UNIVERSIDAD SAN FRANCISCO DE QUITO

Colegio de Ciencias e Ingeniería

Un torneo en el bolsillo: Desarrollo de una aplicación nativa móvil Android para la generación y administración de torneos de fútbol en formato de liga

Andrés Cárdenas

Daniel Fellig, M.Sc., Director de Tesis

Tesis de grado presentada como requisito para la obtención del título de Ingeniero en Sistemas

Quito, diciembre de 2014

Universidad San Francisco de Quito
Colegio de Ciencias e ingeniería

HOJA DE APROBACIÓN DE TESIS

Un torneo en el bolsillo: Desarrollo de una aplicación nativa móvil Android para la generación y administración de torneos de fútbol en formato de liga

Andrés Fabián Cárdenas Alfonso

Daniel Fellig, M.Sc.,
Director de Tesis _____

Fausto Pasmay, M.Sc. ,
Miembro del Comité de Tesis _____

Mauricio Iturralde, Ph.D.,
Miembro del Comité de Tesis _____

Ximena Córdova, Ph.D.,
Decana Escuela de Ingeniería
Colegio de Ciencias e Ingeniería _____

Quito, diciembre de 2014

Firma: ----------
Nombre: Andrés Fabián Cárdenas Alfonso
C. I.: 1715152870
Fecha: Quito, diciembre de 2014

RESUMEN

El presente proyecto de tesis conlleva el diseño y desarrollo de una aplicación nativa para dispositivos Android utilizando el lenguaje de programación Java que permita la generación y administración de torneos de fútbol, ofreciendo esta funcionalidad sin necesidad de una conexión permanente al internet.

Para el funcionamiento de esta aplicación se diseña y desarrolla una serie de Web Services en el lenguaje de programación JavaScript utilizando el Framwork NodeJS. Estos Web Services son necesarios para la conexión entre los dispositivos Android y la base de datos central de torneos que se encuentra en el internet. Estos Web Services pueden ser reutilizados para desarrollar aplicaciones similares para diferentes dispositivos.

La aplicación complementa a un servicio existente solo a través de una plataforma web conocida como Tourngen Web, permitiendo expandir la funcionalidad de este sistema. Tourngen Web permite la creación de usuarios, torneos, equipos, generación automática de partidos, ingreso de resultados y cálculo automático de las posiciones de los equipos dentro del torneo. La aplicación de Android ofrece la misma funcionalidad sin necesidad de mantener una conexión permanente al internet, además de poder administrar los torneos creados vía web y crear torneos que pueden ser administrados vía web.

ABSTRACT

The present thesis project comprises the design and development of a native application for Android devices using the Java programming language that allows the creation and administration of soccer tournaments, offering this functionality without the need for a permanent internet connection.

The inner workings of this application requires the design and development of a series of Web Services written in the JavaScript programming language using the NodeJS Framework. These Web Services are needed for the connection between the Android devices and the central database for the tournaments that is stored in the Internet. These Web Services can be reused for development of similar applications for different devices.

This application complements an existing service only reachable through web browsers known as Tourngen Web, allowing for an expansion of that system´s functionality. Tourngen Web allows the creation of users, tournaments, teams, automatic generation of matches, submitting match results and automatic calculation of positions based on match scores. The Android application offers the same functionality without the need for a permanent Internet connection; besides allowing the user to manage those tournaments already created through the web platform and creating tournaments that can be managed using the web platform.

Tabla de Contenidos

Tabla de Figuras

Tabla de Tablas

Capítulo 1 – Introducción

1.1 Antecedentes

El futbol es considerado el deporte más popular a nivel mundial. Es por esto que se juegan una cantidad muy grande de campeonatos en todo el planeta. Aunque existen algunas variaciones con respecto a las reglas específicas de juego, los formatos de los torneos son esencialmente los mismos, por lo que se han creado algunas herramientas para la administración de todos estos tipos de torneos. Los torneos se pueden clasificar principalmente en tres tipos: formato de liga, eliminación directa y los torneos mixtos o híbridos, que combinan los dos formatos anteriores. El enfoque de esta aplicación es el primer formato mencionado.

Un torneo en formato de liga reúne a todos los equipos participantes en un mismo grupo, en el cual todos los competidores juegan un partido contra todos los demás participantes, a menos de que se defina que el torneo sea “ida y vuelta”, lo que implica que se juegan dos partidos en vez de uno solo. Una liga se puede jugar con un mínimo de dos equipos y no existe un límite máximo definido, aunque la duración del torneo aumenta con la cantidad de equipos participantes. En los torneos de futbol existe el concepto de una jornada, la cual es una agrupación de partidos. Todos los partidos dentro de una jornada se juegan en un mismo margen de tiempo, usualmente un periodo de uno o dos días. Dentro de una jornada, un equipo debe jugar exactamente un solo partido. La única excepción es cuando existen una cantidad de equipos impares. En este caso, cada equipo tiene una jornada de descanso, o dos jornadas de descanso si es que el torneo es “ida y vuelta”. Al jugarse cada partido se asignan puntos a los participantes dependiendo del resultado. Un equipo recibe tres puntos en caso de ser el ganador, un punto en caso de haber empatado y cero puntos por una derrota. La cantidad de puntos es el principal criterio para definir al ganador de este torneo. En caso de empate se suele usar el gol diferencia como segundo criterio. El gol diferencia consiste en la diferencia entre la cantidad de goles hechos y la cantidad de goles recibidos durante todo el campeonato. Al

finalizar todos los partidos de la liga, el equipo en la primera posición es nombrado campeón. En caso de existir empate por la primera posición, después de considerar los puntos y el gol diferencia, cada organizador de torneo suele definir los siguientes criterios para desempate. Estos pueden ser los goles a favor durante todo el torneo o el último resultado entre los equipos involucrados. El organizador del torneo podría incluso organizar un partido entre los equipos empatados para definir al ganador.

La administración de estos torneos se suele hacer a mano, utilizando hojas de cálculo en los mejores casos. La mayoría de organizadores suelen hacer un trabajo muy repetitivo en cada edición de sus torneos. Primero, reciben hojas de inscripción llenadas a mano para ingresar los datos de los equipos a su campeonato. Luego, crean un cronograma de partidos de forma manual, asegurándose que se cumplan los requerimientos. Es decir, deben revisar principalmente que un equipo solo juegue un partido por jornada y que todos los equipos jueguen contra todos los demás equipos exactamente una o dos veces, dependiendo del formato decidido. Después, manualmente se transmite este cronograma a los participantes, ya sea anexando el calendario en un e-mail o imprimiéndolo y publicándolo en algún medio que llegue a los participantes. Una vez que empieza el campeonato, los resultados de los partidos se ingresan en lápiz y papel por alguna persona encargada de supervisar los partidos. Al final de las jornadas se reciben estos papeles y se procesan los resultados manualmente para generar una tabla de posiciones. Este proceso se repite para cada jornada jugada hasta que se termine el campeonato. Estos torneos suelen tener desde cinco hasta 30 equipos participando, además de que un organizador puede estar a cargo de muchos torneos simultáneos. El procesamiento manual de esta información para la administración de un torneo ocupa mucho tiempo, el cual se reduciría en gran cantidad con una herramienta para automatizar este proceso.

1.2 Justificación del Proyecto

Cada día se juegan miles de partidos a nivel mundial. Muchos de estos partidos forman parte de diferentes tipos de torneos, de diferentes magnitudes y a diferentes escalas. Incluso se juegan partidos virtuales, en los que los partidos suceden en consolas de videojuegos. Los torneos se pueden jugar entre un pequeño grupo de amigos o pueden llegar a involucrar a los mejores jugadores de diferentes países. Aunque las magnitudes de estos torneos son muy diferentes, los formatos son prácticamente los mismos, por lo que podrían ser administrados con una sola herramienta.

Todas las personas que administren torneos de futbol se verían beneficiados al usar esta aplicación. Aquellos que organicen pequeños torneos virtuales podrían empezar un torneo sencillo en su teléfono y mantener control sobre los resultados. Muchos jóvenes disfrutan de reunirse a jugar este tipo de torneos. Esta funcionalidad para administrar un torneo simplificado estaría disponible inclusive sin conexión a internet, por lo que se podría jugar un torneo de este tipo sin mayor complicación y sin necesidad de usar un computador.

Por otro lado, existen aquellos que organizan torneos reales más completos. Para la creación de estos torneos se utilizaría la interfaz web propuesta por Ricardo Herrera. El ingreso de resultados se podría hacer de manera simplificada por medio de la aplicación de Android. Además permitiría hacer consultas varias, como por ejemplo: resultados, calendarios de partidos o tablas de posiciones. Esta funcionalidad será posible con conexión a internet o sin ella, implementando una lógica de sincronización fuera de línea.

Se escogió una aplicación nativa para dispositivos Android ya que permite llevar la portabilidad de este aplicativo a las canchas en donde son jugados los partidos. Es mucho más probable que una persona lleve consigo un teléfono celular o una Tablet más que un computador portátil. Los dispositivos Android forman una gran parte del mercado de dispositivos móviles debido a su estabilidad y su accesibilidad en precios.

Por otra parte, se decidió implementar una aplicación nativa ya que una aplicación web no permitiría su uso cuando no hay una conexión permanente al internet, mientras que

una aplicación nativa podría ser usada sin conexión y, cuando el dispositivo se conecte al internet, sincronizar los datos con la base de datos central. La cantidad de dispositivos que requieren de una conexión Wi-Fi para acceder al internet es muy alta y estas conexiones no siempre están disponibles en las canchas donde se juegan estos partidos.

1.3 Objetivo General

El objetivo final del proyecto es el de implementar una aplicación nativa para dispositivos Android que se conecte por medio de internet a un servidor centralizado y permita el ingreso de resultados y consultas de partidos de futbol administrados en este sistema. Además, que permita la generación de torneos. Todas estas funcionalidades están disponibles con acceso a internet o fuera de línea (con ciertas limitaciones).

1.4 Objetivos Específicos

1. Base de datos
    - Analizar base de datos existente de Tourngen Web
    - Definir modificaciones a la base de datos Tourngen Web
    - Diseñar tabla de *tokens* para identificación de clientes móviles

      Tabla utilizada para mantener una asociación de un dispositivo con una cuenta de usuario para evitar que se deba mandar nombre de usuario y contraseña en cada interacción con los Web Services.

        i. Token
        ii. Identificador del usuario correspondiente
        iii. Identificador del dispositivo UUID
        iv. Fecha de expiración
2. Implementación de modificaciones a la base de datos
    - Generar script para implementar modificaciones
    - Ejecutar script en motor de la base de datos de desarrollo
    - Generar script para cargar datos de prueba

- Ejecutar script para cargar datos de prueba
- Programar respaldo automático de la base de datos
- Probar que la aplicación web siga funcionando adecuadamente

3. Tourngen Web
    - Analizar funcionalidad que se debe obtener de Tourngen Web
    - Definir modificaciones o necesidades para la aplicación móvil
        i. Se requiere tener acceso a la funcionalidad de autenticación y generación de hash de contraseñas de usuarios para poder usar esta funcionalidad
    - Implementar modificaciones o definir formas de acceso para los Web Services a estas funcionalidades
    - Probar que la aplicación web siga funcionando adecuadamente
4. Implementar Web Services para login
    - Escucha una llamada GET a https://www.tourngen.com:2800/login/:username
        i. Establece una conexión segura con el cliente usando https (Esto se repite en todos los Web Services por lo que no se menciona en los siguientes casos)
        ii. El cliente envía nombre de usuario, contraseña y un identificador para el cliente (UUID)
        iii. Se comparan los datos recibidos con la base de datos
            1. Si son correctos, genera un *token* para identificar al cliente en conexiones futuras. El *token* se almacena en la base de datos y se lo retorna al cliente.
            2. Si son incorrectos retorna un estatus de error. (En caso de error, siempre se envía un mensaje de error por lo que no se menciona en los siguientes Web Services).

5. Implementar Web Services para un conjunto de torneos
    - Escucha una llamada GET a https://www.tourngen.com:2800/Tournament/
        i. El cliente envía *token* de acceso, del cual se identifica el usuario y los privilegios correspondientes. (Esto se repite en todos los Web Services que siguen por lo que no se menciona en los siguientes casos)
        ii. Retorna una lista con todos los torneos que el usuario puede ver
    - Escucha una llamada POST a https://www.tourngen.com:2800/Tournament/
        i. El cliente envía un JSON representando el torneo que será creado.
        ii. Si el torneo es creado correctamente, retorna el id con el que se identificará el torneo en la base de datos
6. Implementar Web Services para un torneo específico
    - Escucha una llamada GET a https://www.tourngen.com:2800/Tournament/:id
        i. El cliente envía el id del torneo que desea obtener
        ii. Retorna un JSON con los datos del torneo seleccionado
    - Escucha una llamada PUT a https://www.tourngen.com:2800/Tournament/:id
        i. El cliente envía un JSON representando los datos del torneo que serán modificados
        ii. Si el torneo es editado correctamente, retorna un estatus de éxito
    - Escucha una llamada POST a https://www.tourngen.com:2800/Tournament/:id
        i. Web Service usado para la sincronización de bases modificadas offline

ii. El cliente envía un JSON con los identificadores del torneo, los equipos y los partidos que lo componen, además de las fechas de última actualización de cada uno.

iii. El Web Service valida las fechas de última actualización del JSON con las de la base de datos. Si se reconoce que las fechas no coinciden, se considera que esa entidad debe ser actualizada.

1. En caso de no necesitar actualización en ninguna entidad, retorna un status de “actualizado”
2. En caso de que alguna entidad necesite actualización, retorna un status de “debe actualizar”, acompañado de un JSON que contiene los identificadores de las entidades a ser actualizadas, el tipo de entidad que se debe actualizar, y la operación a ser realizada (GET o PUT) para cada entidad.

7. Implementar Web Services para un conjunto de equipos

- Escucha una llamada GET a https://www.tourngen.com:2800/Tournament/:id/Team/

i. El cliente envía el id del torneo al que pertenecen los equipos

ii. El Web Service Retorna un JSON con los equipos que participan en el torneo

8. Implementar Web Services para un equipo específico

- Escucha una llamada GET a https://www.tourngen.com:2800/Team/:id

i. El cliente envía el id del equipo que desea ver

ii. Retorna un JSON con la información del equipo deseado y los partidos que juega ese equipo en el torneo.

9. Implementar Web Services para un conjunto de jornadas

- Escucha una llamada GET a https://www.tourngen.com:2800/Tournament/:id/Fixture/
    i. El cliente envía el id del torneo al que pertenecen las jornadas.
    ii. Retorna un JSON con las jornadas que componen al torneo

10. Implementar Web Services para una jornada específica
    - Escucha una llamada GET a https://www.tourngen.com:2800/Fixture/:id
        i. El cliente envía el id de la jornada que desea ver
        ii. Retorna un JSON con la información de la jornada y los partidos que la componen.

11. Implementar Web Services para un conjunto de partidos
    - Escucha una llamada GET a https://www.tourngen.com:2800/Tournament/:id/Match/
        i. El cliente envía el id del torneo al que pertenecen los partidos
        ii. Retorna un JSON con todos los partidos que componen al torneo, organizados por jornadas.

12. Implementar Web Services para un partido específico
    - Escucha una llamada GET a https://www.tourngen.com:2800/Match/:id
        i. El cliente envía *token* de acceso y el id del partido que desea ver
        ii. Retorna un JSON con la información del partido.
    - Escucha una llamada PUT a https://www.tourngen.com:2800/Match/:id
        i. El cliente envía el id del partido que desea editar y un JSON representando los datos del partido que serán editados.
        ii. Retorna un estatus de éxito si la actualización fue exitosa.

13. Implementar módulo en aplicación de Android para login
    - Solicita el nombre de usuario, la contraseña, y un nombre para identificar al cliente

- Ejecuta una llamada GET a https://www.tourngen.com:2800/login
    i. Establece una conexión segura con el servidor usando https (Todas las conexiones al servidor usan https por lo que no se menciona en los siguientes módulos).
    ii. Envía nombre de usuario, contraseña y nombre para identificarse
    iii. Recibe un JSON con un status
        1. Si el status es de éxito, almacena el *token* en el almacenamiento permanente del teléfono.
        2. Si el status es de error, muestra un mensaje de error al usuario y vuelve a solicitar el login

14. Implementar funcionalidad para la sincronización
    - Envía un JSON con los identificadores de cada entidad que compone a un torneo (torneo, equipos, jornadas y partidos) y la última fecha de modificación de estos
    - El Web Service retorna un status de actualizado o no actualizado. En caso de que esté actualizado se considera sincronizado el torneo
    - Si no está actualizado, el JSON retornado contiene el identificador de la o las entidades no actualizadas y la operación que se debe ejecutar para que esa entidad esté sincronizada de nuevo.
    - Se ejecutan las operaciones en secuencia, procurando que todas se ejecuten correctamente.
15. Implementar módulo en aplicación de Android para Torneos
    - Muestra la lista de torneos a las que el usuario tiene acceso
    - Ejecuta una llamada GET a https://www.tourngen.com:2800/Tournament/
        i. Envía token para identificarse (En todos los siguientes módulos se repite este proceso por lo que no es mencionado)
        ii. Recibe un JSON con un status

1. Si el status es de éxito, extrae los torneos que puede ver del JSON y los carga en la base local.

- Después de haber actualizado los torneos, despliega la lista para que el usuario pueda escoger un torneo
- Ofrece la opción de actualizar manualmente haciendo llamada POST a https://www.tourngen.com:2800/Tournament/:id
    i. Envía un JSON con los identificadores del torneo, los equipos y los partidos que lo componen, además de las fechas de última actualización de cada uno.
    ii. Recibe un JSON con un status
        1. Si el status es “actualizado” no hay cambios en la base
        2. Si el estatus es de “debe actualizar”, recibe un JSON describiendo las operaciones GET y PUT que deben ejecutarse para sincronizar la base de datos. Este módulo se encarga de que se ejecuten estas operaciones antes de continuar y de almacenar las modificaciones en la base de datos local.

16. Implementar módulo de Android de creación de Torneos

- Solicita los datos básicos del Torneo
    i. Nombre
    ii. Fecha de inicio
    iii. Fecha de finalización
    iv. Ida y vuelta
    v. Descripción
    vi. Público (Booleano)
- Solicita los equipos que participarán en el torneo
    i. Nombre del equipo

ii. Descripción

- Genera las jornadas de los partidos de manera aleatoria
    i. Si las jornadas generadas no son del agrado del usuario se puede volver a generar hasta llegar a una jornada que se acepte
- Al finalizar, ofrece la opción de publicar el torneo si el cliente tiene conexión al internet para cargar el torneo en la base de datos central.
    i. Ejecuta una llamada POST a https://www.tourngen.com:2800/Tournament/
    ii. Envía un JSON representando el torneo a ser creado
    iii. Recibe un JSON con un status
        1. Si el status es de éxito, recibe los identificadores de las entidades que fueron creadas en el servidor central (torneo, equipos, jornadas y partidos)

17. Implementar módulo de Android de consulta de equipos
    - Despliega una lista de equipos que participan en un torneo.
    - Si uno selecciona un equipo despliega la información del equipo, así como un listado de los partidos del equipo en ese torneo.
18. Implementar módulo de Android de consulta de jornadas
    - Despliega una lista de jornadas que componen un torneo.
    - Si uno selecciona una jornada, despliega la información de la jornada, así como un listado de los partidos que se van a jugar o se jugaron en esa jornada.
19. Implementar módulo de Android de consulta de partidos
    - Despliega una lista de partidos que componen un torneo, agrupados por la jornada a la que corresponden.
    - Si uno selecciona un partido, despliega la información del partido.

- Mientras se tiene un partido seleccionado, se puede editar su resultado. Si no hay conexión modifica los datos de la base de datos local. Si hay conexión se conecta al Web Services correspondiente.
    i. Ejecuta una llamada PUT a https://www.tourngen.com:2800/Match/:id
    ii. Establece una conexión segura con el servidor usando https
    iii. El cliente envía *token* de acceso, el id del partido que desea ver y un JSON con los datos actualizados del partido.
    iv. Recibe un JSON con un status
        1. Si el status es de éxito actualiza la información en la base de datos local.

20. Implementar módulo de Android de tabla de posiciones
    - Despliega la tabla de posiciones para un torneo seleccionado. Carga los datos de la base local y procesa los resultados de los partidos para desplegar la tabla de posiciones de los equipos ordenada por puntos conseguidos.
21. Probar aplicación
    - Comprobar que los datos cargados previamente se puedan visualizar correctamente en la aplicación.
    - Comprobar que la aplicación pueda generar torneos nuevos y manejar la información adecuadamente.
    - Comprobar que se respeten los permisos previamente establecidos para un torneo creado en la interfaz web.
22. Distribuir aplicación
    - Subir el instalador de la aplicación a una web para que pueda ser descargado directamente.

1.5 Metas

1. Investigar tecnología adecuada para los Web Services necesarios teniendo en mente escalabilidad, desempeño y complejidad, además de los costos monetarios que podrían estar involucrados.
2. Investigar métodos para consumir Web Services de forma segura, y como se maneja el inicio de sesión en los clientes móviles de servicios similares.
3. Implementar los Web Services de forma estructurada y documentada, tal que en un futuro sean fácilmente reusados para desarrollar otros clientes.
4. Implementar la aplicación de Android que permita consumir e interactuar con estos Web Services, de manera que se puedan crear torneos, equipos, jornadas y partidos.
5. Implementar la siguiente funcionalidad en la aplicación de Android: consultar torneos, equipos, jornadas y partidos existentes en la base de datos central y poder interactuar con estos torneos respetando los diferentes niveles de permisos.
6. Implementar las estrategias de sincronización sin conexión que permita que la aplicación pueda ser usada sin una conexión permanente al internet.
7. Ejecutar pruebas del correcto funcionamiento de la aplicación y que se mantenga la coherencia de la base de datos, tanto local como central.
8. Distribuir la aplicación por medio de una descarga directa desde la aplicación web.

1.6 Actividades

1. Diseño de la base de datos
2. Implementación de Web Services de Login
3. Implementación de Web Services de Torneos
4. Implementación de Web Services de Equipos
5. Implementación de Web Services de Jornadas
6. Implementación de Web Services de Partidos

7. Diseño de la base de datos para Android
8. Implementación de módulo de Android para Login
9. Implementación de módulo de Android para Torneos
10. Implementación de módulo de Android para Crear Torneos
11. Implementación de módulo de Android para Equipos
12. Implementación de módulo de Android para Jornadas
13. Implementación de módulo de Android para Partidos
14. Implementación de módulo de Android de tabla de posiciones
15. Pruebas de la aplicación
16. Distribuir aplicación

1.7 Requerimientos de la aplicación

1.7.1 Registro

El usuario podrá usar el cliente móvil para crear un usuario que pueda ser usado de igual manera en el cliente móvil, así como la interfaz web y todas las demás interfaces que hagan uso de los *Web Services* desarrollados anteriormente. Para esto, el usuario escogerá la opción de registrarse en la vista de *login*. Se le solicitará su nombre, apellido, cuenta de e-mail y la contraseña que escoja. En caso de registrarse con éxito se le regresará a la ventana de *login* con un mensaje de éxito. En caso de error se notificará el error al usuario.

