ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN SISTEMA PROTOTIPO WEB EN ANDROID PARA GESTIONAR LOS PARTIDOS DE UNA LIGA BARRIAL DE FÚTBOL DE QUITO

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

FERNANDO JOSÉ HERRERA ORDÓÑEZ

DIRECTOR: M.Sc. XAVIER ALEXANDER CALDERÓN HINOJOSA

Quito, marzo 2020

AVAL

Certifico que el presente trabajo fue desarrollado por Fernando José Herrera Ordóñez, bajo mi supervisión.

M.Sc. XAVIER ALEXANDER CALDERÓN HINOJOSA
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Fernando José Herrera Ordóñez, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

_____

FERNANDO JOSÉ HERRERA ORDÓÑEZ

DEDICATORIA

A mis padres Aníbal Herrera y Myriam Ordóñez por enseñarme el valor de la perseverancia y la infinita entrega de amor hacia mí y mis hermanas. ¡Los amo!

A mis queridas hermanas Fernanda y Gabriela Herrera por formar parte fundamental en mi vida y por haberme brindado un apoyo incondicional para que yo consiga lograr este objetivo.

A mi querida novia Carolina Unapanta por brindarme todo su apoyo a lo largo de toda mi carrera y haber llenado de alegría y amor todos mis días.

A mis amigos que siempre estuvieron apoyándome desde los primeros semestres hasta el último momento y por todos los momentos compartidos.

A mi Director de Trabajo de Titulación por haber tenido la paciencia y conocimiento suficiente para guiarme a lo largo de todo el trabajo.

Fernando Herrera

AGRADECIMIENTO

Agradezco a Dios por brindarme la salud y conocimiento suficiente para poder conseguir un objetivo muy importante en mi vida.

A mis padres Aníbal Herrera y Myriam Ordóñez por toda su paciencia y amor que me dieron fuerzas para continuar y no darme por vencido.

A mis hermanas Fernanda y Gabriela Herrera que siempre estuvieron presentes compartiendo momentos de felicidad y alegría.

A mi novia Carolina Unapanta por haber compartido conmigo momentos inolvidables llenos de amor y felicidad.

A mi director y profesor M.Sc. Xavier Calderón por todas sus enseñanzas compartidas a lo durante los últimos semestres y a lo largo del Trabajo de titulación.

Fernando Herrera

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

ÍNDICE DE ABREVIATURAS

AES	Advanced Encryption Standard
AWS	Amazon Web Services
CRUD	Create, Read, Update, Delete
DAO	Data Access Object
DDL	Data Definition Language
DML	Data Manipulation Language
HAL	Hardware Abstraction Layer
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IIS	Internet Information Services
JSON	JavaScript Object Notation
REST	Representational State Transfer
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSMS	SQL Server Management Studio
TCP	Transmission Control Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
WAS	Windows Process Activation Services
WCF	Windows Communication Service
XML	Extensible Markup Language

RESUMEN

El presente trabajo consiste en el desarrollo de un prototipo de sistema para gestionar los partidos de la liga deportiva barrial “La Armenia”. El prototipo se encuentra conformado principalmente por dos partes: un Cliente Android y un servicio WCF (Windows Communication Foundation) basado en REST (Representational State Transfer).

El presente documento se encuentra constituido por cuatro capítulos:

En el primer capítulo se presentan conceptos sobre el servicio WCF basada en REST, sistema operativo Android, bases de datos, la metodología ágil Kanban, el cifrado AES (Advanced Encription Standard) utilizado para el envío y recepción de credenciales de usuario. Finalmente se presenta las herramientas de desarrollo utilizadas.

El segundo capítulo presenta la metodología empleada para el desarrollo del presente trabajo, el cual consiste en dos partes: diseño e implementación. La parte de diseño presenta el establecimiento de requerimientos de usuario y diseño de los componentes del sistema. En la parte de implementación se presenta la codificación de la lógica de los componentes.

