de consulta: 19/04/2017

[4] Título: NetBeans IDE
https://netbeans.org
Fecha de consulta: 20/04/2017

[5] Título: Introduction to Android
https://developer.android.com/guide/index.html
Fecha de consulta: 20/04/2017

[6] Título: Explore the benefits of JSON and XML in Android applications
http://www.ibm.com/developerworks/library/x-andbene1
Fecha de consulta: 25/04/2017

[7] Título: Json vs XML
https://blog.udemy.com/json-vs-xml-como-json-es-superior-a-xml/
Fecha de consulta: 25/04/2017

[8] Título: Application
https://developer.android.com/reference/android/app/Application.html
Fecha de consulta: 24/04/2017

[9] Título: Understanding the Android Application Class
https://guides.codepath.com/android/Understanding-the-Android-Application-Class
Fecha de consulta: 01/05/2017

[10] Título: AsyncTask
https://developer.android.com/reference/android/os/AsyncTask.html
Fecha de consulta: 02/05/2017

[11] Título: Socket
https://developer.android.com/reference/java/net/Socket.html
Fecha de consulta: 01/05/2017

[12] Título: Sending and Receiving Data with Sockets
https://github.com/codepath/android_guides/wiki/Sending-and-Receiving-Data-with-Sockets
Fecha de consulta: 01/05/2017

[13] Título: HttpURLConnection
https://developer.android.com/reference/java/net/HttpURLConnection.html
Fecha de consulta: 01/05/2017

[14] Título: Transmitting Network Data Using Volley
https://developer.android.com/training/volley/index.html
Fecha de consulta: 01/05/2017

[15] Título: Android working with Volley Library
http://www.androidhive.info/2014/05/android-working-with-volley-library-1/
Fecha de consulta: 01/05/2017

[16] Título: An Introduction to Volley
http://code.tutsplus.com/tutorials/an-introduction-to-volley--cms-23800
Fecha de consulta: 01/05/2017

[17] Título: Model-View-Controller
https://msdn.microsoft.com/en-us/library/ff649643.aspx
Fecha de consulta: 24/04/2017

[18] Título: HTML5
https://www.w3schools.com/html/html5_intro.asp
Fecha de consulta: 23/04/2017

[19] Título: Ajax Introduction
https://www.w3schools.com/xml/ajax_intro.asp
02/05/2017

[20] Título: Database Directory
http://www.databasedir.com
Fecha de consulta: 19/04/2017

[21] Título: PL/SQL - Triggers
https://www.tutorialspoint.com/plsql/plsql_triggers.htm
Fecha de consulta: 28/04/2017

[22] Título: Create Trigger (Transact-SQL)
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql
Fecha de consulta: 28/04/2017