1.7.2 Login

Con una cuenta creada en el móvil o en el cliente web, el usuario podrá ingresar su nombre de usuario y su contraseña en el módulo de *login*. Este hace la validación en la base de datos central y recibe el *token* con el que la aplicación de identificará con los *Web Services* mientras el *token* siga siendo válido. Se puede hacer uso de la aplicación sin hacer *login*, pero no existirá la funcionalidad de publicar torneos a los servidores centrales o consultar torneos de internet.

1.7.3 Crear Torneo

En el caso de crear torneo, se le solicitará al usuario los datos del torneo, tales como el nombre, la fecha de inicio y fecha de fin, si es ida y vuelta, si es público o privado y una descripción. Luego se le solicita ingresar los equipos que participarán en el torneo, solicitando solo un nombre del equipo y una descripción opcional. Una vez completa la lista de equipos se muestra la pantalla de generación de partidos. El dispositivo crea un cronograma de partidos aleatoriamente respetando las condiciones que tienen los torneos en formato de liga. Es decir, todos los equipos juegan exactamente una o dos veces entre sí (dependiendo si es ida y vuelta o no) y que ningún equipo juega más de un partido por jornada. Si el usuario no está contento con el resultado puede generar de nuevo el cronograma de manera aleatoria. Cuando se acepte un cronograma se ofrece la opción al usuario de publicar el partido al internet (si ha iniciado sesión y cuenta con conexión a internet). Una vez finalizado el proceso se regresa a la lista principal de torneos.

1.7.4 Consultar Torneo

La experiencia del usuario al consultar un torneo varía ligeramente dependiendo del nivel de privilegio que tiene. Después de escoger un torneo del listado principal de torneos, el usuario ve la tabla de posiciones del torneo. Ahí tiene tres diferentes opciones. Puede ver los equipos del torneo, un listado de todos los partidos del torneo o ver los partidos por jornada.

- Al escoger un equipo, el usuario ve el nombre del equipo, la descripción y un listado con los partidos que juega o ha jugado el equipo en el torneo.
- Al escoger ver todos los partidos de un torneo, se le presenta un listado completo de los partidos con los resultados (si ya se jugó), ordenados por jornadas.
- Al escoger ver los partidos por jornada se le presenta al usuario un listado de partidos jugados en una sola jornada, con la opción de escoger la jornada que se desea consultar.

Al escoger un partido en cualquiera de los listados previos se presenta la pantalla de detalle del partido. Dependiendo del nivel de privilegio, el usuario tendrá diferentes interfaces.

1. Administrador de torneo o digitador

   Se le presenta el resultado actual del partido, así como la opción de editar el marcador y editar el estado del partido entre jugado o no jugado.

2. Representante de equipo o Seguidor

   Se le presenta el resultado del partido.

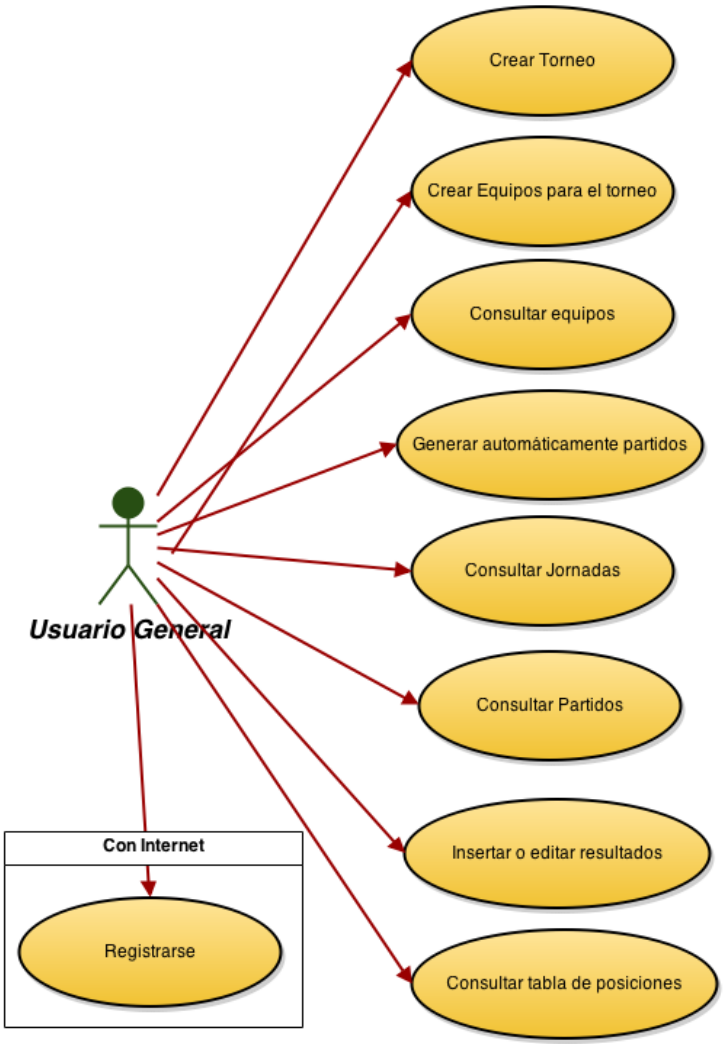

Figura 1.1: Casos de uso del Usuario General

La figura 1.1 muestra todas las tareas que puede ejecutar un usuario sin necesidad de haberse autenticado con el servidor central de *Tourngen Web*. Casi todas estas actividades pueden ser ejecutadas con o sin conexión a internet, considerando que

el torneo a ser consultado ya exista en la memoria del dispositivo. La única actividad que necesita acceso al internet es la de registrarse, que permite crear un usuario nuevo para poder ejecutar las tareas en la figura 1.2.

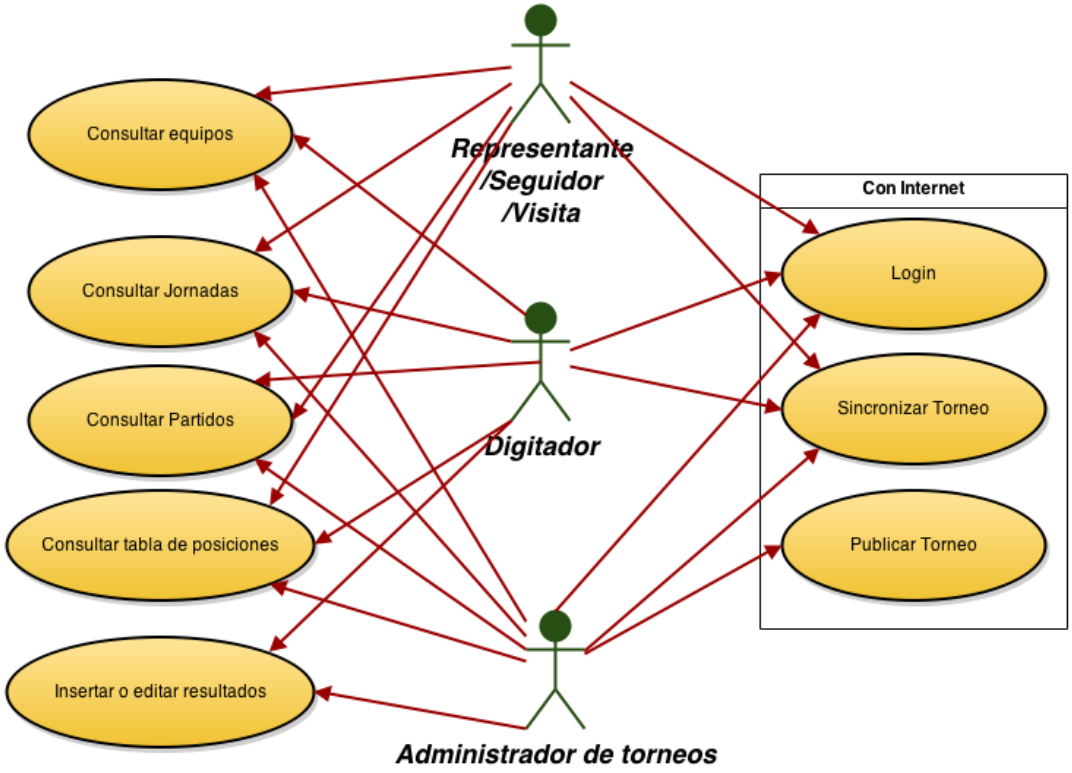

Figura 1.2: Casos de uso por privilegios

Una vez se está autenticado se pueden ejecutar las tareas correspondientes al privilegio que tenga el usuario con respecto al torneo específico. Las tareas entre los diferentes privilegios son esencialmente las mismas, excepto que el digitador y al administrador de torneos pueden ingresar o editar resultados de partidos del torneo. Otra diferencia es que solo el administrador de torneos puede publicar un torneo, lo cual es lógico ya que para publicar un torneo este debe haber sido creado en el dispositivo, lo que automáticamente convierte al creador en un administrador de torneos.

Capítulo 2 – Marco Teórico

2.1 Tecnología a ser Utilizada

2.1.1 Lenguajes de programación

Definición

Un lenguaje que ha sido diseñado para que un ser humano pueda darle instrucciones a un computador. Tienen sintaxis estrictas tal que las instrucciones dadas sean muy claras y no haya errores de interpretación. Los lenguajes de programación se suelen diferenciar entre sí por la sintaxis, la manera en la que manejan variables, las funciones, el manejo de condiciones y el manejo de lazos. Algunos lenguajes son más óptimos para diferentes tipos de tareas, tales como programación en dispositivos móviles, aplicaciones de escritorio o aplicaciones web. (Halfpap, 2013) El saber escoger el lenguaje de programación adecuado para la tarea requerida es uno de los pasos más importantes para el éxito de la tarea.

Javascript

Un lenguaje de programación interpretado. Su sintaxis es similar a los lenguajes de programación C y JAVA. Sin embargo, su principal diferencia con los anteriores es que los tipos de las variables no deben ser declaradas. Cada variable define su tipo dependiendo del valor asignado. Actualmente se usa principalmente como lenguaje para generar páginas web dinámicas, por lo que suele ser ejecutado en los exploradores de internet de los clientes. (Crockford, 2008) Como suele ser ejecutado del lado del cliente, generalmente es muy ligero y requiere poco poder de procesamiento. Es por esto que se han generado frameworks para utilizarlo como lenguaje de parte del servidor.

Java

Java es otro lenguaje de programación con una gran cobertura en el mundo del desarrollo informático. Su sintaxis es similar a C y C++, pero su principal diferencia es que es un lenguaje orientado a producción y no orientado a la investigación. Es decir, mientras C y C++ fueron diseñados y desarrollados para hacer investigaciones, Java se

diseñó para ambientes de producción, por lo que tiene incorporado funciones para portabilidad y manejo automático de memoria. Se usa para aplicaciones de escritorio, aplicaciones móviles, aplicaciones web, entre otros. (Gosling et al, 2000) Aunque Java es un lenguaje de programación de más de 15 años, los sistemas más modernos aún lo utilizan por su portabilidad y por la gran cantidad de desarrolladores de Java a nivel mundial. Java sigue siendo uno de los primeros lenguajes de programación aprendidos por los estudiantes de carreras relacionadas a la informática.

2.1.2 Base de Datos

Definición

Un conjunto de información relacionada que se encuentra agrupada o estructurada. La base de datos debe permitir el acceso a los datos y debe tener información acerca de los datos que contiene. (Pérez, 2007) Casi todos los sistemas actualmente funcionan utilizando bases de datos para almacenar cualquier tipo de información. Para que una base de datos sea utilizable se necesita un motor de base de datos.

Motor de base de datos

El motor es el servicio principal que permite el almacenamiento, el procesamiento y la protección de los datos. Es la tarea del motor el controlar el acceso y el procesamiento de los datos que se contienen en la base de datos. Es importante separar al motor de la base ya que el motor es el servicio o el programa que controla la base, mientras la base de datos son los datos en sí. (Microsoft, 2012) Para escoger el motor adecuado es necesario considerar la cantidad de datos, cantidad de transacciones, lenguaje de programación con el que se va a interactuar y hardware en el que va a funcionar, entre otras características.

MySQL

MySQL es un motor de base de datos relacionales muy popular a nivel mundial. Al ser de código libre ha conseguido muchos aportes y mucho soporte en la

comunidad de desarrolladores. Este nivel de soporte que tiene lo ha convertido en uno de los motores de bases de datos esenciales para todos los desarrolladores, especialmente en su versión totalmente gratuita. Esta versión cumple con los requisitos mínimos funcionales necesitados por la mayoría de aplicaciones que requieren una base de datos, tales como seguridad, estabilidad, bajo costo, entre otros. (Suehring, 2002) La compatibilidad con MySQL se ha convertido en uno de los requisitos esenciales que deben tener los lenguajes de programación actuales para poder tener un gran impacto en la comunidad de desarrolladores.

2.1.3 Protocolos de comunicación en web

HTTP

Hypertext Transfer Protocol [Protocolo de transferencia de hipertexto] es el conjunto de reglas establecidas para la comunicación entre exploradores web. La navegación de páginas web está basado en este protocolo, por lo que la mayoría de direcciones a una página web empieza con http. Provee métodos para la comunicación como GET, PUT, POST y DELETE. Se utilizan mensajes de *request* [solicitud] y *response* [respuesta] entre los clientes y servidores que se comunican utilizando http. (Mitchell, n.d) Al ser un protocolo estandarizado y muy utilizado tiene muchas optimizaciones, por los *Web Services* también pueden hacer uso de estas optimizaciones.

HTTPS

Se trata de un protocolo para proveer mayor seguridad en el momento de utilizar comunicación basada en HTTP. Principalmente se encarga de establecer un túnel seguro entre el cliente y el servidor para asegurar que los paquetes que sean transferidos no puedan ser leídos por alguien que intente interceptar la comunicación. La comunicación por HTTPs suele usar un puerto diferente al HTTP, que usa el puerto 80 por lo general. (Rescorla, 2000) En esencia la funcionalidad y los métodos (GET, POST, PUT y DELETE) siguen funcionando de igual manera que HTTP, pero se agrega una capa de seguridad casi transparente para el desarrollador y para los clientes.

2.1.4 Framework para *Web Services*

Definición

Los Web Services previamente mencionados deben ser ejecutados por alguna aplicación que corra en un servidor web. La lista de posibles opciones para escoger la plataforma de estos Web Services es muy amplia. Se puede utilizar cualquier servidor de web y exponer los Web Services utilizando cualquiera de las posibles tecnologías para páginas web que existe. Se debe considerar las características principales como la complejidad de desarrollo, el costo final y la carga en el servidor.

NodeJS

JavaScript siempre fue un lenguaje de programación orientado a exploradores web por lo que se fue optimizando para ser muy ligero ya que era ejecutado en el cliente, el cual suele contar con relativamente pocos recursos de procesamiento. NodeJS aprovecha la estructura ligera que tiene JavaScript y lo ejecuta en el lado del servidor. Mantiene el bajo consumo de recursos que caracteriza a JavaScript pero aumenta su seguridad al ser ejecutado por completo en el servidor. NodeJS es un *framework* diseñado para desarrollar aplicaciones escalables. (Kiessling, 2014) La curva de aprendizaje de JavaScript es baja y NodeJS no tienen ningún costo para su uso.

2.2 Metodología

2.2.1 Tecnologías Investigadas

Web Services

El crecimiento del internet y del alcance que este tiene ha permitido que diferentes tareas y procesos puedan empezar a dividirse en diferentes equipos a miles de kilómetros de distancia. Un computador en otro país puede estar encargado de proveer una lista de datos, mientras que otro computador puede estar encargado de publicar más datos a esta lista y un tercer computador puede necesitar leer estos datos para algún procesamiento. Todos estos computadores solo pueden comunicarse por medio del Internet. Un *Web Service* es un método utilizado para que dos distintos programas de computador se comuniquen utilizando el Internet como medio. Esta metodología tuvo un

gran crecimiento, especialmente desde el surgimiento de las aplicaciones móviles. Es por esto que se desarrollaron algunos estándares para el desarrollo de estos *Web Services*.

Uno de los principales estándares es REST, acrónimo para *Representational state transfer*. Este estándar se generó como una abstracción de la arquitectura existente del protocolo HTTP 1.0 y del *World Wide Web*, también conocido como la Red informática mundial. Los *Web Services* basados en REST cuentan con mucha libertad para la implementación de estos, con tal de que se adhieran a las limitaciones especificadas por el estándar. Entre estas condiciones a cumplir deben contar con un URI base (es decir, una dirección web base de la cual derivan todos los *Web Services*), un formato para los datos transferidos como JSON o XML y usar los métodos de HTTP (GET, PUT, POST, y DELETE).

La principal competencia para REST viene del estándar SOAP. Este estándar ha sido implementado ya por muchas infraestructuras de desarrollo. Su nombre es una abreviación de *Simple Object Access Protocol*, o Protocolo Simple de Acceso a Objetos. Es considerablemente más estricto que REST, pero a su favor tiene que ya integra muchas funcionalidades básicas como seguridad y preservación de estado entre el cliente y el servidor. Los *Web Services* SOAP suelen ser mucho más complejos y requieren de más recursos que los REST, y están orientados a aplicaciones masivas con gran cantidad de demanda. Por lo general la curva de aprendizaje necesaria para implementar *Web Services* SOAP es mucho más lenta que la de REST.

En este caso la aplicación no requiere de tanta complejidad ni funcionalidades adicionales y al ser un trabajo investigativo, la alternativa REST permite tener un mayor control sobre el desarrollo y la implementación de cada uno de los *Web Services* necesarios para esta aplicación. Además, el uso de métodos básicos en HTTP permite probar los *Web Services* usando herramientas de línea de comando sin necesidad de desarrollar un cliente exclusivo para pruebas.

Sistema operativo móvil

La aplicación nativa para dispositivos móviles debe correr en uno de los dos principales sistemas operativos para dispositivos móviles que existen actualmente, iOS o Android. Para decidir entre la plataforma móvil a ser utilizada se consideraron los requisitos que se debe tener para cada una. El desarrollo en dispositivos iOS requiere un computador corriendo Mac OSX y una cuenta de desarrollador de Apple, la cual tiene un costo de $100 al año. Además, para poder probar la aplicación se necesita de un dispositivo Apple que funcione con iOS. El lenguaje de programación usado para aplicaciones de iOS es objective-c, el cual no es muy comúnmente usado para otros fines que no sean aplicaciones iOS. Por otro lado, el desarrollo para Android puede ser en una PC o en OSX sin inconveniente alguno. Además, no es necesario tener ninguna cuenta de desarrollador para poder crear y probar la aplicación en dispositivos reales. Solo se requiere de un pago único de $25 para poder publicar aplicaciones a Google Play. El lenguaje de programación para las aplicaciones nativas de Android es Java, un lenguaje muy común que la mayoría de desarrolladores aprenden a utilizar desde muy temprano. Considerando que el costo de desarrollo en Android es mucho menor, y que se cuenta con dos dispositivos Android para poder correr las pruebas de la aplicación, la plataforma escogida es Android.

2.2.2 Diseño

Para poder llevar a cabo este proyecto se define el alcance que este tendrá. Esto se hace por medio de los casos de uso para los diferentes niveles de privilegios que puede manejar la aplicación. Es importante tener en mente que un mismo usuario puede tener diferentes privilegios para diferentes torneos. Es decir, un usuario puede ser administrador de torneos para un torneo, digitador de otro torneo y ser seguidor de un tercer torneo. Los casos de uso ayudarán a definir concretamente el alcance de la aplicación, su funcionalidad y lo que se puede esperar como producto final.

El siguiente paso es definir la estructura de la base de datos. Es importante seguir el proceso de normalización de la base de datos para asegurar que la base

mantenga su integridad y consistencia. Para ofrecer la funcionalidad sin conexión permanente al internet, el dispositivo debe contar con una base de datos local. Esta base de datos solo va a duplicar los datos del servidor central que sean relevantes para el usuario. Además, va a contener los torneos que se creen en el dispositivo y que el usuario escoja no publicar al internet. Estos torneos pueden ser creados por usuarios que solo desean tener la funcionalidad para un torneo corto, como un torneo en videojuegos o un torneo relámpago de pocos partidos. De esta manera la aplicación se podrá ejecutar (con menos funcionalidades) cuando no cuente con una conexión a internet. El diseño de esta base secundaria es importante ya que debe mantener algún tipo de mapeo o de relación a la base de datos del servidor central, para que los identificadores utilizados sean universales entre los diferentes clientes que hagan uso de esta base.

Una vez definida la base de datos se procede a diseñar la estructura de *Web Services* que serán necesarios para poder interactuar con los datos de la base de datos. Una de las consideraciones más importantes es escoger una estructura de *Web Services* que sea escalable y reusable. Esta es una primera versión del proyecto y a futuro hay muchas funcionalidades más que se podrían agregar, además de diferentes clientes que se podrían desarrollar usando los *Web Services* ya existentes. Una estructura escalable permite agregar funcionalidades fácilmente y sin bajas en el desempeño. Para poder sacar provecho a las nuevas herramientas emergentes basadas en *Web Services*, es importante que estos se atengan los estándares establecidos. Así, futuros desarrollos de nuevos clientes podrían ser más eficientes en tiempo y costo.

A partir de este punto empieza la etapa de diseño y desarrollo de código. El uso de una herramienta de control de versiones para el código es esencial. Esto facilita el respaldo del código y automatiza la creación de versiones con el tiempo. Si se desea consultar anteriores versiones del código, estas herramientas ofrecen varias soluciones.

2.2.3 Desarrollo

El primer *Web Service* a ser implementado debe ser el de Login, ya que todas las interacciones con los *Web Services* requieren autenticación de usuario. Sin pasar por el Login, el resto de los *Web Services* pasarían a ser inutilizables. En este componente, la seguridad es crucial. El login se encarga de generar el método de autenticación que toda la aplicación usará, por lo que un login mal diseñado podría comprometer la integridad de todos los datos almacenados en la solución. Luego sigue el *Web Service* para las consultas de Torneo. Se escogió proceder con los torneos ya que estos son el núcleo de la aplicación y toda su funcionalidad está basada en los ellos. Siguen los equipos ya que pueden existir independientemente de los partidos, pero los partidos necesitan que existan los equipos que los van a jugar. Luego las jornadas ya que, en sí, son agrupaciones de partidos. Finalmente se implementa el *Web Service* de partidos. Se escogió esta secuencia para poder ejecutar pruebas mientras se avanza con el desarrollo de los *Web Services*. El último *Web Service* a desarrollar debe ser el encargado de la sincronización con las bases de datos de los dispositivos locales, ya que para su correcto funcionamiento deben estar definidos los demás *Web Services.*

Cuando estén finalizadas las pruebas de los *Web Services* se procede al desarrollo de la aplicación de Android. Antes de escribir el código, debe estar completamente definida la estructura que tendrá la aplicación, así como el diseño y toda la funcionalidad. La primera parte de la aplicación que se debe estructurar es la de la base de datos local, la cual ya fue diseñada previamente. A partir de esto se debe programar el módulo de Login para conectarse a la base central y autenticarse en ella. El siguiente paso es implementar la funcionalidad de sincronización entre las bases de datos, ya que es la única manera de comprobar que la base de datos local se puede mantener sincronizada de manera adecuada con la base de datos central.

Después de comprobar que el código se conecte e interactúe de manera correcta con el servidor central se pasa a implementar la interfaz gráfica de la aplicación, empezando con el módulo de Login. Luego sigue el módulo de consulta de torneos,

seguido por los módulos de consulta de equipos, jornadas y partidos, en ese orden. Finalmente se implementa el módulo de generación de partidos, ya que se pueden usar todos los módulos anteriormente implementados para comprobar su correcto funcionamiento. Por último se procede a implementar el módulo de la tabla de posiciones ya que depende de todos los módulos anteriores.