En el tercer capítulo se presentan los resultados de las pruebas realizadas a los componentes del prototipo. Además, se presentan los resultados de encuestas de validación realizadas por el personal administrativo y las correcciones de errores realizadas tanto en el cliente Android como en el servidor.

El cuarto capítulo contiene conclusiones y recomendaciones obtenidas durante la realización del presente trabajo.

Finalmente se tiene los anexos, entre los cuales están las encuestas de requerimientos, scripts de la base de datos, código fuente del servicio WCF, código fuente del cliente Android y las encuestas de validación.

PALABRAS CLAVE: Servicio WCF, Tecnología REST, Tecnología Volley, Interfaz

ABSTRACT

The present project consists of the development of a prototype of system to manage football games of Sports League “La Armenia”. This prototype is conformed by two main parts: An Android Client and a WCF (Windows Communication Foundation) service based on REST (Representational State Transfer).

The current document is constituted by four chapters:

The first chapter presents concepts about WCF service based on REST, Android operating system, agile methodology Kanban, AES (Advanced Encription Standard) encryption used to sending and receiving user credentials are presented. Finally, the development tools used are presented.

The second chapter presents the methodology used for development of the current project, which consist of two parts: design and implementation. The first part presents the establishment of users requirements and design of the system components. The coding of the logic of the components is presented in the implementation part.

The third chapter presents the test results done on the prototype’s components. Also, it presents the result of validation surveys completed by the administrative staff and bug fixes made on the Android client and the server.

The fourth chapter contains conclusions and recommendations obtained during the development of the current project.

Finally, there are annexes such as: users requirement survey, database’s script, WCF service source code, Android client source code and the validation survey.

KEYWORDS: WCFServices, REST Technology, Volley Technology, Interface

1. INTRODUCCIÓN

El desarrollo tecnológico incrementa con el pasar del tiempo, ya que la investigación por mejorar y producir sistemas, programas, materiales, prototipos dirigidos hacia el internet de las cosas, la medicina, transporte, medios de comunicación, entretenimiento, entre otros, ha generado progreso en la ciencia y la calidad de vida de las personas.

Uno de estos aportes son las aplicaciones en sistemas móviles o smartphones que permiten un fácil acceso a la información en tiempo real almacenada en distintos servidores. En varias ocasiones se requiere de una conexión a internet para ingresar a aplicaciones como WhatsApp, Facebook, juegos, deporte, entre otros.

Las aplicaciones de entretenimiento enfocadas al deporte como el fútbol tienen millones de descargas a nivel mundial y son utilizadas por los fanáticos que les gusta mantenerse al día en cuanto a sus equipos favoritos, ya que es el deporte más popular y practicado en todo el mundo [1].

Existen casos de éxito de estos sistemas web de equipos reconocidos del mundo como son: *FotMob, Forza Football y OneFootball* [2], sin embargo, uno de los limitantes es que se encuentran desarrollados de forma general sin cumplir con las características propias de un campeonato específico y a menor escala como las ligas barriales.

Por lo anteriormente mencionado se ha desarrollado el presente estudio técnico que consiste en desarrollar un sistema prototipo web en Android para gestionar los partidos de la Liga Deportiva Barrial “La Armenia”.

Esta aplicación es desarrollada con el propósito de mantener actualizados a los usuarios que participan de este deporte en dicha liga, ya que brinda información acerca del calendario de partidos, resultados en línea, tabla de goleadores, jugadores sancionados y el portal de noticias. Toda esta información el usuario la podrá obtener desde cualquier dispositivo móvil.

Cabe mencionar que información importante como la anotación de un gol, inicio y finalización del partido es notificada de forma inmediata a los usuarios.

Toda la información mencionada es almacenada en una base de datos, que una vez gestionada se mostrará en la pantalla del dispositivo móvil.