Una vez completada la aplicación nativa de Android es necesario correr pruebas a nivel local y a nivel del servidor. Primero se debe comprobar que el la funcionalidad local sin conexión sea correcta. Luego, la interacción con torneos creados en la aplicación y publicados al servidor central debe ser correcta. Finalmente se debe comprobar que se respeten los permisos en la aplicación móvil sobre los torneos creados en la web como los torneos creados en el dispositivo y publicados.

Estando completa la aplicación y comprobado su funcionamiento se procede a publicar el instalador en la página de la aplicación web.

2.3 Revisión de Literatura

Desde la aparición de los teléfonos inteligentes, el mundo revolucionó su forma de operar a diario. Esto abrió el campo a los desarrolladores para crear aplicaciones móviles. Una aplicación en un celular tiene muchas ventajas sobre las aplicaciones de escritorio. Una de las grandes ventajas es la portabilidad, “Usage anytime, anywhere” [Uso en cualquier momento, cualquier lugar] (Coburn, 2011) Las aplicaciones de teléfonos móviles tienen grandes aplicaciones en el campo, libre de las ataduras de los cables y las fuentes de poder requeridas por los computadores de escritorio. Además, los dispositivos móviles inteligentes están superando a las PCs en la población mundial. “By the end of this year [2013], 6% of the global population will own a tablet, 20% will own PCs, and 22% will own smartphones.” [Al finalizar este año [2013], 6% de la población global tendrá una Tablet, 20% tendrá una PC y 22% tendrá un teléfono inteligente] (Heggestuen, 2013) Actualmente, las plataformas móviles que dominan el mercado son aquellas que usan el sistema operativo Android o iOS. Sin embargo, la cobertura del

mercado de los dispositivos Android es mucho mayor, teniendo aproximadamente 79% del mercado mientras iOS tiene el 15%. (Fingas, 2014) Esto se debe en parte a que los costos asociados con los dispositivos Android en general son menores a comparación con los costos asociados con iOS, para usuarios y para desarrolladores. Es por esto que el desarrollo para Android provee un mayor alcance en el público a menor costo.

La interacción de dispositivos móviles con aplicaciones centrales en el internet se hace a través de Web Services. Estos Web Services proveen la comunicación entre dos dispositivos a través del Internet. Por el momento existen dos estrategias prevalentes para el diseño de una arquitectura de Web Services, los cuales son SOAP y REST. (Flanders, 2009) SOAP significa Simple Object Access Protocol [Protocolo de acceso simple a objetos]. Esta arquitectura establece un protocolo, o conjunto de reglas, para la comunicación entre dispositivos utilizando XML como lenguaje y estructura de transmisión de datos. Una estructura SOAP debe definir un Web Services Definition Language [Lenguaje de definición de servicio web] para describir los servicios que provee y otras especificaciones que requiere el servicio para ser utilizado. (Dhingra, 2013) Suele ser usado para estructuras de aplicaciones grandes con gran complejidad de código por lo que su curva de aprendizaje y tiempo de programación suele ser mayor. (Flanders, 2009) Por el otro lado está REST, o Representational State Transfer [Transferencia de estado representacional]. REST no presenta un protocolo sino una estandarización de arquitectura en la que la información se accede por medio de un identificador único de un recurso. Si se utiliza sobre HTTP (Protocolo de transferencia de hipertexto) se usan las funciones nativas de HTTP (GET, POST, PUT y DELETE) para declarar diferentes tipos de interacciones con los datos. La dificultad de implementación de estos Web Services suele ser menor que al usar SOAP, especialmente en aplicaciones de baja complejidad y de transacciones sencillas. (Dhingra, 2013) Los servicios REST deben mantenerse sin estado, es decir, no mantienen una conexión o información de sesión en si. La ventaja de esto es que un reinicio de los servidores mantiene funcionalidad completa de las

interacciones con los servicios REST, mientras que los servicios SOAP tendrían que seguir el proceso de establecer una sesión entre el cliente y el servicio. Una solución REST también es fácilmente escalable por el hecho de no mantener el estado de la conexión. Un conjunto de servidores con un dispositivo de balanceo de carga permitiría que el servidor más adecuado maneje cada llamada a estos servicios. El cliente no siente diferencia si en cada llamada un servidor diferente le responde, ya que no existe información de la sesión almacenada en estos servidores. (Dhingra, 2013)

Capítulo 3 – Tecnologías e Implementación

3.1 Tecnologías utilizadas para el desarrollo de los *Web Services*

Primero se hizo un análisis de la tecnología en la que funciona actualmente Tourngen Web, considerando que se desea mantener la compatibilidad entre Tourngen Web y la aplicación a ser desarrollada. La versión web corre en un servidor con Ubuntu 12.10 y utiliza como motor de base de datos MySQL 5.5.37. Además utiliza Django 1.6 como *framework*. Es importante considerar el *framework* de la aplicación web para poder asegurar la compatibilidad entre procesos básicos como la autenticación de los usuarios con los sistemas.

El desarrollo ocurrió en dos sistemas de manera a veces simultánea, un computador OSX y un computador Windows 8. En el sistema OSX se utilizó TextWrangler como editor de código mientras que en Windows 8 se utilizó notepad++. Para mantener la consistencia del código y poder tener un control de versiones adecuado, se creó un repositorio en GIT con el código de estos *Web Services*, el cual puede verse a través de https://github.com/Dres90/TourngenWS.

Los *Web Services* son escritos usando el lenguaje JavaScript. Como se explica en la sección 2.1.1, este suele ser un lenguaje para ejecutar código en el cliente, usualmente un explorador web. Gracias al *framework* NodeJS se puede ejecutar el código JavaScript del lado del servidor. Es decir, toda la lógica es ejecutada por el servidor. Al ser un lenguaje orientado para clientes, su carga sobre el procesador es ligera, por lo que usa los recursos del servidor de manera muy eficiente. La versión de NodeJS usada en este proyecto es la 0.10.26. Este *framework* cuenta con una gran cantidad de paquetes desarrollados por la comunidad para simplificar o aumentar la funcionalidad que posee. El módulo de NodeJS instalado para generar un servidor web y poder exponer los *Web Services* al internet se llama Express 4.0.0. Se debieron instalar 4 paquetes adicionales para completar la funcionalidad necesaria de los *Web Services*.

El paquete conocido simplemente como mysql (igual que el motor de base de datos definido en la sección 2.1.2) permite establecer una conexión a una base de datos MySQL y realizar consultas, transacciones, consultas en masa y autenticación. Por defecto, las llamadas a las bases de datos son asíncronas, es decir, no bloquean el proceso principal hasta obtener la respuesta de la base de datos. Esto en la mayoría de casos puede significar una mayor eficiencia ya que el código no se detiene hasta obtener una respuesta, sin embargo en este caso si es necesario que el código se detenga hasta que la base de datos responda ya que la respuesta que envían los Web Services deben incluir los datos de la base.

En respuesta a este inconveniente se instaló el paquete Q. Este es utilizado en conjunto con mysql para el manejo de llamadas a base de datos para que las llamadas a la base de datos no sean asíncronas y el código se ejecute en el orden deseado.

Aparte se instaló un paquete llamado validator 3.6.0. La funcionalidad ofrecida por este es la de hacer validaciones de cadenas de texto. En este caso se utilizó para verificar que los correos electrónicos ingresados son direcciones válidas.

Finalmente se instaló body-parser 1.02, un módulo de NodeJS que facilita la interacción del código con los diferentes componentes de los requests y responses http que maneja la aplicación. Este módulo permite leer el cuerpo, los parámetros del URL y todos los demás componentes del request como si fueran propiedades de un objeto de JavaScript. De igual forma permite escribir en los diferentes elementos del response como se modifican las propiedades de un objeto.

3.2 Desarrollo de los *Web Services*

3.2.1 Análisis de la base de datos actual

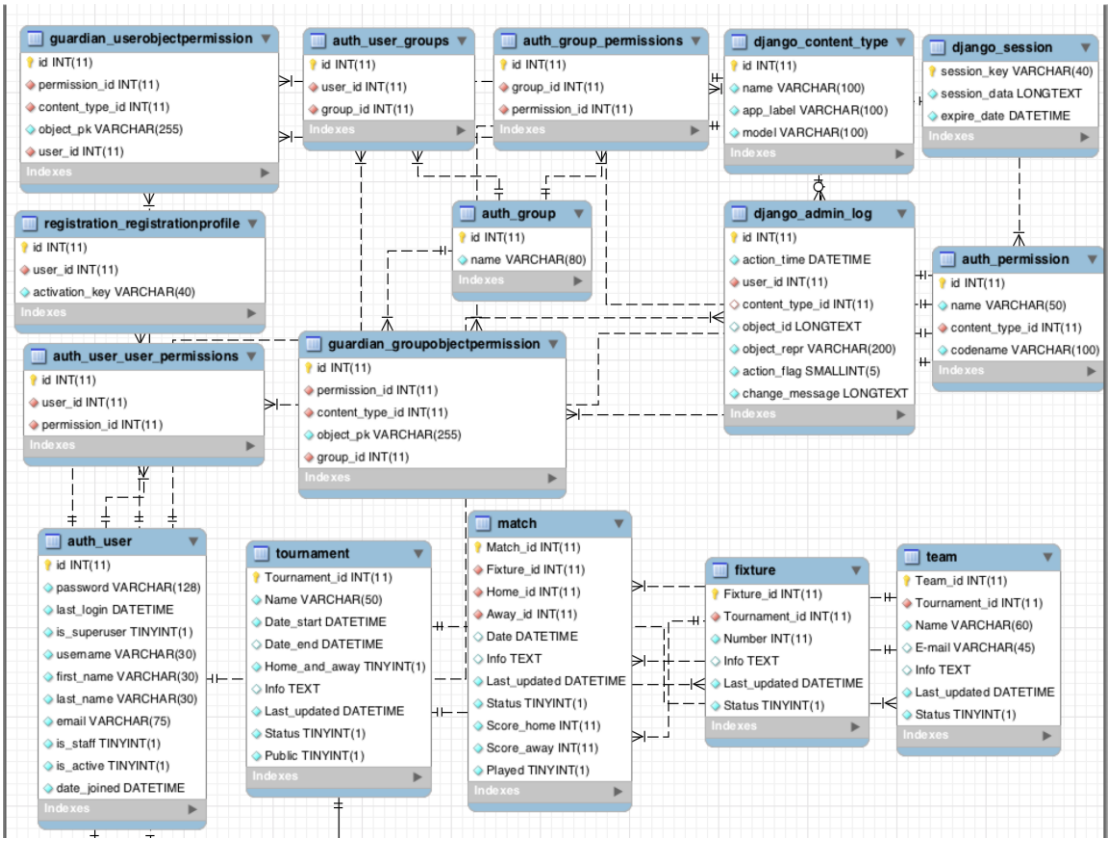

Figura 3.1: Diseño actual de la base de datos de Tourngen Web

Para poder garantizar la compatibilidad entre los dos sistemas se debe analizar la base de datos del sistema en producción. Django aumenta bastantes tablas para el funcionamiento de la aplicación, lo cual aumenta la complejidad del análisis. Es por esto que el análisis debe centrarse en la funcionalidad que requieren los *Web Services* y las tablas que serían involucradas.

Tabla	Contenido	Uso
Tournament (Torneo)	Identificador Nombre del torneo Fecha de inicio Fecha de finalización Ida y vuelta Información adicional Última modificación Estado Público	Contiene la información sobre los torneos que son creados en la aplicación.
Team (Equipo)	Identificador Torneo al que pertenece Nombre del equipo Correo electrónico Información adicional Última modificación Estado	Contiene la información sobre los equipos participantes en los torneos.
Fixture (Jornada)	Identificador Torneo al que pertenece Número de jornada Información adicional Última modificación Estado	Contiene la información básica de las jornadas que componen a los torneos.
Match (Partido)	Identificador Jornada a la que pertenece Equipo local Equipo visitante Fecha Información adicional Última modificación Estado Goles local Goles visitante Jugado	Contiene la información de los partidos del torneo. Los componentes más importantes son los equipos que juegan y el resultado del partido. Estos valores son usados para la generación de la tabla de posiciones.
auth_user	Identificador Contraseña (hash+salt) Último inicio de sesión Nombre de Usuario Nombre Apellido Correo electrónico Estado Fecha de registro	Contiene la información de los usuarios del sistema. Se usa para la autenticación.

Tabla 3.1: Tablas importantes de la base de datos

Las tablas principales son torneo (tournament), partido (match), jornada (fixture) y equipo (team). Estas tablas contienen los principales datos que manipulan ambas aplicaciones. Estas tablas no requieren de modificación alguna. La principal tabla generada por Django que afecta a los *Web Services* es la de auth_user. Esta tabla contiene (entre otros atributos) el nombre de usuario, un hash de la contraseña y la

información básica del usuario. Esta tabla es esencial para la autenticación y la creación de usuarios dentro de la aplicación.

Para el manejo de roles y privilegios, Django utiliza Django-guardian, un módulo adicional para asignar privilegios a los usuarios sobre un contenido específico. Esto es ideal para la funcionalidad de los diferentes privilegios que pueden tener los usuarios en diferentes torneos. Sin embargo, haciendo un análisis de la aplicación web se descubrió que tiene un error ya que no permite asignar diferentes privilegios a un mismo usuario. Por lo tanto, se deben crear usuarios distintos cada que se desea asignar un privilegio. Esta no es la funcionalidad deseada por la aplicación por lo que se deben hacer ciertas modificaciones a la base de datos.

3.2.2 Modificaciones a la base de datos

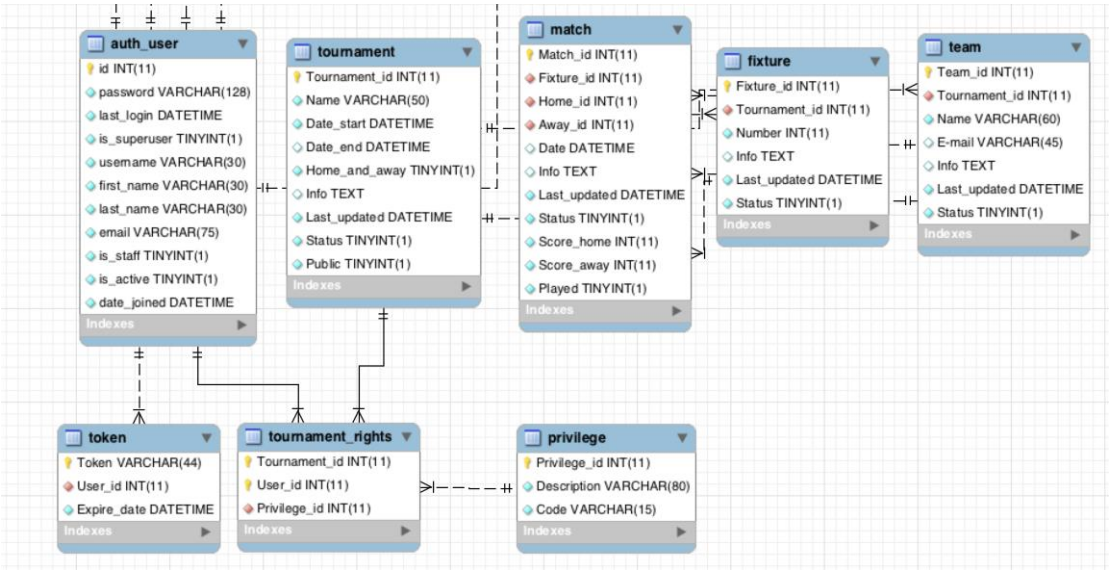

Figura 3.2: Tablas utilizadas por los *Web Services*

Las cinco tablas mencionadas anteriormente son usadas por los Web Services sin necesidad de alguna modificación. Sin embargo, se agregan tres tablas nuevas, las que permiten cumplir con toda la funcionalidad deseada por la aplicación a ser desarrollada.

Tabla	Contenido	Uso
Token	Token Usuario Fecha de expiración	Relaciona los tokens almacenados con el usuario correspondiente.
Tournament_rights (derechos de torneo)	Torneo Usuario Privilegio	Contiene el privilegio que tiene un usuario con un torneo.
Privilege (privilegio)	Identificador Descripción Código	Es una tabla estática, contiene descripciones para los diferentes niveles de privilegios disponibles.

Tabla 3.2: Tablas nuevas para los *Web Services*

La tabla token se utiliza para almacenar una cadena de caracteres que identifica a un usuario conectado a un dispositivo. Esto es necesario para que el dispositivo móvil no tenga que almacenar el usuario y la clave ya que tendría que mandar esto en cada llamada a un *Web Service*. Al consumir el *Web Service* inicial de Login enviando el nombre de usuario y contraseña, se genera el token para identificar al usuario en las siguientes llamadas. El uso de tokens permite tener al mismo usuario conectado desde diferentes dispositivos de manera simultánea y aumenta la seguridad de las credenciales del usuario ya que no se usan en cada llamada y la contraseña no es almacenada en ninguna parte.

La tabla privilegio (privilege) contiene un identificador y una descripción de los diferentes niveles de privilegios en la aplicación. Estos identificadores son usados en la tabla permisos de torneo (tournament_rights). Esta tabla contiene el identificador del usuario, el identificador del torneo y el privilegio que tiene este usuario sobre ese torneo específico. Al utilizar el torneo y el usuario como claves primarias, no permite que haya más de un privilegio para la misma combinación de usuario y torneo, manteniendo la consistencia de la aplicación. En caso de que no exista una entrada para un usuario y un torneo en esta tabla, se considera que el usuario puede ver el torneo y sus resultados solo si es un torneo público. En caso contrario se considera que no tiene ningún privilegio sobre el torneo y no puede ni ver su existencia.

La tabla privilegios es esencial para completar la funcionalidad deseada de la aplicación Tourngen. Uno característica esencial en la aplicación es que uno puede tener

diferentes roles sobre los torneos utilizando el mismo usuario. Por ejemplo, el usuario A puede ser administrador del torneo 1, digitador del torneo 2 y solo ver resultados del torneo 3. Actualmente, Tourngen Web no permite esta funcionalidad ya que se debe crear un usuario nuevo para cada privilegio que se desee asignar a un usuario. Con esta modificación a la base de datos, la aplicación garantiza que un mismo usuario puede tener diferentes privilegios para diferentes torneos. Esta funcionalidad se debe adaptar en la interfaz web en futuros desarollos.

3.2.3 Arquitectura

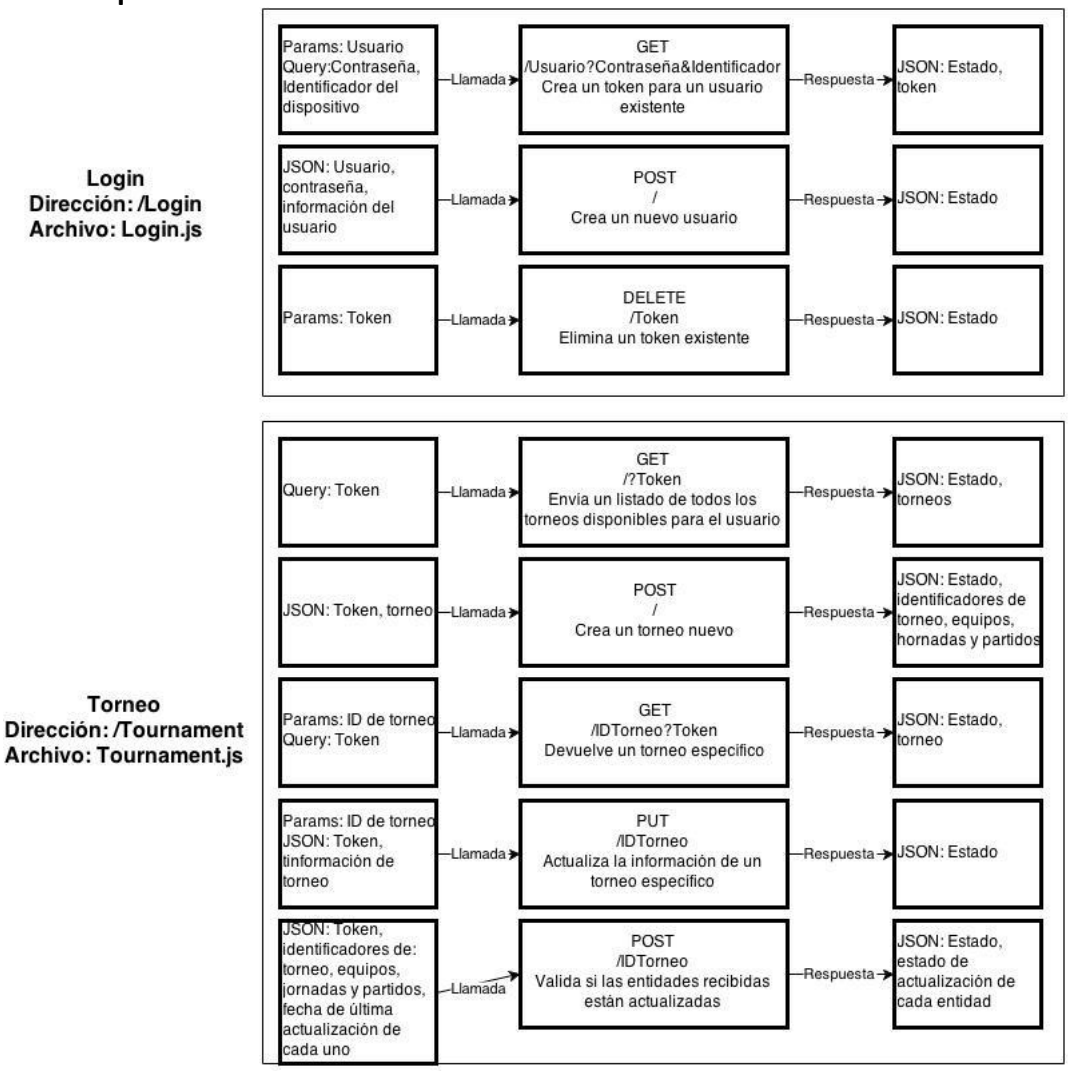

Figura 3.3: Diagrama de diseño de los Web Services 1

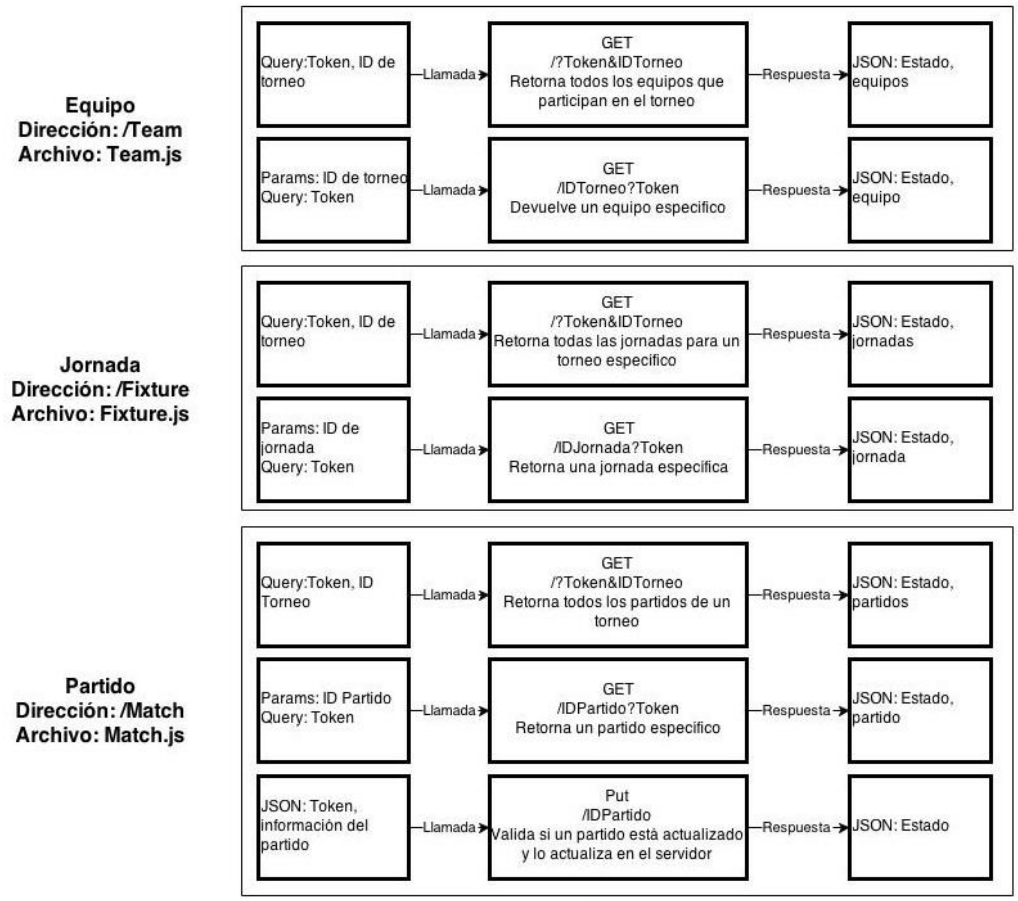

Figura 3.4: Diagrama de diseño de los Web Services 2

Para el funcionamiento de los *Web Services* se diseñó la arquitectura mostrada en las figuras 3.3 y 3.4. El código fue separado en varios archivos .js para mantener simplicidad a la hora de leer el código. Se creó un archivo por cada entidad principal (torneo, equipo, jornada y partido) que contiene el código para los *Web Services* correspondientes a cada una de esas entidades. Además, se creó el archivo login.js, responsable de los *Web Services* encargados de la autenticación del usuario y del manejo de token.

Para poder hacer cambios esenciales en la configuración principal de la aplicación se creó un archivo connect.js. En este archivo se encuentran las credenciales para la conexión a la base de datos, la ubicación de la base de datos, los certificados a ser usados para poder establecer conexiones HTTPS, y el puerto en el que van a correr los *Web Services*.

Otro archivo generado es utilities.js, el cual contiene funciones requeridas por varios de los *Web Services*, como llamadas a la base de datos o dar formato a las fechas. Este archivo se creó para poder reusar código de manera más eficiente.

Finalmente, todos estos archivos se unen por el archivo app.js. Este archivo es el encargado de iniciar los *Web Services* utilizando las configuraciones establecidas. Esencialmente, el archivo que se ejecuta es app.js y este lee de todos los demás archivos para poder levantar la aplicación.

3.2.4 Desarrollo

La seguridad en el proyecto se estableció desde el principio, al levantar el servidor web como un servidor que utiliza https. Por ser un servidor de desarrollo, se utilizó un certificado de seguridad firmado por el desarrollador. Es decir, es un certificado que no es reconocido por ninguna autoridad de certificados, por lo que la mayoría de browsers y clientes arrojarían un error al intentar establecer una conexión segura. Este error puede ser ignorado por la mayoría de clientes para pruebas mientras se termina el desarrollo y se sigue el proceso necesario para obtener un certificado firmado oficial. Una vez las aplicaciones estén listas para un ambiente de producción se procederá a conseguir un certificado reconocido por una autoridad legítima.

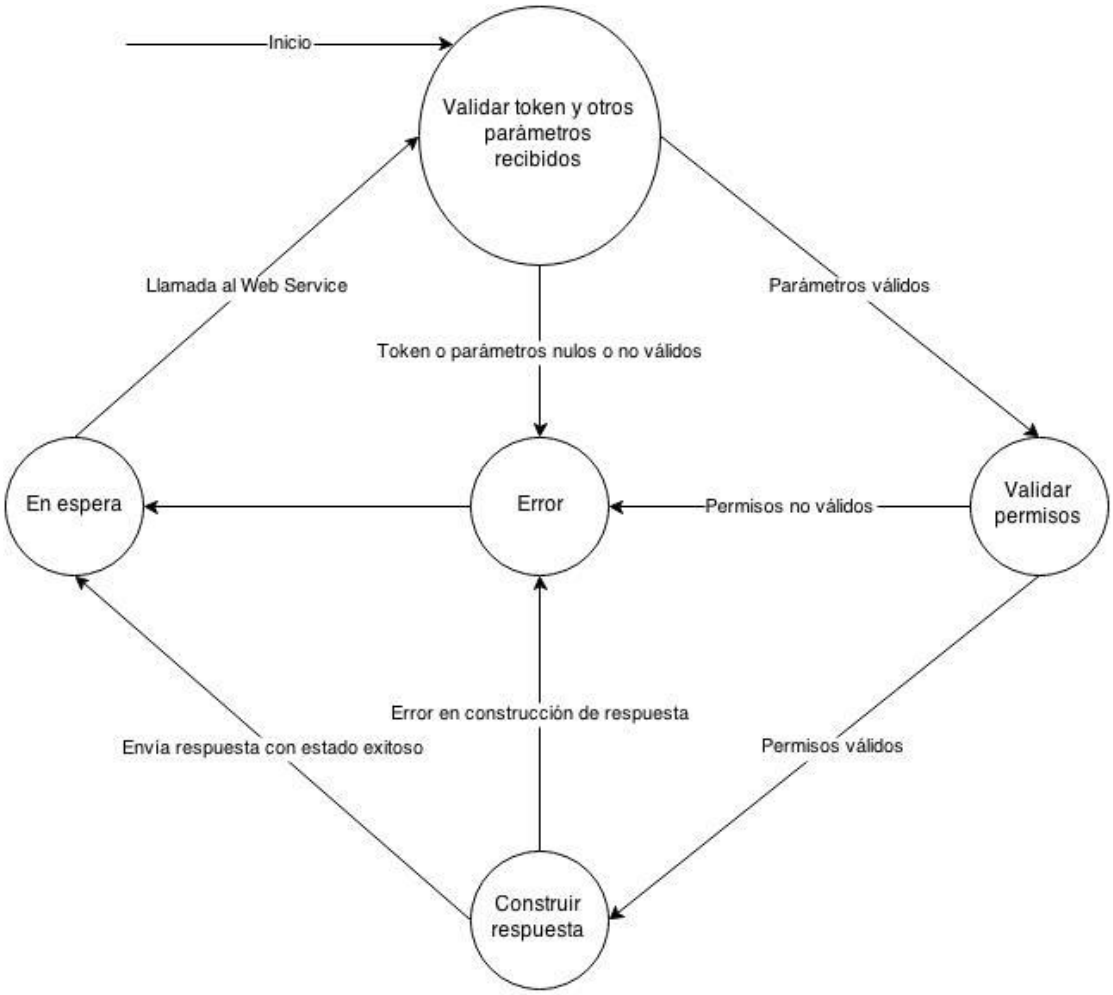

Figura 3.5: Diagrama de estado de los Web Services

El comportamiento de los *Web Services* es muy similar entre la mayoría de estos. La secuencia de decisiones lógicas que siguen en general se puede apreciar en la figura 3.5. Los *Web Services* se encuentran en reposo hasta recibir una llamada. La llamada contiene los datos necesarios para diferenciar el *Web Service* que debe ser ejecutado y los parámetros con los que se lo debe ejecutar. La mayoría de estos requieren que entre estos parámetros esté el token que identifica el usuario ya que de este dependen los privilegios que tenga. Si los parámetros son válidos y los permisos también, el *Web Service* pasa a construir la respuesta que espera el cliente. Esta respuesta varía dependiendo de la llamada pero todos contienen un mensaje de estado para identificar si la llamada fue exitosa o no. En caso de cualquier error durante la ejecución del *Web Service* se construye un mensaje de error y se envía en la respuesta a

la llamada. Sea exitosa o no la respuesta, el *Web Service* regresa a un estado de espera después de haberse ejecutado.

3.2.5 Sincronización de bases de datos

Una de las principales características de los *Web Services* es que deben ofrecer la funcionalidad para facilitar la sincronización de los datos del cliente con el servidor ya que la aplicación puede ser ejecutada sin necesidad de tener una conexión permanente al internet. Para esto todas las entidades de la aplicación deben contar con una propiedad conocida como *last_updated* (Actualizado). Esta propiedad almacena la fecha y la hora de la última actualización de esa entidad. El dato de Actualizado debe existir en la base de datos central y en el almacenamiento de los clientes ya que en base a este campo surge la comparación necesaria para sincronizar las bases de datos.

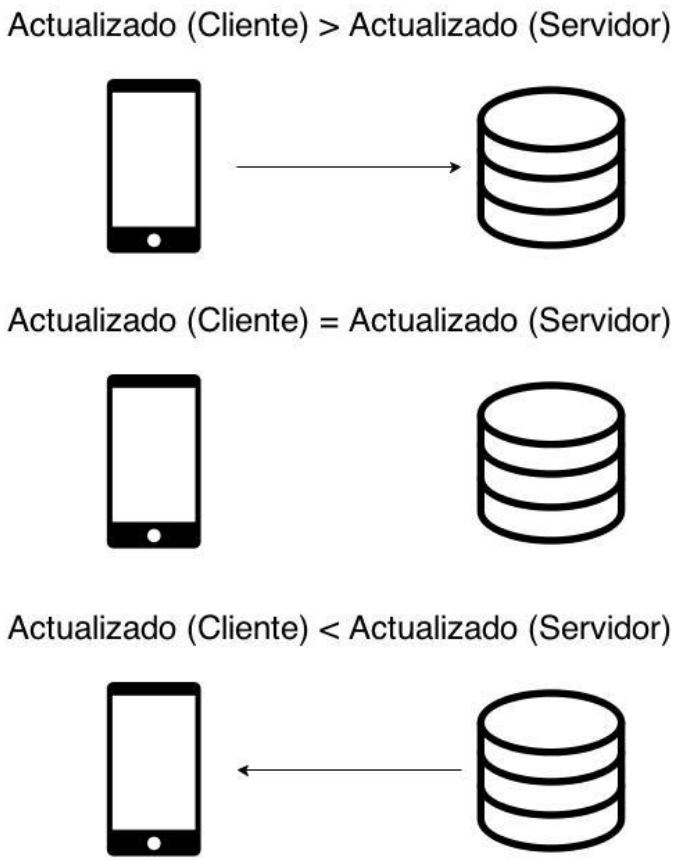

Figura 3.6: Diagrama de sincronización de entidades

El *Web Service* de sincronización recibe el campo de Actualizado de todas las entidades del torneo almacenadas en el dispositivo del cliente y las compara con el

campo de Actualizado de las entidades correspondientes en el servidor. Usando esta información se informa al cliente el estado de sincronización de cada una de las entidades para que proceda a actualizar la base de datos correspondiente, si es necesario. La figura 3.6 muestra la comparación que hace el *Web Service* para cada entidad y la acción correspondiente que debe ejecutar el cliente. Si el Actualizado del cliente es más reciente que el Actualizado del servidor, se envía la información completa de la entidad del cliente al servidor y se almacena. En caso contrario, el cliente es informado que debe hacer una llamada GET al servidor para obtener la versión más actualizada de la entidad correspondiente. Si las fechas son iguales no se hacen cambios en ninguna de las bases. Vale recalcar que este *Web Service* respeta los privilegios del usuario. En caso de que el Actualizado del cliente sea más reciente que el del servidor, pero el privilegio del usuario no sea de administrador ni digitador, el *Web Service* envía la señal de que el cliente debe hacer la llamada GET para actualizarse con el servidor y no modifica la base de datos central.

3.2.6 Pruebas

Para probar el correcto funcionamiento de los *Web Services* se utilizó *Curl*, una herramienta de línea de comando de Linux. *Curl* permite, entre otras funciones, hacer llamadas http de los diferentes tipos (GET, POST, PUT y DELETE) a cualquier servidor y mostrar los resultados en línea de comando. Esta herramienta permite modificar el contenido de la llamada por lo que se podrían generar llamadas similares a las que harían los clientes de los *Web Services* y de esta forma comprobar que los resultados retornados por estos sean los esperados.

3.3 Tecnologías utilizadas en el desarrollo de la aplicación de Android

El desarrollo de las aplicaciones nativas para Android se lleva a cabo en Java. Para más información de este lenguaje de programación referirse a la sección 2.1.1. En este proyecto se usa el Java Development Kit (JDK) versión 1.7. Como IDE se utilizó Eclipse con el complemento Android Developer Tools (ADT). Eclipse suele ser un

ambiente de desarrollo para múltiples plataformas pero ADT especializa Eclipse para el desarrollo de aplicaciones de Android. La versión de ADT usada en el proyecto es la 22.6.2. Actualmente existe la versión beta de Android Studio, un IDE desarrollado exclusivamente para las aplicaciones de Android, pero por cuestiones de estabilidad se escoge la herramienta con mayor antigüedad y documentación disponible a la fecha.

3.4 Desarrollo de la aplicación de Android

3.4.1 Análisis de almacenamiento de datos

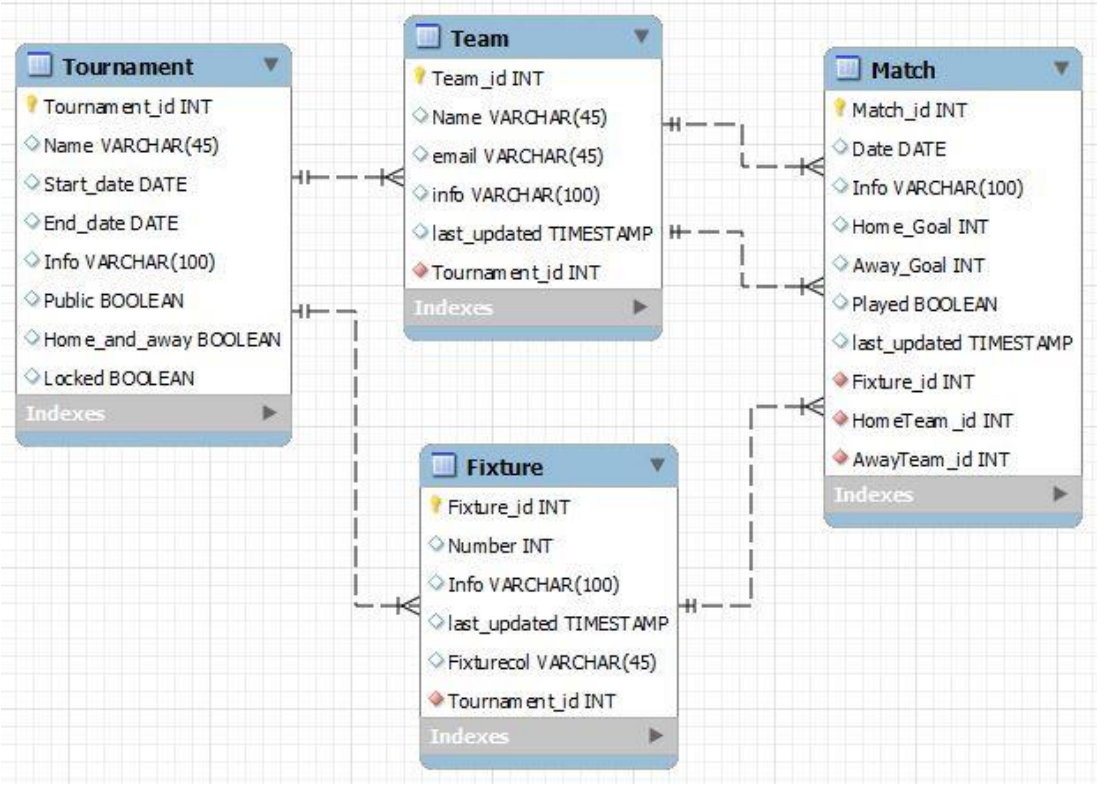

Figura 3.7: Diagrama de base de datos en dispositivo Android

La aplicación nativa para Android debe contar con alguna forma de almacenamiento de datos independiente al del servidor web ya que debe ofrecer funcionalidad sin conexión a internet, por lo que la primera opción es la de replicar la base de datos en el dispositivo Android, incluyendo solo las partes de la base de datos que son relevantes a la aplicación. Para el manejo de bases de datos, Android ofrece *SQLite*, un sistema de administración de bases de datos diseñado para dispositivos de poco poder de procesamiento. La principal diferencia de *SQLite* a la mayoría de motores de bases de datos es que *SQLite* no corre como un proceso aparte o independiente, sino

que es parte del proceso de la aplicación que levanta la base de datos. De esta forma solo consume recursos cuando se necesita.

La otra alternativa es crear clases basadas en los datos que deben ser almacenados y serializarlos para almacenarlos como archivos físicos en el teléfono. Esto es casi equivalente a implementar un *ORM (Object Relational Mapping)*, un sistema usualmente usado para simplificar la interacción con bases de datos al tratar los datos en la base como objetos. Sin embargo, en esta alternativa no se almacenarían en una base sino en archivos. La constante lectura de archivos del almacenamiento del disco suele ser lenta. Sin embargo, la serialización simplifica el almacenamiento de los datos ya que no se necesita escribir sentencias SQL, solo heredar de la interfaz *Serializable* de *Java*.

Figura 3.8: Diagrama UML de clases serializables para el almacenamiento de torneos

Ya que se trata de pocos tipos de objetos que deben ser almacenados en la base de datos local y que una vez que se carga un torneo no es necesario tener los demás torneos en memoria, se elije la alternativa *Serializable*. Cada torneo se almacena en un archivo en el dispositivo. Esto evita tener que escribir sentencias SQL o cargar una

base de datos. Además, trabajar con objetos es mucho más intuitivo que trabajar con sentencias SQL. Al solo tener un torneo cargado en memoria también se ahorran recursos del dispositivo. La figura 3.8 muestra la implementación de estos objetos en diferentes clases para simplificar la manipulación y el almacenamiento de los torneos.

3.4.2 Estructura de la aplicación

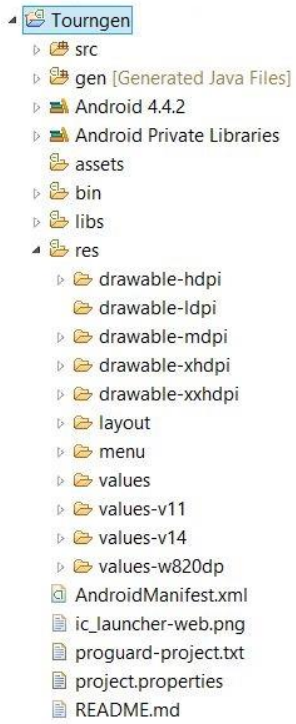

Figura 3.9: Directorios que componen aplicación de Android

Las aplicaciones de Android tienen una estructura predefinida de directorios, los cuales son generados de forma automática al iniciar un proyecto. Esta estructura se muestra en la figura 3.9. En el directorio *src* está el código fuente generado por el desarrollador para el funcionamiento de la aplicación. La estructura de archivos dentro de este directorio depende exclusivamente del desarrollador. El directorio *gen* contiene archivos generados automáticamente por el ambiente de desarrollo. Contiene algunos archivos de referencias dinámicas. No es necesario que el desarrollador manipule estos archivos. Los archivos de imágenes, videos, audio, entre otros son almacenados en el directorio *assets*. En *libs* se encuentran las librerías externas que utiliza la aplicación. La carpeta *res* contiene varias subcarpetas, pero esencialmente contiene archivos XML que

definen interfaces gráficas, animaciones o valores definidos para la aplicación. Esto puede ser usado para tener diferentes valores dependiendo de la resolución de pantalla, idioma y otras condiciones del dispositivo.

Afuera de todos estos directorios existen archivos adicionales de configuración, pero el más importante es *AndroidManifest.xml*. En este archivo se especifica la actividad inicial de la aplicación. Además, se especifican los permisos que la aplicación necesita solicitar al sistema operativo. También contiene un listado de todas las actividades de la aplicación, y en caso de que la aplicación utilice una navegación en jerarquía se usa para especificar la precedencia entre las actividades.

3.4.3 Actividades de la aplicación

En Android, una actividad es el equivalente a una vista o pantalla de la aplicación. Las actividades son la principal manera en la que la aplicación muestra información al usuario y recibe información de este. Cada actividad necesita escencialmente de dos archivos, un .XML y un .java. El archivo XML se encarga de definir la estructura estática de la actividad, es decir los componentes que se muestran al usuario, así como el órden y la ubicación de estos. Toda la funcionalidad dinámica de una actividad debe estar incluida en el archivo .java. Este archivo define el .XML que va a usar para presentar información al usuario. También define todo el comportamiento y toda la lógica que deben tener los componentes de la actividad para interactuar con el usuario. Ya que existen archivos .java que no son usados para las actividades de la aplicación, se suele incluir la palabra *Activity* en el nombre de los archivos .java que se usan para las actividades. Es una convención general pero no es obligatorio.

Archivo de Actividad	Funcionalidad
AddTeamsActivity.java	Al crear un torneo nuevo, esta actividad contiene un listado de los equipos del torneo y permite agregar o borrar equipos.
DatePickerFragment.java	Un fragmento utilizado para escoger las fechas de inicio y de fin de los torneos.
FixtureListActivity.java	Esta actividad muestra los partidos correspondientes a una jornada y contiene un filtro para mostrar los partidos de otras jornadas.
MainActivity.java	La actividad inicial de la aplicación. Por defecto la aplicación empieza en la pantalla de *Log In*, donde el usuario ingresa su nombre y contraseña o puede escoger crear un usuario nuevo. Si el usuario ya se encuentra autenticado, la aplicación se salta esta actividad y pasa directamente al listado de torneos.
MatchActivity.java	Esta actividad muestra el detalle de un partido en específico. Además si el usuario tiene los permisos necesarios para cambiar los resultados, esta actividad se encarga de mostrar los controles necesarios para que el usuario pueda editar y mandar los resultados al servidor.
MatchListActivity.java	Contiene un listado de todos los partidos que forman parte de un torneo, ordenados por las jornadas a las que corresponde cada uno.
NewMatchesActivity.java	Al crear un torneo nuevo, esta actividad muestra el listado de partidos y jornadas generadas automáticamente por la aplicación. También muestra un botón que permite al usuario volver a generar un cronograma distinto hasta que el usuario esté satisfecho con el resultado.
NewTeamFragment.java	Este fragmento contiene los campos necesarios para ingresar la información correspondiente a un equipo nuevo para agregarlo al listado de equipos participantes en un torneo nuevo.
NewTournamentActivity.java	La primera actividad en el proceso de generación de un torneo nuevo. Esta muestra los campos necesarios para ingresar la información del torneo.
SignupActivity.java	El proceso de creación de un usuario nuevo es manejado por esta actividad, presenta los campos necesarios al usuario y envía la información al servidor.
TeamActivity.java	Esta actividad muestra la información de un equipo específico que participa en un torneo. Además de mostrar los detalles de un equipo también presenta un listado de los partidos que tiene ese equipo en el campeonato.
TeamListActivity.java	Contiene un listado de todos los equipos que forman parte de un torneo y se los presenta al usuario para que este pueda ver el detalle de cada uno.
TournamentActivity.java	Esta actividad presenta la información del torneo. Además está encargada de generar y presentar la tabla de posiciones del torneo.
TournamentListActivity.java	Presenta al usuario el listado de torneos al que tiene acceso. Además le informa si es que puede editar los resultados de ese torneo o no. También incluye la funcionalidad para cerrar sesión y sincronizar todos los torneos del usuario.