1.1 Objetivos

El objetivo general de este trabajo de titulación es: desarrollar un sistema prototipo web en Android para gestionar los partidos de una liga barrial de fútbol de Quito.

Los objetivos específicos de este trabajo de titulación son:

- Analizar los conceptos fundamentales para el desarrollo del proyecto propuesto.
- Diseñar los módulos del sistema considerando los requisitos que debe cumplir el prototipo.
- Implementar los módulos del sistema según el diseño realizado.
- Analizar los resultados obtenidos en las pruebas realizadas.

1.2 Alcance

El Sistema Prototipo Web en Android obtiene la información gestionada en la Liga Deportiva Barrial "La Armenia", la misma que consta de 3 categorías: Senior, Femenino e Infantil, a las cuales pertenecen 20, 7 y 10 equipos respectivamente, con un aproximado de 22 jugadores por equipo en la categoría Senior y 15 en las de Femenino e Infantil, los partidos se los juega los sábados y domingos.

La aplicación desarrollada brinda información del calendario de partidos, resultados en línea, tabla de goleadores, jugadores sancionados y un portal de noticias. Toda esta información el usuario la puede obtener desde cualquier dispositivo que tenga un Sistema Operativo[1] Android 2.2.1, solo se necesita una conexión a internet.

Toda la información gestionada es almacenada en una base de datos, la información obtenida se procesa y su resultado será reflejado en la pantalla del dispositivo.

El sistema prototipo web se desarrolla basándose en la metodología ágil Kanban. Además, está conformado por: cliente, que es la interfaz por la cual el usuario manipula la información de la aplicación, Servicio web tipo Restful en WCF (Windows Communication Foundation) que se encarga de la lógica de los partidos, y de una base de datos[2] SQL que almacena la información correspondiente a los partidos.

[1] Software que está compuesto por un conjunto de programas que gestiona los recursos de hardware.
[2] Es un conjunto de datos almacenados sistemáticamente para ser usados posteriormente.

El sistema prototipo web cuenta con los siguientes módulos:

a. Partidos en vivo

Antes de cada partido se tiene la opción de pronosticar al equipo ganador o si el partido termina en empate. El resultado de los pronósticos se muestra en porcentaje según el número de usuarios que votaron. Una vez empezado el partido se lo resalta de un color diferente para diferenciarlo de los partidos que no se estén jugando, el prototipo permite indicar con un cronómetro el avance del partido y su marcador, además se enviarán notificaciones al usuario informando en el momento que se marque un gol o exista una expulsión de un jugador. Existe la opción de votar por el equipo que se considere que ganará el partido a disputar

Cuando el partido termina se muestra el resultado final con los jugadores que marcaron gol y las tarjetas mostradas a los mismos.

b. Interfaz de usuario

La interfaz son las diversas ventanas de la aplicación con las que el usuario interactúa con el sistema web tales como partidos, estadísticas, posiciones, entre otras.

c. Calendario

En el calendario se indica todos los partidos por jugar de cada uno de los equipos, al seleccionar uno de los mismos se obtiene información detallada de su calendario y de los jugadores que conforman el plantel.

d. Tabla de posiciones

Las posiciones de los equipos se actualizan en tiempo real según se vayan dando los resultados de los partidos, una vez finalizada la fecha se indica la cantidad de partidos jugados, goles a favor, goles en contra y el número de puntos acumulados de cada equipo.

e. Noticias

En este módulo estarán las novedades existentes en la liga como jugadores sancionados por tarjeta roja o acumulación de tarjetas amarillas, fechas pospuestas y canceladas.

f. Alineaciones

Antes de cada partido se enviará una notificación para informar al usuario que ya está disponible la alineación con los jugadores que van a ser parte del encuentro.

g. Estadísticas

Las estadísticas se muestran en un gráfico indicando la cantidad de partidos ganados, empatados o perdidos que tiene cada uno de los equipos, además se indica mediante una tabla los jugadores que mayor cantidad de anotaciones hayan marcado a lo largo del campeonato y finalmente se muestra en un gráfico el rendimiento de cada uno de los equipos.

h. Videos

El sistema prototipo cuenta con la opción de subir videos por parte del usuario los mismos que serán filtrados según su contenido por el administrador del sistema.

Además, se puede ingresar un correo electrónico al cual se le enviará de forma recurrente los resultados de los partidos, la tabla de posiciones y la próxima fecha a jugarse.

1.3 Marco Teórico

El presente trabajo de titulación está enfocado en la gestión de los partidos de la Liga Deportiva Barrial "La Armenia" para lo cual se desarrolla este prototipo en Android que es el sistema operativo más utilizado en dispositivos móviles a nivel mundial.

A continuación se repasan algunos conceptos fundamentales que nos permitirán entender de una mejor manera el funcionamiento del sistema prototipo empezando desde la implementación de los servidores hasta llegar al cliente desarrollado en Android.

1.3.1. Windows Communication Foundation (WCF)

Windows Communication Foundation [3] es un framework[3] que nos permite desarrollar aplicaciones orientadas al servicio, su uso nos permite enviar datos desde un endpoint [4] [1] de servicio hacia otro. Este endpoint puede ser parte de un servicio alojado en Internet Information Services (IIS) el mismo que puede tener una disponibilidad continua, o a su vez puede ser un cliente de un servicio que solicita datos de un endpoint de un servicio.

Un endpoint permite comunicarse con el servicio mediante el uso de una determinada tecnología de transporte. El servicio es creado mediante la implementación de una colección de endpoints. Un cliente es un aplicativo que permite intercambiar información con uno o más endpoints.

Un endpoint de un servicio está compuesto de: un address, un binding y un contract.

[3] Estructura conceptual con módulos de software en base a la cual se pueden realizar otros proyectos.
[4] Ubicación física de un punto de contacto proveniente de un servicio web .

- Address: la dirección establece el lugar en donde se encuentra alojado el servicio y donde se envían y reciben los mensajes entre los endpoints.
- Binding: el binding especifica la codificación del mensaje, el protocolo de transporte, las características de seguridad y más detalles que requiere la comunicación entre el cliente y el servidor. La forma más sencilla de comunicación es que tanto el cliente como el servidor tengan el mismo binding.
- Contract: en el contrato se establece los acuerdos con los que el servicio WCF [4]se comunica con las aplicaciones, debe existir por lo menos un contrato para que el servicio pueda recibir las peticiones de los clientes.

WCF define cuatro contratos: Contrato de servicio que define las operaciones que puede realizar el cliente en el servicio, el contrato de datos que define el tipo de datos que se envían desde y hacia el servicio, el contrato de fallos que establece los errores provocados por el servicio y la forma en que los administra y envía al cliente, y el contrato de mensajes que permite que el servicio interactúe directamente con el mensaje.

Arquitectura WCF

Los servicios WCF se encuentran conformados por una arquitectura por capas como se muestra en la figura 1.2.

La capa aplicación hace uso de un lenguaje de programación como C#, en el cual se establecen las funcionalidades que ofrece el servicio, implementando todos los métodos definidos dentro de la capa contrato.