Tabla 3.3: Actividades de la Aplicación de Android

La tabla 3.3 muestra los archivos de actividades utilizados por la aplicación. Además, incluye los archivos de fragmentos. Los fragmentos son usados por ventanas de diálogos usados en diferentes partes de la aplicación. Son iguales que los archivos de actividades pero los fragmentos se consideran subcomponentes de una actividad. Una actividad puede tener varios fragmentos para mostrar diferentes interfaces sin cambiar de actividad.

3.4.4 Herramientas de la aplicación

El directorio *utils* contiene varias clases de java cuya función es diferente a las demás clases de la aplicación. Cada uno de estos archivos cumple con diferentes funcionalidades para la aplicación. Muchos de estos contienen funciones que son reusadas en varios lugares del código.

Existen dos clases *Singleton*, es decir que no pueden ser instanciadas más de una vez y solo existe un objeto de esta clase. Una de estas es *Config.java*. Este archivo contiene varios datos como el nombre de usuario y el token que identifica al usuario con los web services. Además, contiene los identificadores de los torneos que están almacenados en el dispositivo. Esta clase es serializable ya que se almacena en la memoria física del dispositivo y se carga cada que se abre la aplicación. La otra clase *Singleton* es DataHolder.java. Esta aplicación es un contenedor usado por todas las actividades para saber el torneo que se está ejecutando. Las actividades se limpian de la memoria ocasionalmente y pasar el torneo entre actividades requiere bastante código para encapsular el objeto. Cuando una sola clase tiene la referencia al torneo que se está usando facilita el acceso desde todas las actividades. El torneo se mantiene en memoria para que no haya demoras al intentar leer el torneo desde el almacenamiento del dispositivo.

Otra clase de utilidades es WSRequest.java. Esta clase simplifica el código necesario para hacer llamadas a los *Web Services*. Ya que se deben hacer varias llamadas durante el ciclo de la aplicación, esta clase permite reusar muchas líneas de

código y simplifica la sintaxis de la generación de estas llamadas. Todos los objetos de esta clase retornan un JSON con la respuesta de los web services, Además, ofrecen una función para validar si el dispositivo tiene conexión a internet. Esta se utiliza para habilitar o deshabilitar ciertas funciones de la aplicación que dependen de tener acceso al internet.

Como estas llamadas son asíncronas, las actividades deben llamarlas utilizando un hilo separado. Es por esto que las actividades que hacen estas llamadas deben tener una clase aparte encargada de hacer la llamada y manejar los resultados. Esta clase es una clase privada dentro de la actividad y extiende *AsyncTask*, una clase que provee Android para ejecutar procesos en segundo plano. Esta clase permite que el proceso a ser ejecutado no bloquee la interfaz gráfica y provee una funcionalidad sencilla para que pueda interactuar con la interfaz cuando ya se termine la ejecución del proceso. En caso de dispositivos con varios núcleos de procesador, esta clase permite un mejor aprovechamiento de los núcleos usando diferentes hilos para ejecutar tareas pesadas.

3.4.5 Generación automática de partidos

Los torneos de este tipo han existido desde hace muchos años y existen en muchos deportes, por lo que existen varios algoritmos para la generación de los cronogramas de partidos.

Figura 3.10: Paso 1 Algoritmo de generación automática de partidos

El más general es un algoritmo cíclico en el que se dividen a todos los equipos en dos grupos iguales, si existe un número impar de equipos se crea un equipo temporal

para emparejar los grupos. Para mejor representación visual, se alinean los grupos en dos arreglos horizontales, como se ve en la figura 3.10. Es en esta etapa que se tiene la primera jornada de partidos, el equipo del arreglo superior se empareja con el equipo del arreglo inferior correspondiente. En caso de tener un número impar de equipos, el equipo que sea emparejado con el equipo temporal tiene descanso en esta jornada de partidos. El emparejamiento se puede apreciar en la figura 3.11.

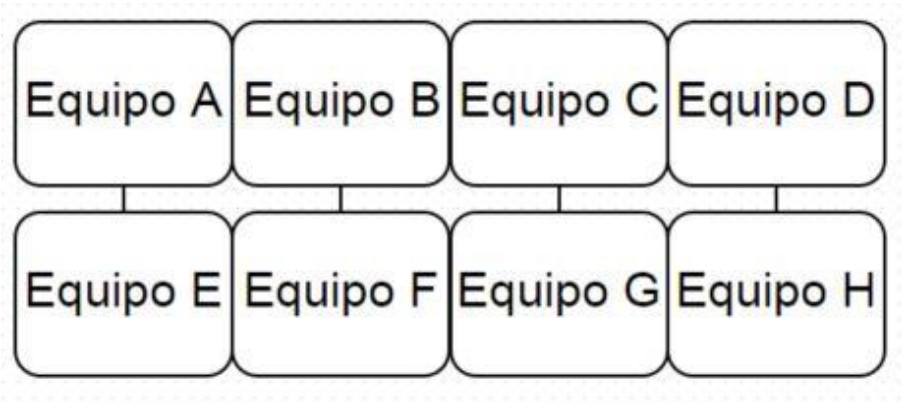

Figura 3.11: Paso 2 Algoritmo de generación automática de partidos

A partir de este punto se escoge un equipo pivote y los demás equipos son rotados en la dirección de las manecillas del reloj o en contra de estas, lo importante es que la dirección sea la misma para las siguientes iteraciones. Esta rotación se puede observar de manera más sencilla en las figuras 3.12 y 3.13 que muestran las siguientes dos iteraciones del algoritmo cíclico

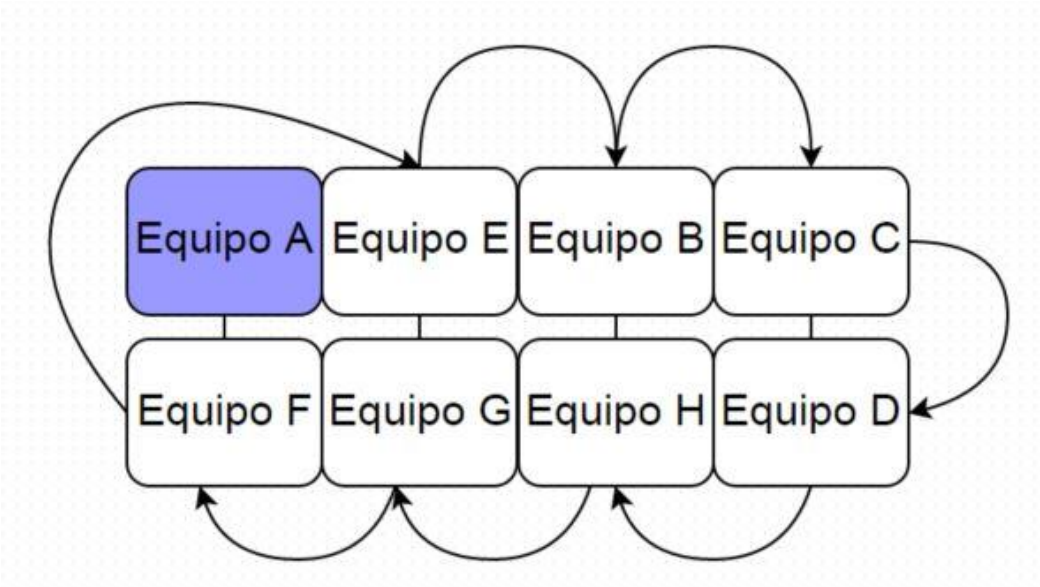

Figura 3.12: Paso 3 Algoritmo de generación automática de partidos

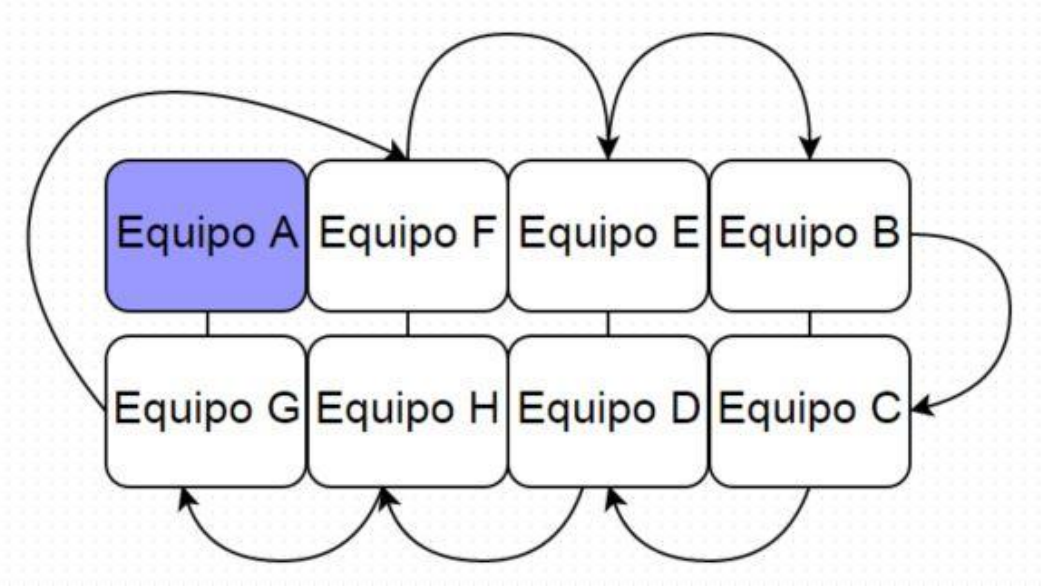

Figura 3.13: Paso 4 Algoritmo de generación automática de partidos

Una vez los equipos regresen a la posición inicial se tienen todas las jornadas necesarias para el juego. En esta etapa se pueden reordenar de manera aleatoria para que la elección del pivote no sea un claro determinante en el orden de los partidos a ser jugados. En caso de que el torneo sea de ida y vuelta, los partidos de vuelta son generados al tomar todos los partidos de ida y revertir el orden de los equipos y el orden de las jornadas, así como se ilustra en la figura 3.14.

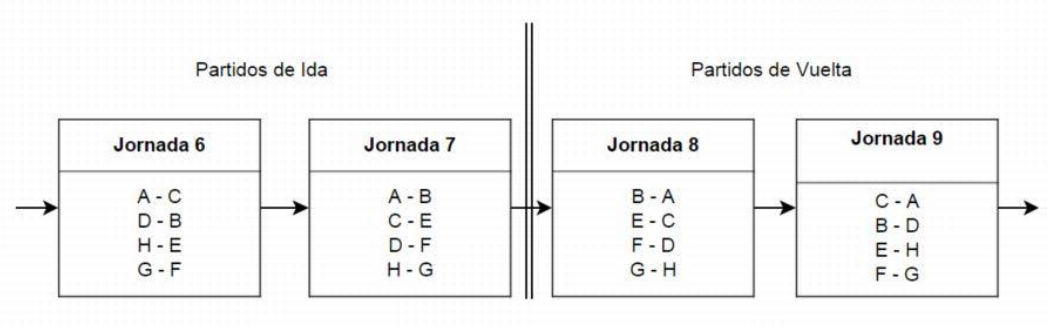

Figura 3.14: Generación de partidos de vuelta

Cuando ya es aceptado el cronograma por el usuario, este se almacena como parte del torneo. Por limitaciones de interfaz la aplicación no permite al usuario generar manualmente los cronogramas de juego ya que ingresar y modificar todos los partidos en una pantalla pequeña y táctil puede resultar incómodo y no muy intuitivo. Esta funcionalidad se puede conseguir en la aplicación Web.

Capítulo 4 – Análisis y Resultados

4.1 Metas y Actividades

Los *Web Services* se encuentran activos y funcionales, mientras la aplicación de Android está finalizada y disponible para descargar. Se cumplieron las actividades planteadas para este proyecto, las cuales son:

1. Diseño de la base de datos
2. Implementación de Web Services de Login
3. Implementación de Web Services de Torneos
4. Implementación de Web Services de Equipos
5. Implementación de Web Services de Jornadas
6. Implementación de Web Services de Partidos
7. Diseño de la base de datos para Android
8. Implementación de módulo de Android para Login
9. Implementación de módulo de Android para Torneos
10. Implementación de módulo de Android para Crear Torneos
11. Implementación de módulo de Android para Equipos
12. Implementación de módulo de Android para Jornadas
13. Implementación de módulo de Android para Partidos
14. Implementación de módulo de Android de tabla de posiciones
15. Pruebas de la aplicación
16. Distribuir aplicación

Meta	Cumplida
Investigar tecnología adecuada para los Web Services necesarios teniendo en mente escalabilidad, desempeño y complejidad, además de los costos monetarios que podrían estar involucrados.	Si
Investigar métodos para consumir Web Services de forma segura, y como se maneja el inicio de sesión en los clientes móviles de servicios similares.	Si
Implementar los Web Services de forma estructurada y documentada, tal que en un futuro sean fácilmente reusados para desarrollar otros clientes.	Si
Implementar la aplicación de Android que permita consumir e interactuar con estos Web Services, de manera que se puedan crear torneos, equipos, jornadas y partidos.	Si
Implementar la funcionalidad en la aplicación de Android consultar torneos, equipos, jornadas y partidos existentes en la base de datos central y poder interactuar con estos torneos respetando los diferentes niveles de permisos.	Si
Implementar las estrategias de sincronización sin conexión que permita que la aplicación pueda ser usada sin una conexión permanente al internet.	Si
Ejecutar pruebas del correcto funcionamiento de la aplicación y que se mantenga la coherencia de la base de datos, tanto local como central.	Si
Distribuir la aplicación por medio de una descarga directa desde la aplicación web.	Si

Tabla 4.1: Metas Cumplidas

4.2 Actividad 1 – Diseño de la base de datos

Para el funcionamiento de los *Web Services* se debieron hacer algunas modificaciones a la base de datos de Tourngen Web como se detalla en la sección 3.2.2. Estas modificaciones fueron necesarias para agregar las funcionalidades que requerían los *Web Services* y asegurar que ambos sistemas puedan seguir compartiendo la misma base de datos. Durante el desarrollo se usó una base de datos temporal para no obstruir con el funcionamiento de Tourngen Web.

4.3 Actividad 2 – Implementación de Web Services de Login

Los Web Services de Login fueron los primeros en ser implementados. Esto se hizo para asegurar desde el principio la compatibilidad y la interoperabilidad de los dos sistemas. La autenticación, la creación de usuarios y el manejo de tokens se encuentran funcionando.

Para mantener el método de autenticación que utiliza Django se debió crear un script de Python que llama a los mismos métodos que utiliza Django. Esto se hizo ya que

la implementación de los algoritmos de hash de las contraseñas era diferente entre NodeJS y Django. Para solucionar esta incompatibilidad, NodeJS llama este script pasando los parámetros necesarios y obtiene los resultados ejecutados por Python. Esta es la única funcionalidad de los Web Services que requirió código en un lenguaje de programación además de JavaScript.

Otro inconveniente que emerge con NodeJS es la dificultad en mantener código limpio y fácil de leer cuando se usan llamadas asíncronas. JavaScript, por defecto, hace llamadas asíncronas para aumentar su eficiencia, pero en algunos casos es necesario tener la respuesta de una llamada antes de continuar con la ejecución del código. Para asegurarse de que el código sea ejecutado de manera síncrona se escribe código dentro de una función conocida como callback, la cual se pasa como parámetro a la llamada a la base de datos. El inconveniente es que cuando se hacían varias llamadas en secuencia se terminaba escribiendo callbacks dentro de callbacks dentro de más callbacks, haciendo que el código sea cada vez más desordenado y más complejo de leer. Esto se conoce como Callback Hell. Si se intenta separar el código en diferentes funciones para facilitar la lectura del código, las funciones son ejecutadas antes de que la llamada asíncrona sea completada, dando errores y resultados vacíos cuando se accede a la base de datos. Es por esto que se instala el paquete de NodeJS llamado Q. Este paquete facilita el manejo de callbacks permitiendo que el código pueda ser separado en diferentes funciones, transformando las llamadas asíncronas en síncronas y de esta manera asegurándose que el código se ejecute en la secuencia deseada. Además, simplifica el manejo de los errores en las llamadas asíncronas, permitiendo capturar los errores con pocas líneas de código. El concepto inicial de Q es algo complejo y difícil de entender al principio, pero después de escribir algunas funciones usando Q se facilita mucho el trabajo y se asegura de obtener los resultados deseados de la aplicación.

4.4 Actividad 3 – Implementación de Web Services de Torneos

Los torneos son los componentes esenciales de la aplicación, por lo que los *Web Services* correspondientes son bastante robustos. La funcionalidad para consultar y actualizar torneos fue sencilla de implementar, aunque requirieron consultas algo complejas a la base de datos para asegurar un resultado acorde a los privilegios del usuario.

La funcionalidad para ingresar un torneo nuevo es más compleja ya que procesa una cantidad considerable de datos. Este *Web Service* debe interpretar el JSON recibido y validar que tiene los datos necesarios para agregar un torneo válido a la base de datos. Este torneo se compone de varias inserciones a la base de datos, por los equipos, las jornadas y los partidos. Una vez completadas las inserciones se deben capturar los identificadores de cada entidad para retornarlos al cliente ya que este los almacena para mantener consistencia y entre los datos almacenados en el servidor y los datos del cliente.

Otra funcionalidad compleja es la de sincronización de los torneos. El *Web Service* encargado debe recibir un JSON con identificadores de cada componente del torneo, así como la fecha de última actualización de ese componente para poder consultar los campos correspondientes en la base de datos. Finalmente, este retorna el JSON con los estados de los componentes indicando al cliente como debe proceder para actualizar el torneo.

Aunque son actividades de gran complejidad, estas fueron completadas satisfactoriamente,

4.5 Actividad 4 – Implementación de Web Services de Equipos

Los *Web Services* de Equipos son simples y concretos, validan los privilegios y retornan el equipo o los equipos pertinentes. Se implementaron sin problemas.

4.6 Actividad 5 – Implementación de Web Services de Jornadas

Las jornadas son partes sencillas de la aplicación ya que en esencia solo son conjuntos de partidos. Como estos componentes no ofrecen la funcionalidad de ser actualizados independientemente, los *Web Services* se implementaron de manera sencilla muy similar a los de Equipos.

4.7 Actividad 6 – Implementación de Web Services de Partidos

La primera parte en la implementación de los *Web Services* correspondientes a los partidos fue idéntica a la de los partidos y jornadas. Sin embargo, los partidos ofrecen la funcionalidad para actualizar los partidos. La complejidad de esta actividad es mayor a la que se espera ya que, para asegurar consistencia, se debe validar la última fecha de actualización del partido y actuar en respuesta a esto. Este *Web Service* debe actualizar solo si es que la última fecha de actualización recibida es igual a la fecha en el servidor. En caso de que si lo sea, el servidor se encarga de retornar la nueva fecha para que el cliente la almacene. En caso contrario, no debe hacer cambios y enviar un error para que el cliente actualice el partido con el del servidor antes de poder enviar sus cambios.

4.8 Actividad 7 – Diseño de la base de datos para Android

La base de datos de la aplicación de Android se basó en los componentes esenciales de los torneos. Al observar la simple estructura de estos componentes y concluir que las posibles consultas a la base de datos no serían nada complejas, se respaldó la decisión de almacenar en archivos para cada torneo en vez de levantar una base de datos que acapare más recursos de los necesarios.

4.9 Actividad 8 – Implementación de módulo de Android para Login

El módulo de login es lo primero que se cargar al iniciar la aplicación. Este módulo fue el primero en ejecutar llamadas a los *Web Services*, por lo que incluyó la implementación de funciones que faciliten estas llamadas para poder reusar código en las futuras actividades.

Surgió un inconveniente al implementar las primeras llamadas a los *Web Services*. Inicialmente se planeaba que el *token* y cualquier información que deba enviar el cliente al servidor esté incluida en el cuerpo del *request* y NodeJS permitió implementar los *Web Services* de esta forma. Sin embargo, el estándar de HTTP establece que todas las llamadas GET no deben contener información en el cuerpo del *request* y las librerías de Android se adherían a este principio de tal forma que la librería para hacer llamadas a *Web Services* no permite incluir un cuerpo en un *request* de tipo GET. Por esto, todos los *Web Services* de tipo GET tuvieron que ser rediseñados y modificados para que los parámetros necesarios estuvieran incluidos en el URL del *Web Service*.

Al finalizar esta actividad y hacer las modificaciones necesarias se concluyó que las llamadas a los *Web Services* funcionan correctamente.

4.10 Actividad 9 – Implementación de módulo de Android para Torneos

Como los torneos son esenciales para la aplicación, este módulo fue el siguiente en ser implementado. En esencia es solo un listado de torneos y los detalles de cada torneo, por lo que su implementación no parecía complicada. Sin embargo, fue el primer módulo que tuvo que serializar los torneos recibidos por los *Web Services* y reveló la complejidad del proceso de mapear objetos de JSON a objetos de Java. Es un proceso largo y la implementación fue bastante repetitiva pero una vez finalizado, los torneos se serializan correctamente.

4.11 Actividad 10 – Implementación de módulo de Android para Crear Torneos

Esta funcionalidad es una de las utilidades más importantes que ofrece la aplicación, además que era necesaria para probar el resto de módulos usando torneos generados localmente además de los torneos generados por el aplicativo web. Dentro de este módulo se incluyó la implementación del algoritmo de generación de partidos.

Durante la implementación de este módulo surgió la problemática para diferenciar torneos locales a los que se sincronizan con el servidor web, ya que la aplicación permite crear torneos que solo se mantienen el dispositivo. Para poder

diferenciar estos torneos, se les asigna un identificador con números negativos, lo que no es un identificador válido para el servidor. De esta forma se diferencian los torneos locales a los torneos remotos.

4.12 Actividad 11 – Implementación de módulo de Android para Equipos

El componente de equipos muestra poca información al usuario ya que los equipos no tienen mucha información asignada a sí mismos. El proceso complicado fue la generación de un listado de los partidos exclusivos para un equipo. Para este proceso se usó una función que retorna todos los partidos de un torneo y al momento de presentarlos se filtran solo los partidos en los que jugó el equipo en cuestión. La interfaz usada para presentar el listado de partidos fue reusada en los módulos de jornadas y partidos para mantener un diseño consistente en los diferentes módulos.

4.13 Actividad 12 – Implementación de módulo de Android para Jornadas

El módulo de jornadas presenta el listado de partidos agrupado por la jornada correspondiente. Es por esto que se reusó la interfaz del listado de partidos del módulo de equipos y simplemente se agregó un filtro en la interfaz para poder seleccionar la jornada que va a ser mostrada. El único dato relevante para el usuario es el número de jornada que se está viendo, y este puede mostrarse sin problema en el filtro.

4.14 Actividad 13 – Implementación de módulo de Android para Partidos

La funcionalidad que separa a la aplicación del Tourngen Web se ve principalmente en el módulo de partidos. Ningún componente de los torneos puede ser modificado en la aplicación excepto por los partidos. Es por esto que este módulo implementó funcionalidad adicional a los demás módulos. Esto se logró presentando dos interfaces diferentes al usuario, dependiendo de los privilegios que se tengan sobre el torneo. Cuando se intentan enviar cambios al servidor se debe procesar la respuesta para revertir los cambios o aceptarlos, dependiendo de la última fecha de actualización

del partido. Esto permite siempre trabajar con la última actualización de todos los partidos.

4.15 Actividad 14 – Implementación de módulo de Android de Tabla de Posiciones

La tabla de posiciones es esencial para determinar los ganadores y perdedores en un torneo. Considerando la funcionalidad fuera de línea, este proceso se debe hacer completamente local. El proceso para el cálculo de posiciones es algo largo y debe hacer iteraciones sobre toda la lista de partidos jugados. Para facilitar la lectura del código y simplificar la implementación de este proceso se creó una clase privada llamada *Position*, Esta clase provee métodos que ayudan en el cálculo de la tabla de posiciones. Además, implementa la interfaz *Comparable* para que sea fácilmente ordenado en un arreglo y desplegado de forma ordenada en una interfaz simple de entender para el usuario.

4.16 Actividad 15 – Pruebas de la Aplicación

Una vez finalizada la aplicación, esta se distribuyó a un grupo limitado de personas de diferentes ocupaciones para garantizar el correcto funcionamiento de la aplicación. También se buscó probar que los diferentes usuarios no tengan inconvenientes con la interfaz ya que en dispositivos móviles se busca que la interfaz sea intuitiva y fácil de usar. Se tuvo retroalimentación positiva, acompañada de recomendaciones para una futura versión de la aplicación. Las pruebas fueron satisfactorias y la aplicación se consideró completa.

Los beta testers y cualquier futuro usuario pueden enviar feedback y reportar bugs utilizando Github. El repositorio de la aplicación se encuentra en https://github.com/Dres90/TourngenAPP.

4.17 Actividad 16 – Distribuir la Aplicación

Completada la etapa de pruebas, la aplicación en este momento se encuentra disponible para cualquier usuario de Android 4.1 o mayor. Aunque no se encuentra todavía en la tienda de Google Play, cualquier usuario puede descargarla si tiene

habilitada la opción de descargar de otras fuentes. El instalador pesa menos de 500 Kb y se puede obtener navegando en el explorador del dispositivo a http://tourngen.com:8081/download. Se utilizó un diferente puerto para que el funcionamiento de la descarga no obstruya con el aplicativo web ni con los *Web Services* principales. Se levantó el servicio de descarga como un *Web Service* adicional pero sin usar SSL ya que Android presentaba algunos conflictos para la descarga de archivos sobre SSL con un certificado no reconocido. De todas maneras, el instalador está orientado a todo público por lo que no es necesario usar encriptación en esta descarga.

Capítulo 5

5.1 Conclusiones

- Se juegan y se generan torneos de futbol a diario alrededor del planeta. Estos se solían generar a mano y eran controlados por hojas de Excel o papel y lápiz debido al costo de las herramientas para ejercer esto. Tourngen Web ofreció una herramienta gratuita para hacer esto y ahora Tourngen Android permite esta funcionalidad en un dispositivo móvil sin necesidad de acceso permanente a internet.
- Se mantienen los privilegios establecidos por Tourngen Web, aumentando el control que tienen los usuarios sobre sus torneos y permitiendo delegar tareas a diferentes usuarios que pueden colaborar en la administración de un torneo. Ahora los digitadores pueden llevar cualquier dispositivo Android a la cancha y aportar con los resultados de los partidos y sincronizar cuando tengan acceso a internet.
- La simplicidad de la interfaz y de la generación automática de partidos permite a grupos de amigos jugar y administrar pequeños torneos de videojuegos sin necesidad de un computador ni internet.
- Los *Web Services* ya levantados permiten la creación de diferentes clientes, lo que abre la opción para que desarrolladores de diferentes dispositivos implementen sus propias aplicaciones para diferentes dispositivos.
- Estos *Web Services* están desarrollados de forma modular, por lo que más funcionalidad puede ser añadida al sistema sin mayores complicaciones.

5.2 Recomendaciones

- Para definir el alcance de un proyecto de desarrollo es muy útil la elaboración de casos de uso detallados. La ausencia de estos puede causar que el alcance cada vez crezca y que el proyecto tenga dificultades para ser completado. Los casos de uso probaron ser muy útiles para la planificación de este proyecto y ayudaron a plantear las actividades necesarias para definir el cumplimiento del mismo.
- Los sistemas operativos para dispositivos móviles reciben actualizaciones constantes y cada vez obtienen mejoras. Es importante proveer soporte constante a estas aplicaciones para asegurarse que siempre se mantengan compatibles con las versiones más actualizadas de los dispositivos.
- La estructuración y planificación previa de un proyecto de desarrollo es de gran importancia antes de empezar a escribir código. En este caso se detalló el desarrolló por módulos ordenados lo que permitió que el proyecto se complete teniendo muy pocas modificaciones que hacer a los módulos ya completados. Un cronograma con horas de desarrollo para cada tarea puede ayudar a completar los tiempos planificados para un proyecto.
- Una investigación profunda de los estándares a ser usados en el proyecto es necesaria para garantizar el éxito durante el desarrollo. Un inconveniente por no adherirse al estándar HTTP causó retraso en el desarrollo de algunos módulos y requirió modificaciones a los *Web Services* ya implementados. En proyectos de interoperabilidad entre diferentes tecnologías es importante conocer y respetar los estándares involucrados.

5.3 Futuros Desarrollos

- Aumentar la funcionalidad de los *Web Services* existentes. El desarrollo modular permite agregar fácilmente funcionalidad a la existente. En futuras versiones los *Web Services* podrían permitir la modificación de cada una de las entidades actuales como los equipos y las jornadas.
- Aprovechar la funcionalidad de privilegios en Tourngen Web. Usando las modificaciones a la base de datos de este proyecto se puede adaptar Tourngen Web para que permita al mismo usuario tener diferentes privilegios sobre diferentes torneos.
- Existen proyectos para el crecimiento del alcance de Tourngen Web. Entre estos está la posibilidad de administrar torneos de eliminación directa o el manejo de jugadores dentro de los equipos para obtener tablas de goleadores y permitir el control de expulsiones, lesiones o suspensiones de jugadores. Estas nuevas funcionalidades podrían ser desarrolladas en conjunto a nuevos *Web Services* para poder ofrecer la funcionalidad en los diferentes clientes móviles.
- Un cliente nativo de dispositivos iOS aumentaría la potencial cantidad de usuarios de la aplicación. Los mismos *Web Services* pueden ser reutilizados lo que disminuiría la complejidad del desarrollo de esta nueva aplicación. El desarrollo de esta alternativa es conveniente cuando la aplicación Android y el sistema web hayan adquirido una cantidad considerable de usuarios para agilizar el proceso de admisión a la tienda de aplicaciones de Apple.
- Una interfaz gráfica exclusivamente orientada a tablets puede hacer mejor uso del espacio de pantalla de estos dispositivos. El código puede ser reusado ya que el sistema operativo y la funcionalidad es en esencia el mismo, pero la interfaz del usuario podría ser rediseñada para proveer una mejor experiencia al usuario de tablets.

- Los usuarios que siguen un torneo se verían beneficiados de la implementación de un sistema de alertas directas al teléfono cuando alguno de los partidos del torneo sea modificado. Es importante que sea una opción personalizable por el usuario ya que las notificaciones indeseadas pueden causar la salida del usuario. Sin embargo, existen muchos usuarios que apreciarían recibir alertas cuando sus equipos anoten goles y que sean notificados cuando existan datos nuevos relevantes en el torneo.

Referencias

Coburn, L. (2011). *Developing on Mobile vs Desktop.* DoubleDutch Inc. Retrieved from http://www.slideshare.net/DoubleDutch_Inc/developing-on-mobile-vs-desktop-dreamforce-2011-lawrence-coburn-doubledutch

Crockford, D. (2008). Good Parts In *JavaScript: The Good Parts* (pp. 2-3) United States: O'Reilly Media, Inc. Retrieved from http://books.google.com.ec/books?id=PXa2bby0oQ0C&printsec=frontcover#v=onepage&q&f=false

Dhingra, S. (2013). *REST vs. SOAP: How to choose the best Web Services.* SearchSOA. Retrieved from http://searchsoa.techtarget.com/tip/REST-vs-SOAP-How-to-choose-the-best-Web-service

Fingas, J. (2014). *Android climbed to 79 percent of smartphone market share in 2013, but its growth has slowed.* Engadget. http://www.engadget.com/2014/01/29/strategy-analytics-2013-smartphone-share/

Flanders, J. (2009). *More on REST.* MSDN Magazine. Retrieved from http://msdn.microsoft.com/en-us/magazine/dd942839.aspx

Gosling, J., Joy, B., Steele, G., & Bracha, G. (2000). Introduction In *The java Language Specification* (2nd ed, pp 1-2). California: Sun Microsystems, Inc. Retrieved from http://books.google.com.ec/books?id=Ww1B9O_yVGsC&printsec=frontcover#v=onepage&q&f=false

Halfpap, B. (November, 2013). *Choosing Your First Programming Language.* CTOvision.com. Retrieved from https://ctovision.com/2013/11/choosing-first-programming-language/

Heggestuen, J. (2013). *One In Every 5 People In The World Own A Smartphone, One In Every 17 Own A Tablet.* Business Insider. Retrieved from http://www.businessinsider.com/smartphone-and-tablet-penetration-2013-10

Kiessling, M. (2014) *Server-side Javascript*. The Node Beginner Book. Retrieved from http://www.nodebeginner.org/#server-side-javascript

Microsoft. (2012) *Motor de Base de Datos de SQL Server*. Retrieved from http://msdn.microsoft.com/es-es/library/ms187875.aspx

Mitchell, B. (n.d.). *HTTP*. About.com Retrieved from http://compnetworking.about.com/od/networkprotocols/g/bldef_http.htm

Pérez, D. (October, 2007) *¿Qué son las Bases de Datos?*. Retrieved from http://www.maestrosdelweb.com/principiantes/¿que-son-las-bases-de-datos/

Rescorla, E. (May, 2000). *HTTP Over TLS*. Internet Engineering Task Force. Retrieved from https://tools.ietf.org/html/rfc2818

Suehring, S. (2002). Getting Started In *MySQL Bible* (pp. 7-8) New York: Wiley Publishing, Inc. Retrieved from http://www.chettinadtech.ac.in/g_article/Textbook2%20%20MySQL%20Bible.pdf

Anexo 1: Manual de usuario de la aplicación Android

Instalación

Requisitos previos:
Hasta el momento de la redacción de este manual, la aplicación aún no ha sido subida al Play Store de Google, por lo que la instalación requiere que el dispositivo Android permita instalar de fuentes externas al Play Store. Esta configuración suele estar bajo la categoría de seguridad en los ajustes de cada dispositivo. Además, se debe asegurar que la versión de Android en la que se vaya a instalar el aplicativo sea mínimo Android 4.1.

Paso 1:
Ingresar al explorador web del dispositivo y visitar http://tourngen.com:8081/Download. El instalador se descargará en el dispositivo.

Paso 2:
Una vez completada la descarga, abrir el instalador llamado Tourngen.apk.

Paso 3:
Al finalizar la instalación, buscar entre las aplicaciones instaladas el siguiente ícono:

Figura 6.1: Ícono de la aplicación de Android

Tocar el ícono presentado en la figura 6.1 para iniciar la aplicación

Iniciar Sesión

Requisitos previos:
Tener una cuenta de Tourngen creada en Tourngen Web o usando los pasos señalados en la sección Crear Cuenta Nueva de este manual.

Paso 1:
En la pantalla inicial de la aplicación (figura 6.2) ingresar los datos de usuario y contraseña en los campos correspondientes. Si esta no es la primera pantalla que ve al iniciar la aplicación, por favor seguir los pasos en la sección Cerrar Sesión y volver a intentar.

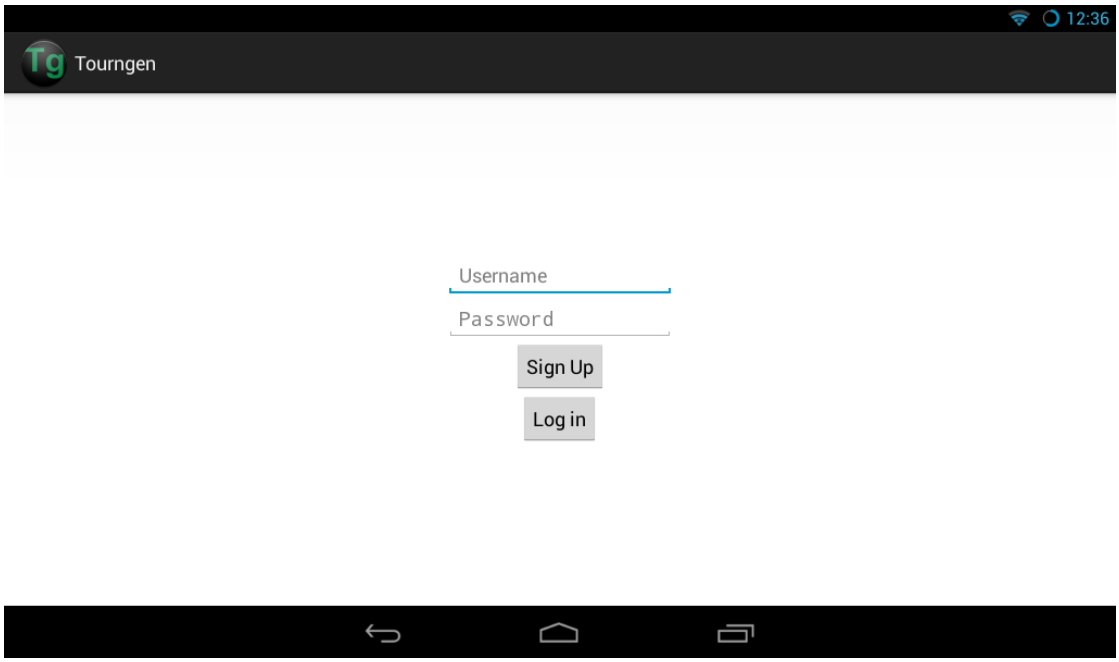

Figura 6.2: Pantalla de Login

Paso 2:
Tocar el botón de *Log In*. Si existe algún error por favor verificar los datos ingresados o la conexión a internet del dispositivo y volver a intentar.

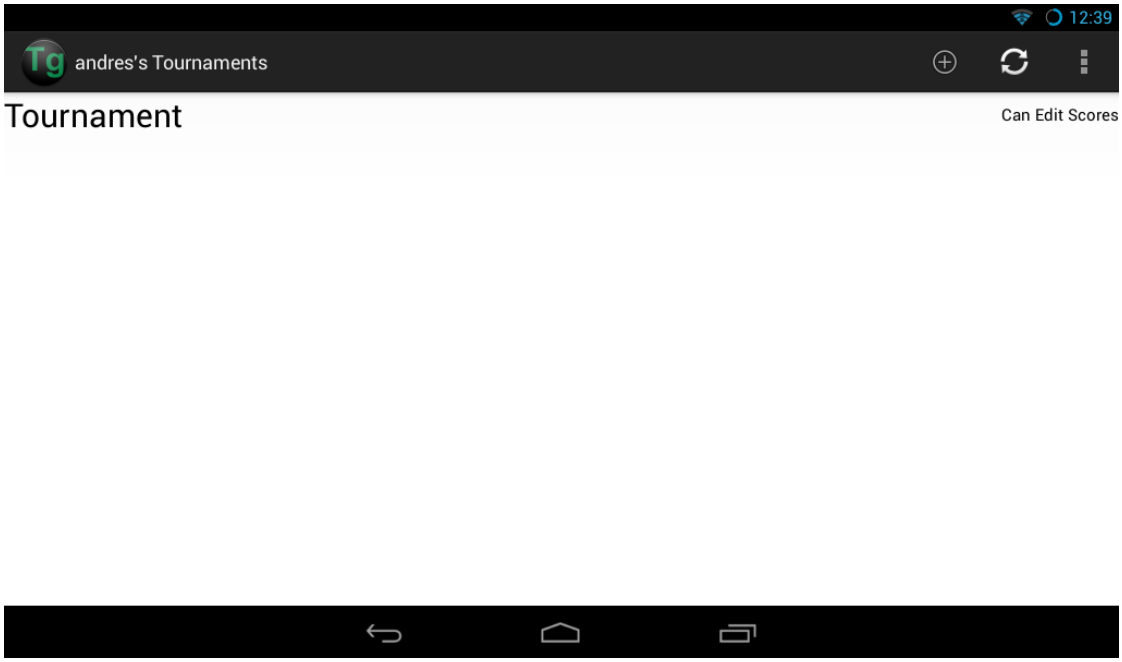

Figura 6.3: Pantalla de Torneos vacía

Paso 3:

Al iniciar sesión correctamente se mostrará la pantalla de torneos presentada en la figura 6.3. Para descargar todos los torneos disponibles para el usuario, presionar el ícono de sincronización mostrado en la figura 6.4 que se encuentra en la esquina superior derecha. La pantalla con todos los torneos disponibles descargados se pueden ver en la figura 6.5. La columna *Can Edit Scores* informa al usuario si tiene los privilegios necesarios para editar los resultados de ese torneo.

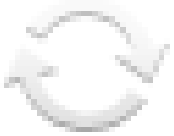

Figura 6.4: Ícono de sincronización

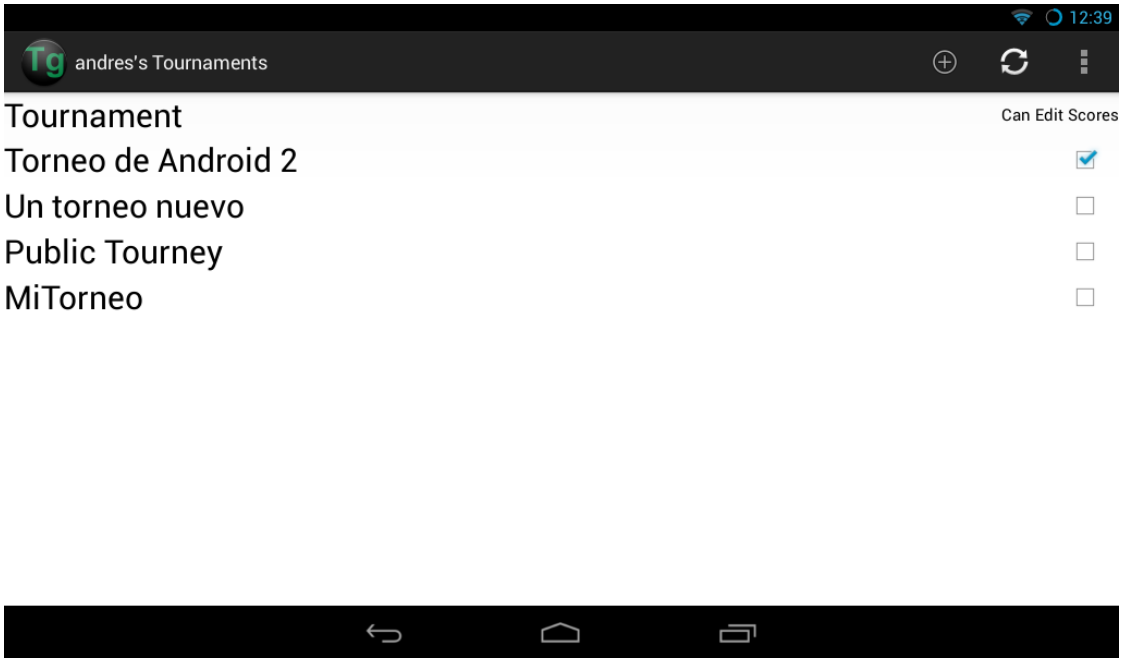

Figura 6.5: Pantalla de Torneos llena

Crear Cuenta Nueva

Paso 1:
En la pantalla inicial de la aplicación (figura 6.2) tocar el botón de *Sign Up*. Si esta no es la primera pantalla que ve al iniciar la aplicación, por favor seguir los pasos en la sección Cerrar Sesión y volver a intentar.

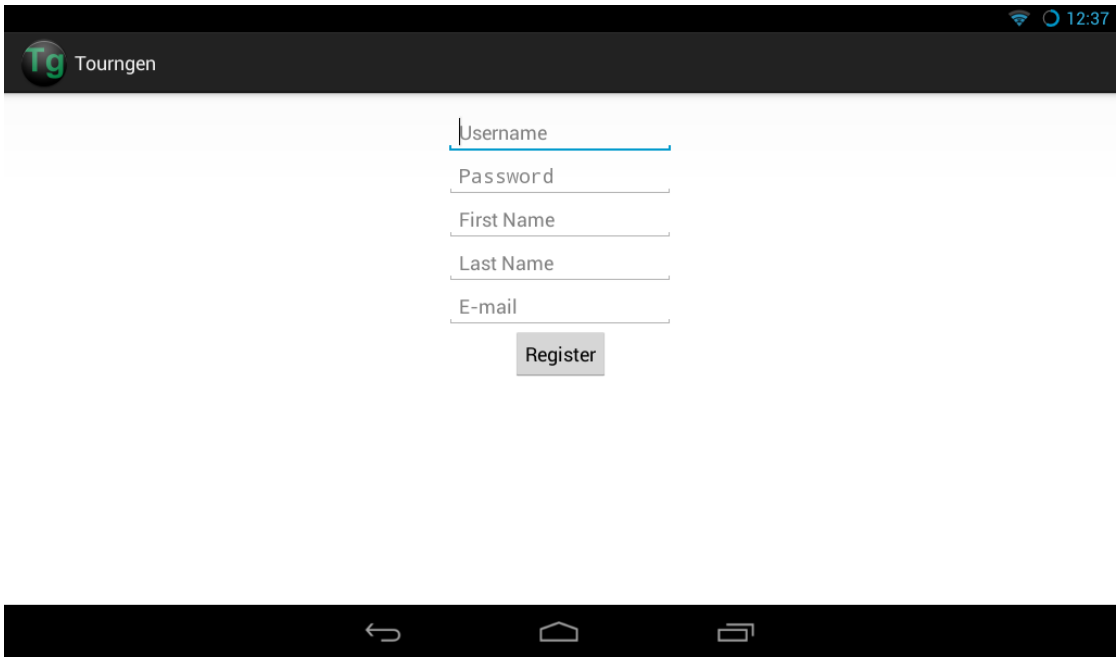

Figura 6.6: Pantalla de Nuevo Usuario

Paso 2:
Llenar todos los campos presentados en la pantalla de Nuevo Usuario (figura 6.6) y tocar el botón *Register*. Si el proceso fue exitoso se notificará con un mensaje.

Paso 3:
Después del mensaje de éxito, será regresado a la pantalla de Log In (figura 6.2). En esta ventana seguir los pasos en la sección Iniciar Sesión usando los datos del usuario nuevo.