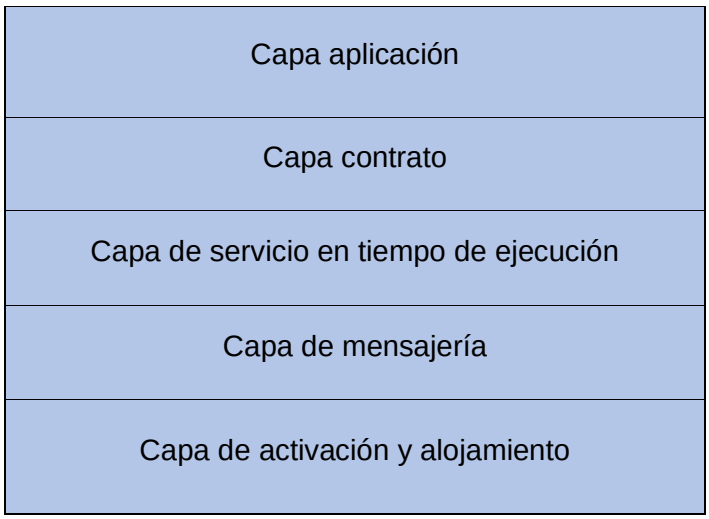


Figura 1.2 Arquitectura WCF

En la capa contrato se establecen los métodos que va a tener el servicio y los tipos de datos que se van a retornar. La capa tiene cuatro tipos de datos: Contrato de datos que establece los datos que van a ser intercambiados por el servicio; contrato de mensajes permite controlar los parámetros de mensajes SOAP; contrato de servicio, permite obtener información de los protocolos de comunicación del servicio; vinculación y políticas, brinda información de los protocolos de seguridad utilizados.

La capa de servicio en tiempo de ejecución permite controlar el comportamiento de los servicios durante el tiempo de ejecución como el comportamiento de los errores, la concurrencia, la cantidad de instancias utilizadas en el servicio, entre otros.

La capa de mensajería establece los canales que pueden ser utilizados para acceder al servicio, como canales seguros que usen un tipo de cifrado, canales TCP, canales HTTP, etc.

La capa de activación y alojamiento utiliza un servidor WAS (Windows Process Activation Service) que permite iniciar aplicaciones que alojen servicios WCF, estableciendo un arranque de forma automática. Para el alojamiento se tienen: Servicios de Windows, cualquier aplicación de .NET, entre otros.

1.3.2. Representational State Transfer (REST)

Es una arquitectura que permite obtener recursos mediante el uso de Uniform Resource Identifier (URI) [5], utiliza el protocolo Hypertext Transfer Protocol (HTTP) lo que la hace compatible con varios lenguajes de programación.

El servicio REST hace que las comunicaciones sean más ligeras, mantenibles y escalables entre el cliente y el servidor además los mantiene independientes entre sí.

REST permite el uso de diferentes formatos de datos como HTML, JSON, XML y archivos en texto plano. Es una arquitectura simple y ligera debido a que no utiliza mensajes especiales.

Características de REST: [6]

- Protocolo cliente/servidor sin estado: cada solicitud tiene la información necesaria para ser procesada por lo que ni el cliente ni el servidor necesitan guardar ningún estado.
- Una única URI identifica a un recurso u objeto REST y nos permite gestionar la información y compartirla con terceros.

- Sistema de capas: es una arquitectura jerárquica y sus capas son administradas según el comportamiento de sus componentes teniendo una funcionalidad específica dentro del sistema REST.
- Hipermedios: la capacidad de una interfaz de desarrollo de aplicaciones de proporcionar al cliente los enlaces adecuados para ejecutar acciones específicas sobre los datos.

La arquitectura REST tiene su propia configuración en cada una de las capas del servicio WCF, en la capa mensajería utiliza el protocolo HTTP o HTTPS para el envío y recepción de datos. En la capa contratos utiliza los métodos, GET que nos permite leer información, POST guarda o crea datos, PUT los edita y DELETE los elimina, estos métodos son propios de HTTP y especifican un contrato uniforme.

La capa de contrato utiliza la librería `System.ServiceModel.Web` que permite implementar esta capa en el *framework* de .NET usando sus respectivos atributos como:

`[ServiceContract]`: Define el contrato del servicio y se lo coloca sobre una clase.

`[OperationContract]`: Define los métodos de un contrato de servicio y se lo coloca sobre los métodos para que puedan ser consumidos por el cliente.

`[WebGet]`: Indica que el método es utilizado para traer información desde el servidor, y se lo puede hacer por medio de WCF basado en REST.