Cerrar Sesión

Paso 1:
En la pantalla de torneos (figura 6.5) tocar el botón de menú. En dispositivos que no tengan este botón físico, automáticamente aparece un botón a la esquina superior derecha de la aplicación como se ve en la figura 6.7

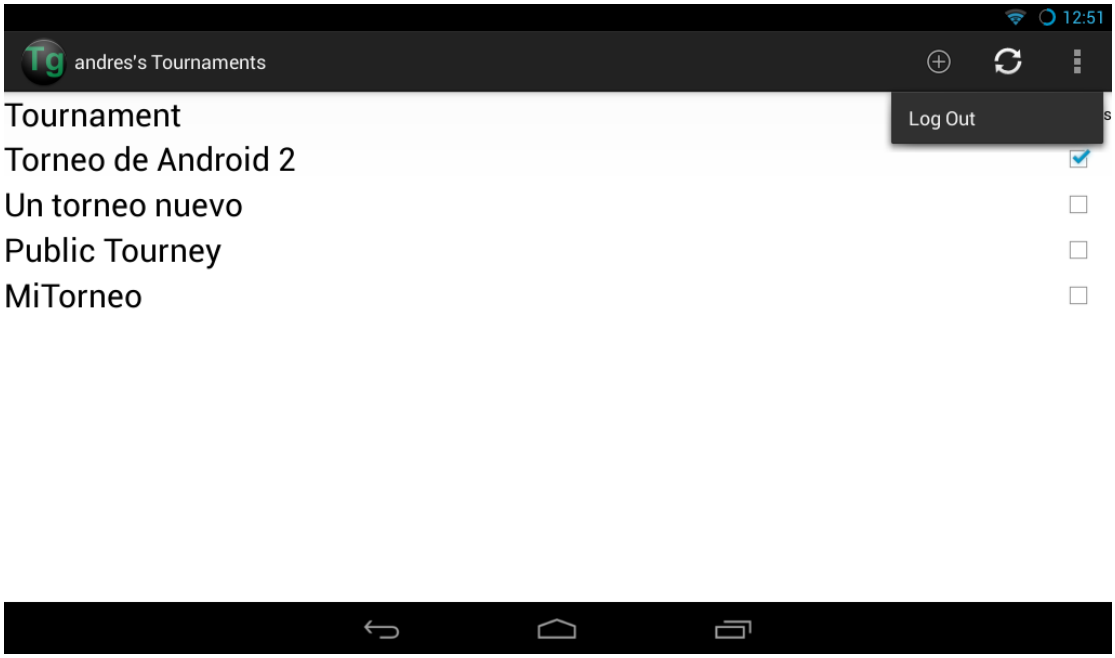

Figura 6.7: Cerrar sesión

Paso 2:
Tocar la opción que aparece llamada *Log Out*. Esto eliminará todos los datos en el dispositivo que no hayan sido sincronizados con el servidor central. Después de que la sesión se cierre correctamente, la aplicación regresa a la pantalla de *Log In.*

Actualizar todos los Torneos

Paso 1:
En la aplicación abrir la pantalla de torneos como se muestra en la figura 6.5

Paso 2:
Tocar el botón de sincronización en la esquina superior derecha. El ícono de sincronización se puede ver en la figura 6.4. Este proceso descargará todos los torneos que tenga disponible el servidor para este usuario y sobrescribirá todos los cambios que no se hayan subido. Para subir los cambios de un torneo, seguir los pasos en la sección Sincronizar un Torneo.

Crear Torneo Nuevo

Paso 1:
En la aplicación abrir la pantalla de torneos como se muestra en la figura 6.5

Paso 2:

Figura 6.8: Ícono para agregar

Tocar el botón para añadir torneos en la esquina superior derecha. El ícono de este botón se puede ver en la figura 6.8. Esto abrirá la pantalla que se muestra en la figura 6.9

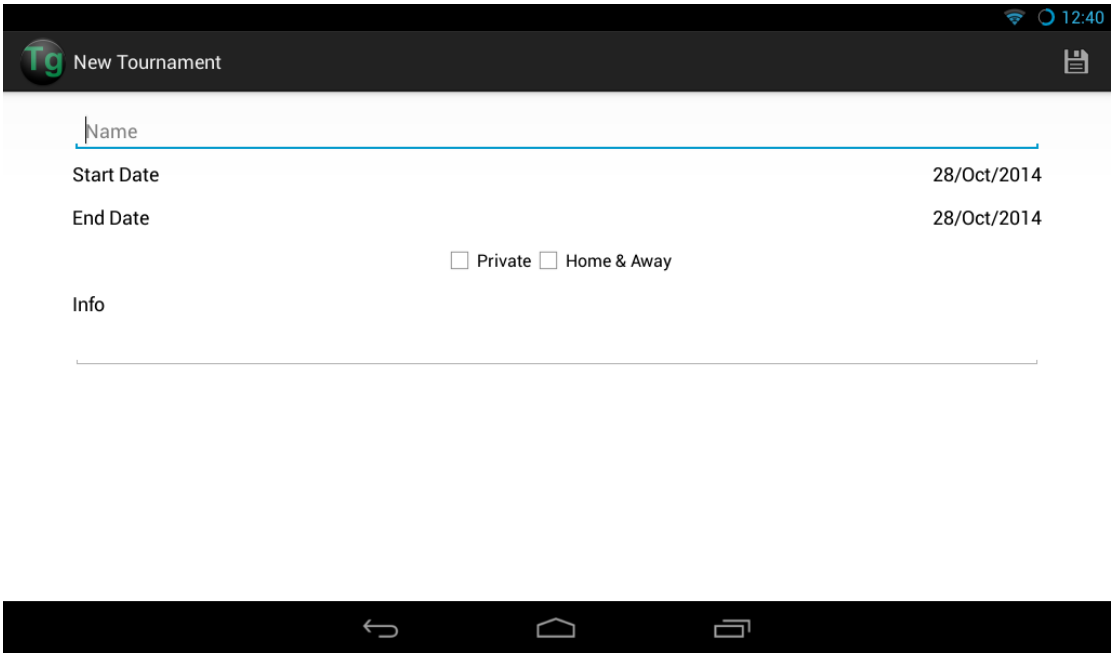

Figura 6.9: Nuevo Torneo

Paso 3:
Ingresar la información correspondiente al torneo nuevo. Para cambiar la fecha se toca el campo de fecha y se abre el editor de fechas que se ve en la figura 6.10. Una vez completas las ediciones a la información del torneo, tocar el botón de guardar en la esquina superior derecha. El ícono de este botón se puede apreciar en la figura 6.11.

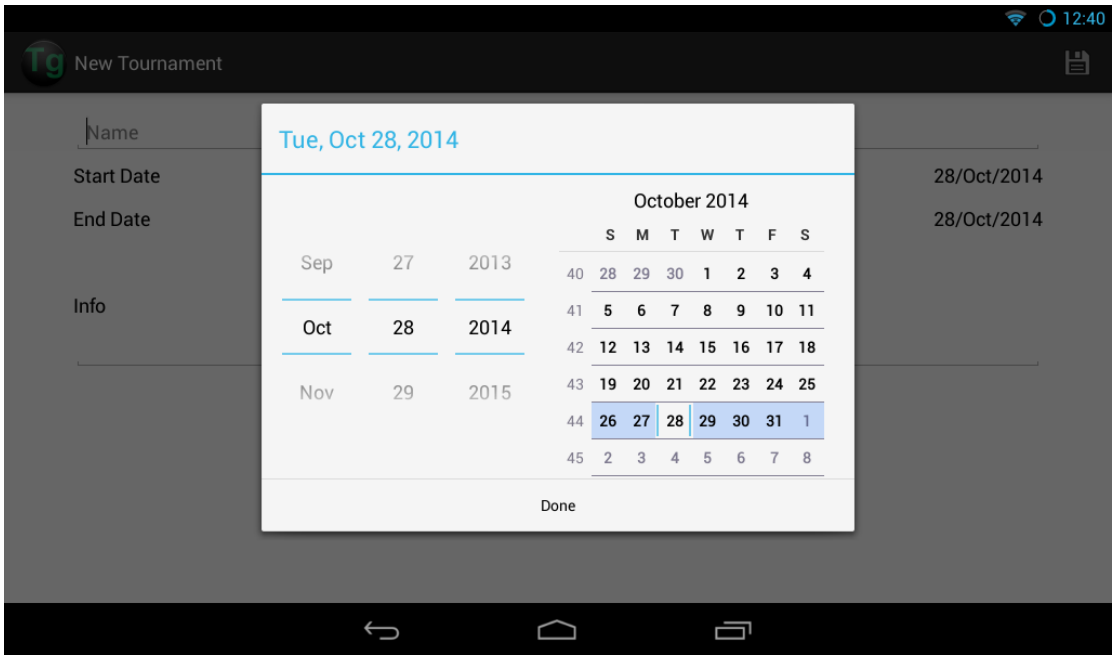

Figura 6.10: Editor de Fechas

Figura 6.11: Ícono para guardar

Paso 4:

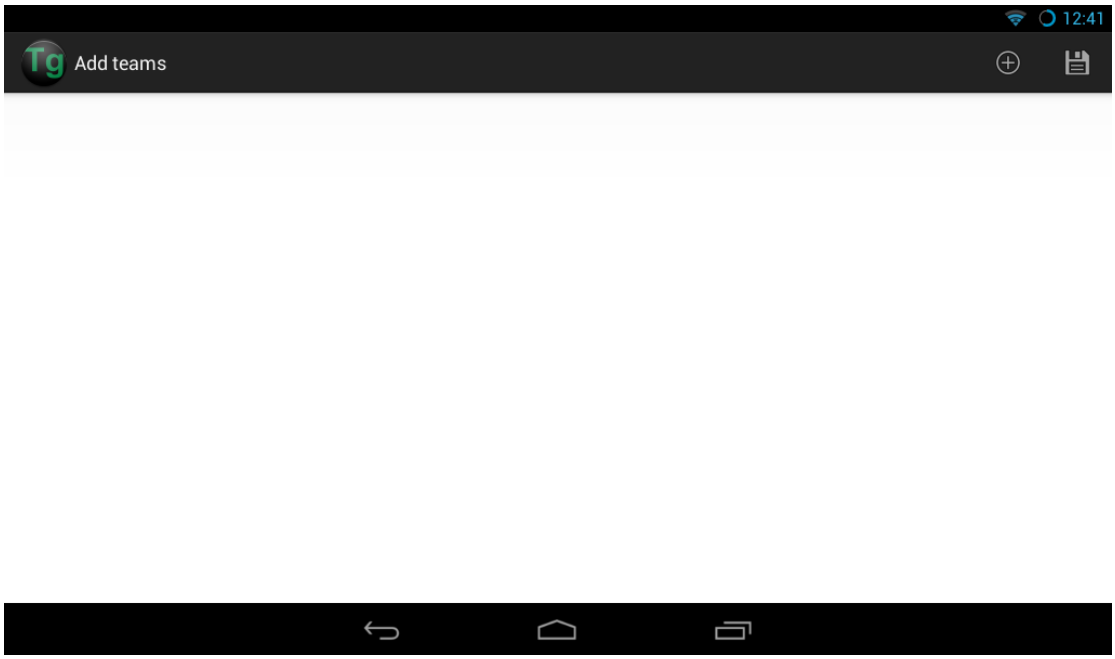

Figura 6.12: Listado de Equipos Nuevos Vacío

En la figura 6.12 se presenta la siguiente pantalla, donde se listan los equipos correspondientes al nuevo torneo. Inicialmente el torneo no tiene equipos. Para agregar equipos, tocar el botón con el ícono de agregar (figura 6.8). Esto abre el editor de equipos que se presenta en la figura 6.13

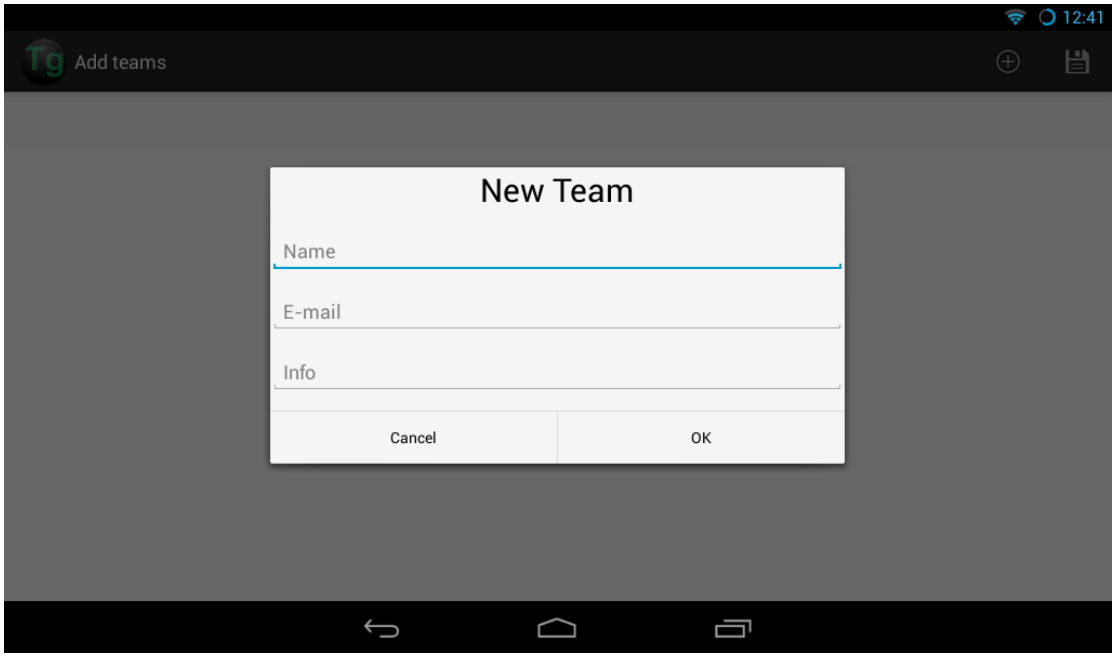

Figura 6.13: Editor de Equipos

Paso 5:
En el editor de torneos ingresar la información del equipo nuevo. Solo es obligatorio ingresar el nombre del equipo, el resto de campos es opcional. Al precionar el botón de OK se almacenará el equipo. Repetir el paso 4 y el paso 5 hasta tener todos los equipos participantes en el torneo.

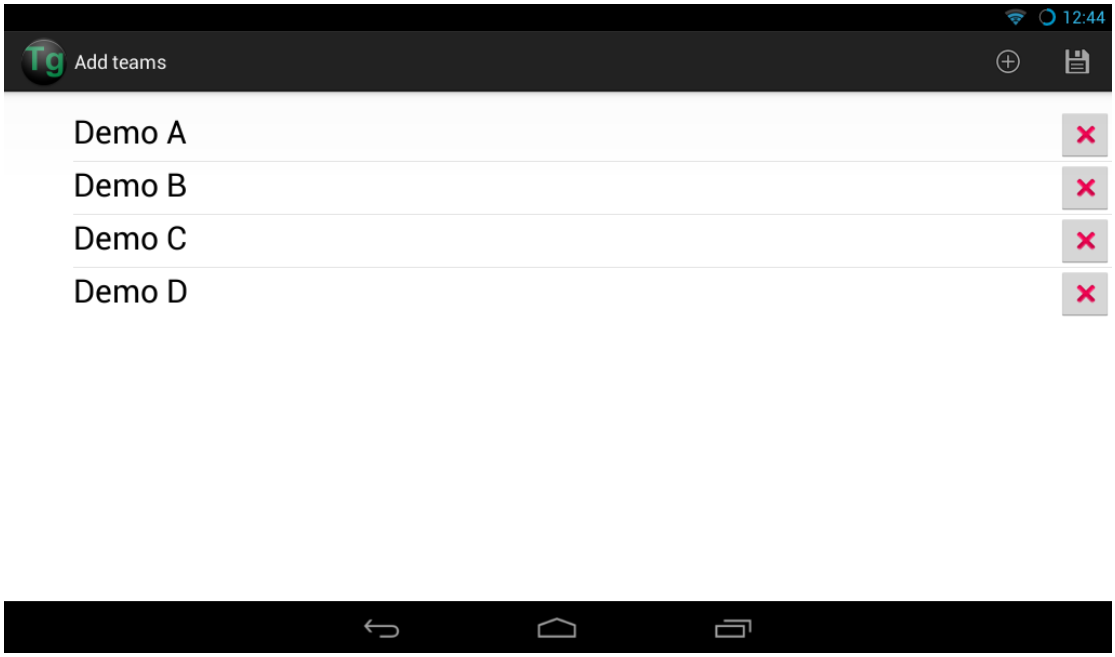

Figura 6.14: Listado de Equipos Nuevos Lleno

Paso 6:
En la figura 6.14 se muestra el listado de equipos nuevos lleno. Los equipos pueden ser borrados usando la X roja del lado derecho de cada equipo. Una vez confirmados los equipos necesarios tocar el botón de guardar (Figura 6.11) en la esquina superior derecha.

Paso 7:

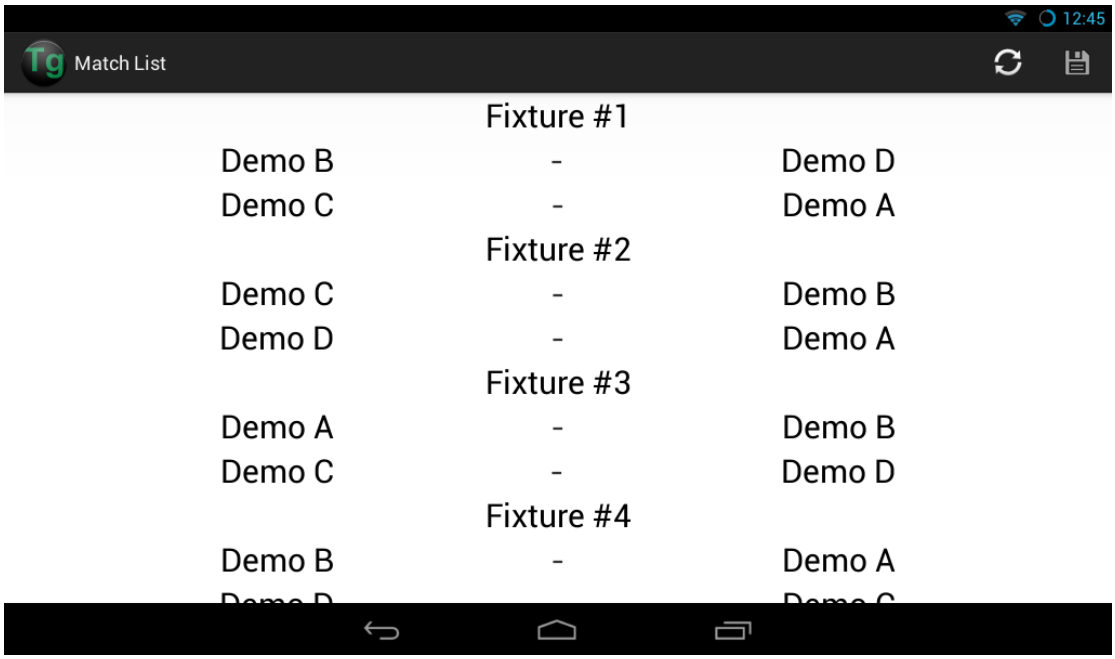

Figura 6.15: Listado de Jornadas Generadas 1

La siguiente pantalla (figura 6.15) presenta el listado de los partidos generados automáticamente por la aplicación. Si este calendario no es del agrado del usuario puede tocar el botón de sincronizar (figura 6.4) para volver a generar un listado nuevo como se ve en la figura 6.16. Cuando se consigue un listado de partidos del agrado del usuario tocar el botón de guardar (figura 6.11) en la esquina superior derecha. Esto guarda el torneo en el dispositivo y lo agrega al listado de partidos como se ve en la figura 6.17.

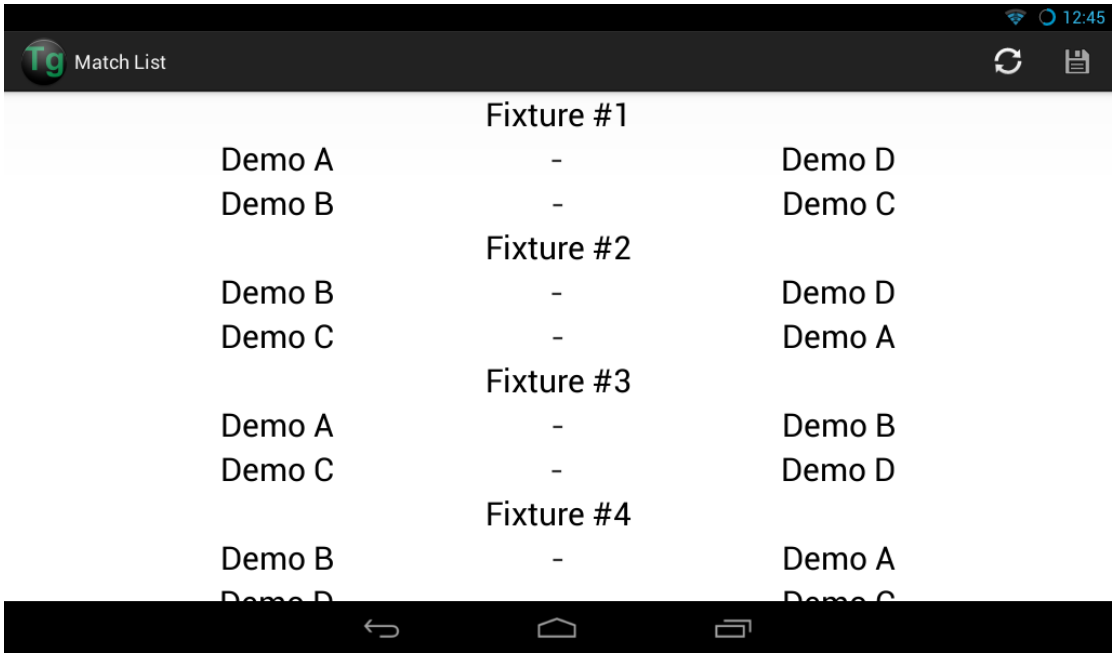

Figura 6.16: Listado de Jornadas Generadas 2

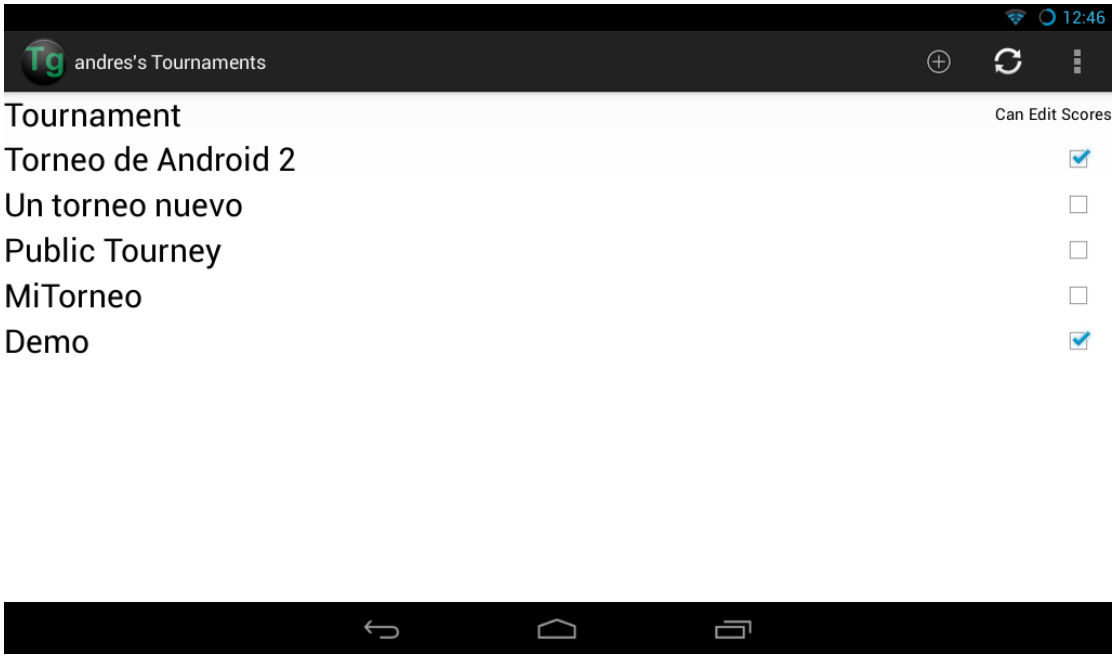

Figura 6.17: Torneo Nuevo Agregado al Listado

Ver Información de un torneo y tabla de posiciones

Paso 1:

En la pantalla de torneos (figura 6.17) tocar el torneo deseado. Esto lleva al usuario a la pantalla de torneo ilustrada en la figura 6.18. Esta pantalla muestra la información correspondiente al torneo y presenta la tabla de posiciones correspondiente al torneo.