`[WebInvoke]`: Indica que el método puede ser utilizado para crear, editar o eliminar información, y se lo puede hacer por medio de WCF basado en REST.

En la línea 12 del código 1.1 se define el contrato del servicio; En la línea 13 se establece el nombre del contrato de servicio como IWebService; En la línea 14 se específica una sola operación; En las líneas 16 a la 18 se utiliza una operación de obtención de información en formato JSON, relacionada a los usuarios del sistema.

```
12      [ServiceContract]
        1 referencia
13      public interface IWebService
14      {
15          [OperationContract]
16          [WebGet(UriTemplate = "/getUsuario",
17              ResponseFormat = WebMessageFormat.Json)]
            1 referencia
18          List<Usuario> getUsuario();
```

Código 1.1. Definición de contrato de servicio

Para la serialización de datos dentro de una clase se debe utilizar la librería `System.Runtime.Serialization`, la misma que utiliza dos tipos de atributos:

`[DataContract]`: Define los tipos de datos que se van a serializar e intercambiar.

`[DataMember]`: Especifica los tipos de datos que se van a serializar e intercambiar.

En la línea 5 del código 1.2 se establece la librería a utilizar para la serialización; En la línea 8 se define el contrato de datos; En la línea 9 se especifica el tipo de dato llamado Usuario. En las líneas 11 – 21 se especifican los atributos que van a ser serializados.

```
5  using System.Runtime.Serialization;
6  namespace WS_Laliga
7  {
8      [DataContract]
       7 referencias
9      public class Usuario
10     {
11         [DataMember]
12         int idLogin;
13
14         [DataMember]
15         string nombreUsuario;
16
17         [DataMember]
18         string password;
19
20         [DataMember]
21         string tipoUsuario;
22     }
23 }
```

Código 1.2 Contrato de datos

Dentro de la arquitectura WCF se tienen las capas de servicio, mensajería y activación que utilizan un archivo de configuración llamado `Web.config`, el cual usa elementos XML que permiten manipular las características que va a tener el servicio. Dentro de este archivo se encuentra las siguientes etiquetas:

`<configuración>`: Es el elemento raíz del archivo `Web.Config`.

`<system.serviceModel>`: Contiene los elementos de configuración del servicio WCF.

`<behaviors>`: Define los extremos del servicio mediante el uso de los elementos `<serviceBehaviors>` y `<endpointBehaviors>`.

`<serviceBehaviors>` Esta sección de configuración representa todos los comportamientos definidos para un servicio específico.

`<endpointBehaviors>` Esta sección de configuración representa todos los comportamientos definidos para un servicio específico.

`<behavior>`	Es un conjunto de configuraciones del comportamiento que va a tener un servicio.
`<serviceMetadata>`	Especifica la publicación de los metadatos del servicio y la información asociada.
`<serviceDebug>`	Especifica las funciones de depuración y de información de ayuda para un servicio de WCF.
`<webHttp>`	Especifica que el servicio debe tener un comportamiento basado en REST.
`<protocolMapping>`	Sección que permite añadir los bindings y sus correspondientes protocolos de comunicación (HTTP, net.tcp, net.pipe, etc) empleados por el servicio.

A continuación, se muestra ejemplos en código de uno de los métodos HTTP el cual es un método en el que se apoya REST:

1. Método Post: permite ingresar nuevos registros en el sistema.

```
[OperationContract]
[WebInvoke(UriTemplate = "/savePartidos",
    Method = "POST")]
int savePartido(Partidos savePartidos);
```

Código 1.1. Ejemplo método Post

2. Método Get: permite obtener recursos desde el sistema.

```
[OperationContract]
[WebGet(UriTemplate = "/getUsuario")]
List<Usuario> getUsuarios();
```