Figura 6.18: Pantalla de Torneo

Sincronizar un Torneo

Paso 1:
En la pantalla de torneo (figura 6.18) tocar el botón de sincronizar (figura 6.4) en la esquina superior derecha. La primera vez que se hace esto sube el torneo al servidor y las siguientes veces sincroniza la información del torneo entre el dispositivo y el servidor.

Ver detalle de equipos de un torneo

Paso 1:
En la pantalla de torneo (figura 6.18) tocar la opción que se encuentra en la parte superior central que lee Teams. Esto lleva al usuario a un listado de equipos como se muestra en la figura 6.19.

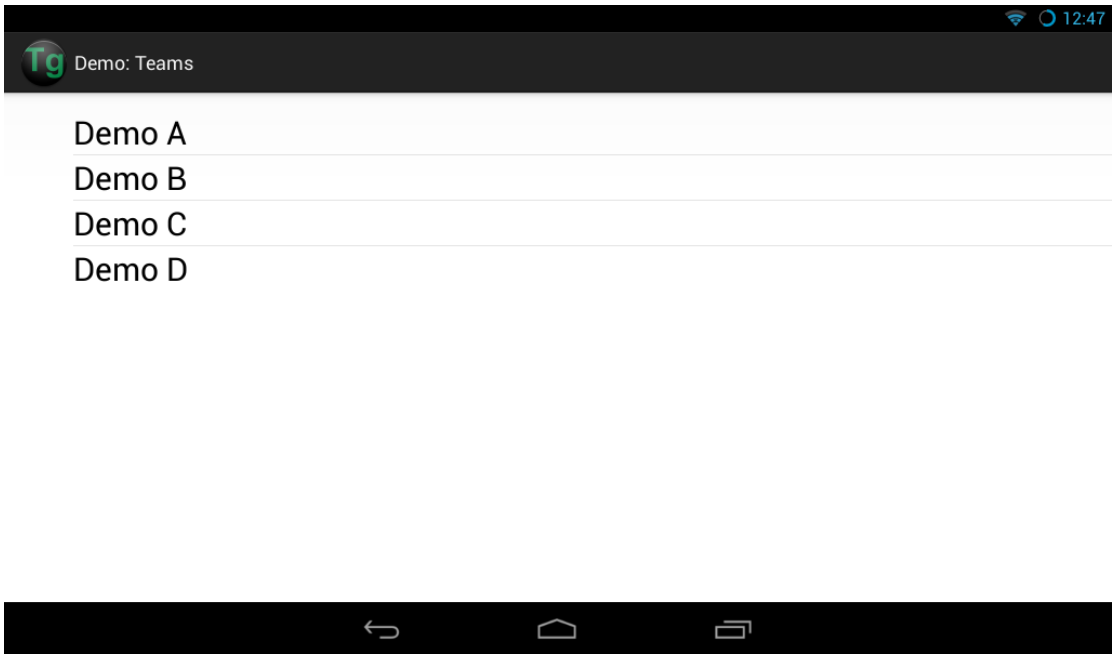

Figura 6.19: Listado de Equipos

Paso 2:

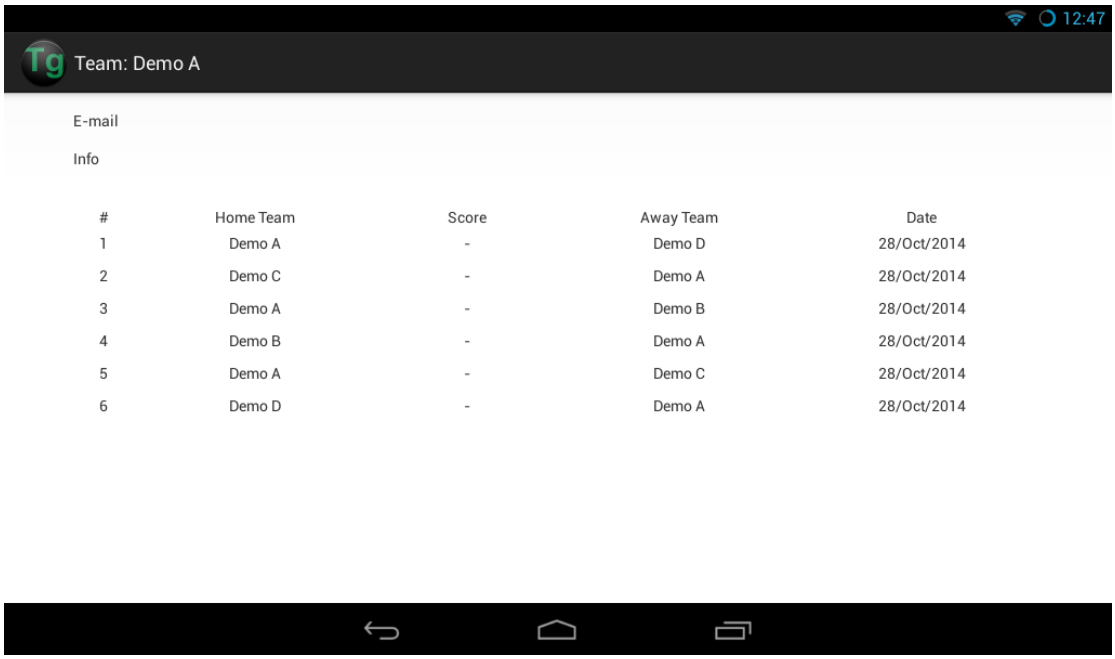

Figura 6.20: Detalle de Equipo

Tocar el equipo del que se desea ver más detalle. Esto lleva a la pantalla de equipo como se muestra en la figura 6.20. Acá se muestra la información del equipo (Nombre, e-mail e información) y el listado de los partidos del equipo en el torneo.

Ver partidos de un torneo por jornadas

Paso 1:

En la pantalla de torneos (figura 6.18) tocar la opción que se encuentra en la parte superior central que lee Fixtures. Esto lleva al usuario a un listado de partidos filtrados por jornadas como se muestra en la figura 6.21.

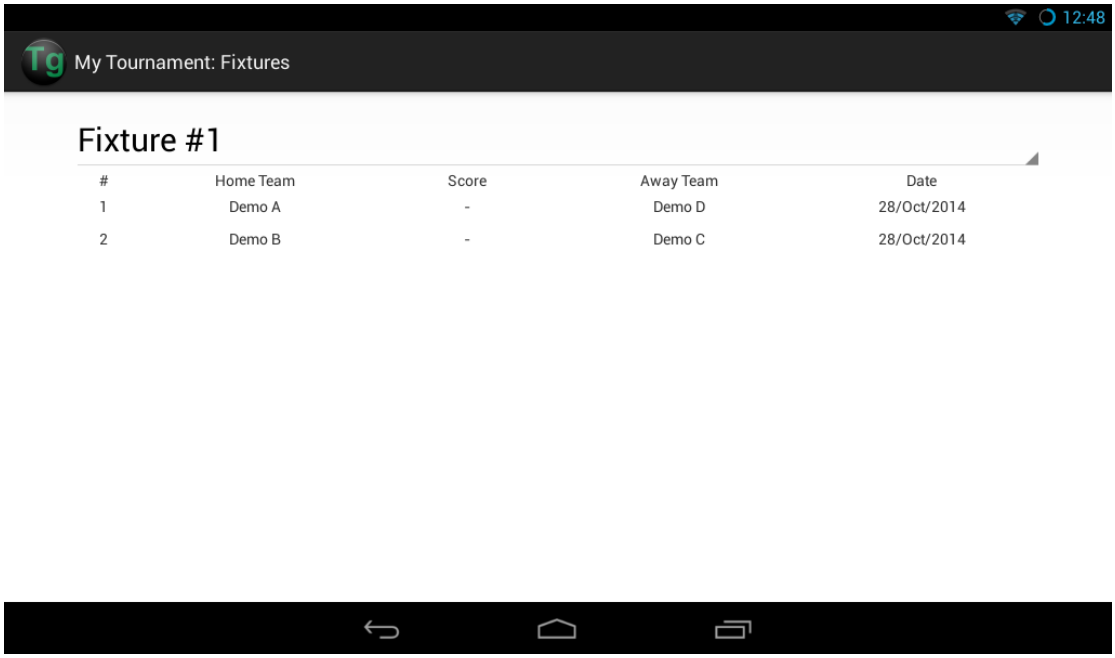

Figura 6.21: Partidos por Jornadas

Paso 2:

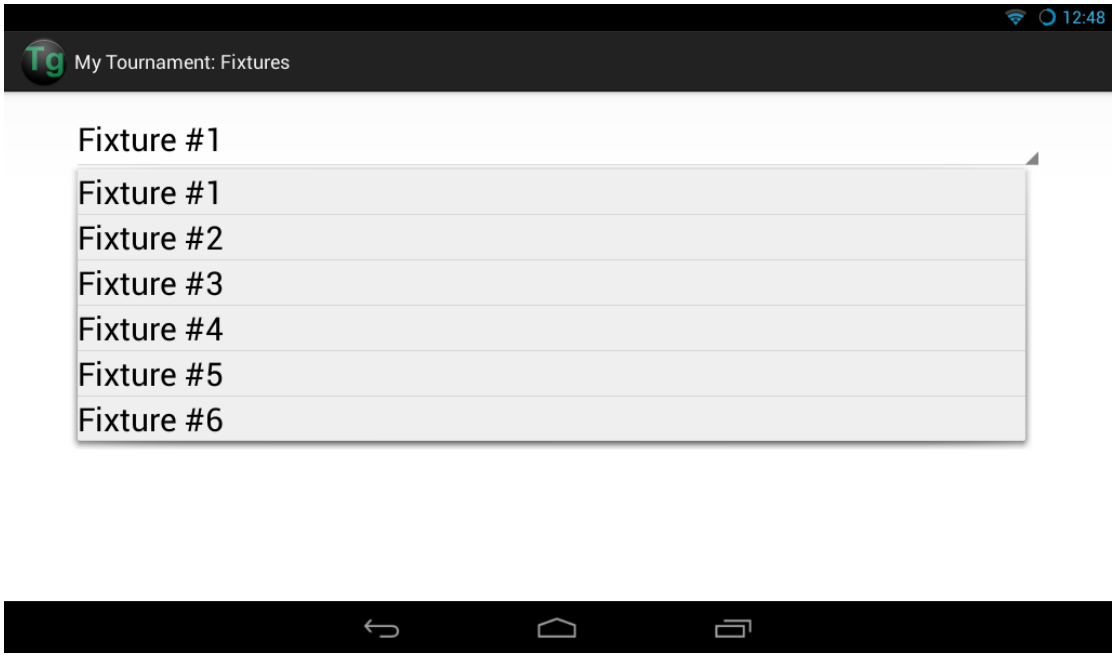

Figura 6.22: Selector de Jornada

Al tocar el número de la jornada aparece un selector de todas las jornadas disponibles como se muestra en la figura 6.22. Al tocar una jornada diferente se muestran solo los partidos correspondientes a esa jornada.

Ver todos los partidos de un torneo

Paso 1:
En la pantalla de torneos (figura 6.18) tocar la opción que se encuentra en la parte superior central que lee Matches. Esto lleva al usuario a un listado de todos los partidos del torneo como se muestra en la figura 6.23.

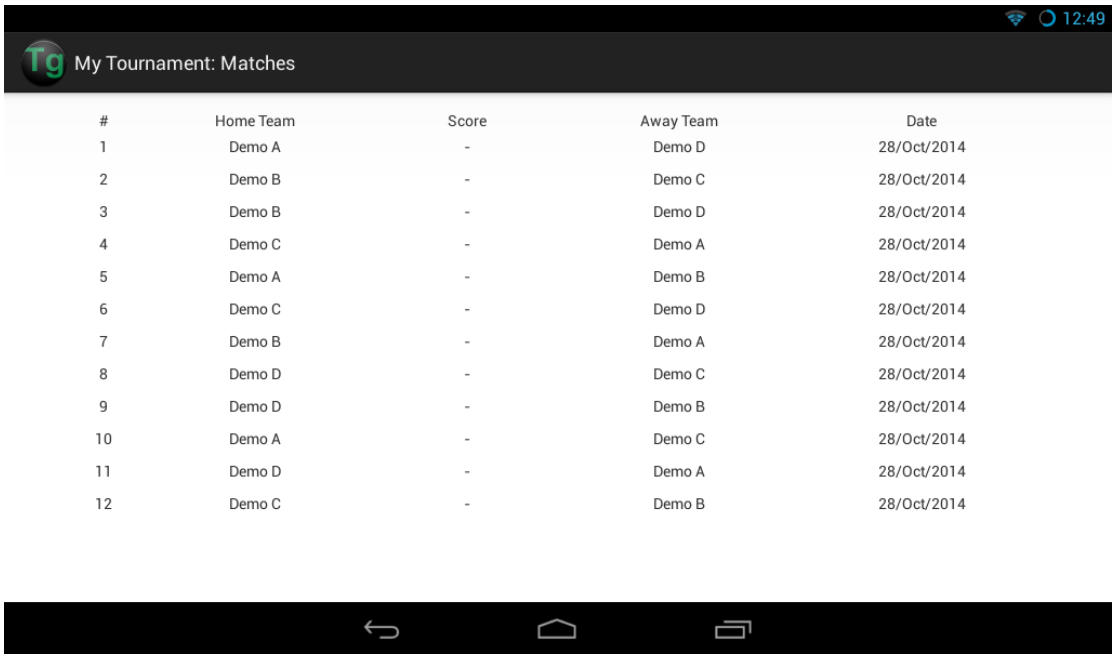

Figura 6.23: Listado de Partidos

Ver o editar detalles de un partido

Paso 1:
Los detalles de un partido pueden ser vistos al tocar un partido en el listado de partidos de un equipo (figura 6.20), listado de partidos por jornada (figura 6.21) o listado de partidos de un torneo (figura 6.21). Todos estos llevan a la pantalla de detalles de partido.

Paso 2:

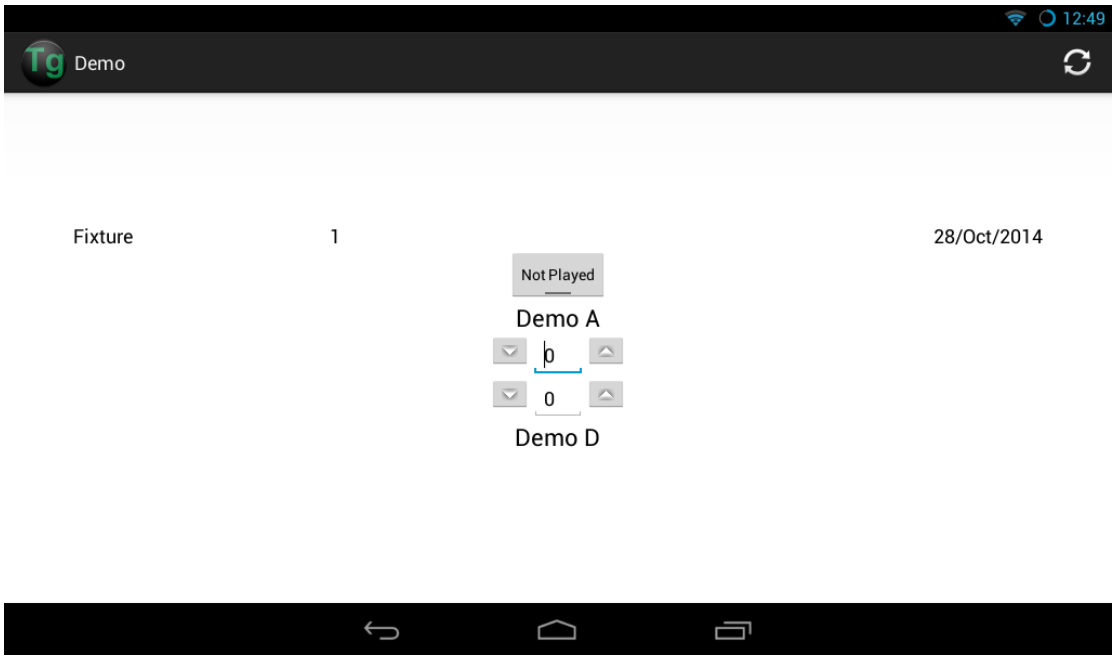

Figura 6.24: Detalle de Partido Editable

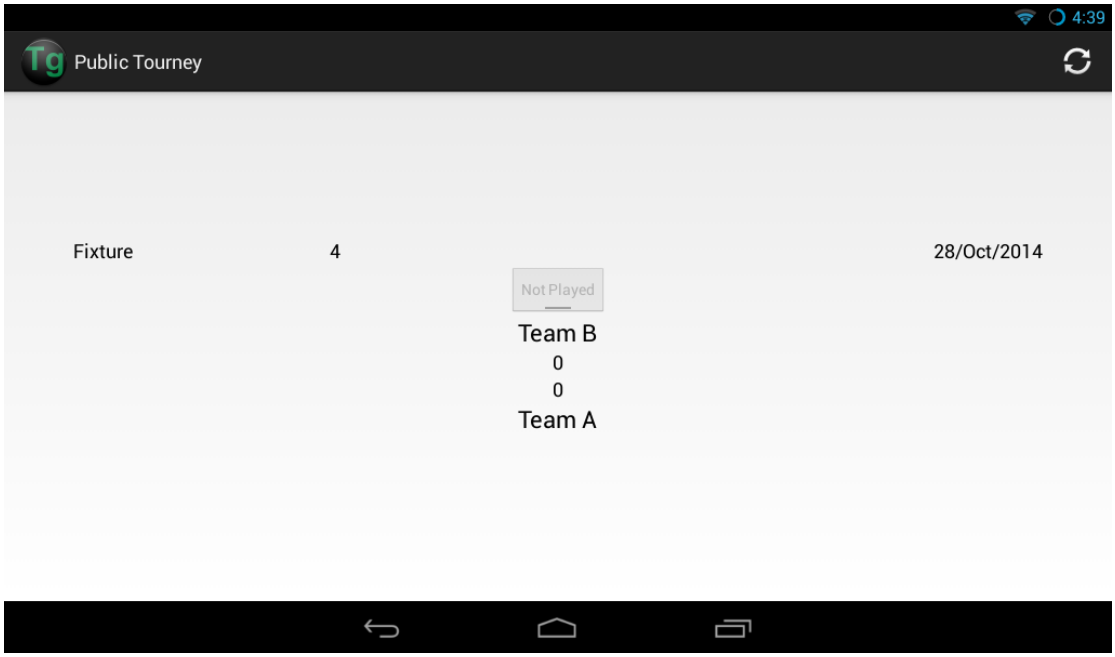

Figura 6.25: Detalle de Partido No Editable

Si el usuario cuenta con privilegios de edición de resultados (Administrador de torneo o digitador) se presenta el detalle de partido editable como se ve en la figura 6.24. En esta pantalla el usuario puede cambiar el resultado y el estado de un partido. Si no cuenta con estos privilegios, se presenta la pantalla de detalle de la figura 6.25 en la cual el usuario solo puede ver los detalles del partido sin poder hacer ediciones. Para sincronizar el

partido con el servidor tocar el botón de sincronización (figura 6.4) que se muestra en la esquina superior derecha.

Borrar un Torneo

Paso 1:
En la pantalla de torneo (figura 6.18) tocar el botón de menú. En dispositivos que no tengan este botón físico, automáticamente aparece un botón en la esquina superior derecha de la aplicación.

Paso 2:

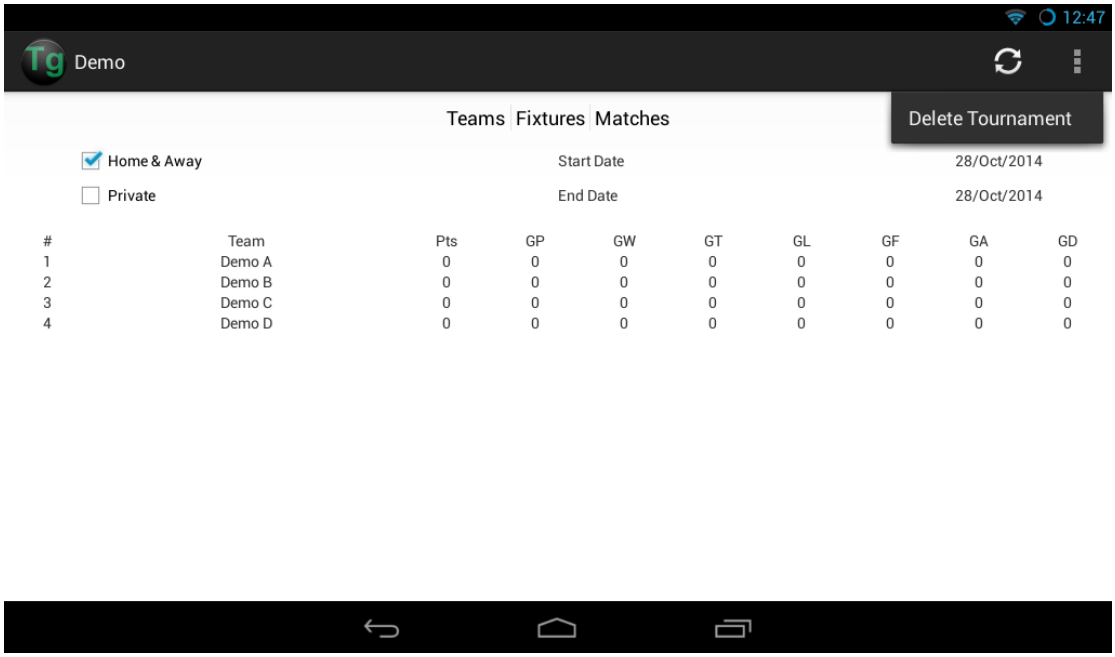

Figura 6.26: Borrar Torneo

Tocar la opción *Delete Tournament* que aparece como se ve en la figura 6.26. Esto borra el torneo de la memoria local y si el torneo está sincronizado con el servidor y se cuenta con conexión al internet, el torneo será borrado.