Código 1.2 Ejemplo método Get

El formato que se utiliza en REST para representar los recursos es JSON que es bastante útil para el intercambio de datos y puede ser utilizado en cualquier lenguaje de programación. Los datos en JSON viajan como texto agrupados en pares de nombre y valor como se muestra en el código 1.3, en la línea 2 se define el nombre del objeto en la línea 4 se tiene como nombre “idUsuario” y de valor el “0” al igual que en las líneas 5 a la 7 se tienen los pares nombre/valor que son parte de los atributos de un Usuario.

```
{
"saveUsuario":
    {
        "idUsuario":0,
        "nombreUsuario":"José",
        "password":"Ortega",
        "tipoUsuario":"Administrador"
    }
}
```

Código 1.3 Ejemplo de un objeto JSON

Con JSON también se puede enviar datos como un arreglo estructurado de texto, como se muestra en el código 1.4. En la línea 2 se establece el nombre del arreglo y de las líneas 4 a la 19 se encuentran 3 objetos con pares nombre y valor que corresponden a los atributos de un usuario.

```
{
"saveUsuario":[
    {
        "idUsuario":0,
        "nombreUsuario":"José",
        "password":"Ortega",
        "tipoUsuario":"Administrador"
    },
    {
        "idUsuario":1,
        "nombreUsuario":"Ernesto",
        "password":"Muzo",
        "tipoUsuario":"Vocal"
    },
    {
        "idUsuario":2,
        "nombreUsuario":"Marcelo",
        "password":"Aguinaga",
        "tipoUsuario":"Vocal"
    }]
}
```

Código 1.4 Ejemplo de un arreglo JSON

1.3.3. Sistema Operativo Android

Android [7] es un sistema operativo creado por Google y desarrollado para dispositivos móviles, utiliza código abierto basado en Linux.

1.3.4. Arquitectura de Android

En la siguiente figura se muestran los componentes principales de la arquitectura actual de Android [8]:

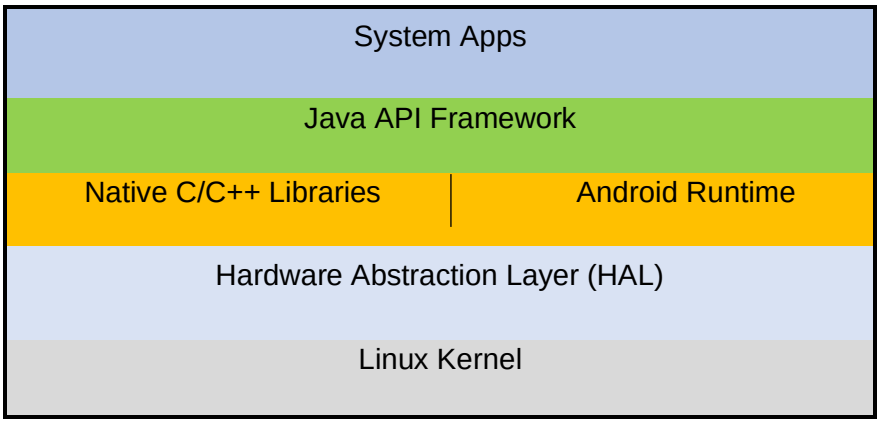


Figura 1.1. Arquitectura de Android [9]

a. **System Apps:** Android viene con algunas aplicaciones instaladas por defecto y posteriormente el usuario podrá agregar más aplicaciones que pueden ser de terceras empresas o de su propio desarrollo.

b. **Java API Framework:** aquí se encuentran las herramientas para el desarrollo de una aplicación.

c. **Native C/C++ Libraries:** aquí se encuentran la mayoría de las librerías de Android desarrolladas en los lenguajes C/C++ las mismas que brindan una gran parte de sus características.

d. **Android Runtime:** a partir de la versión 5.0 todas las aplicaciones ejecutan sus procesos con su respectivo tiempo de ejecución.

e. **Linux Kernel:** el kernel de Linux permite crear controladores de hardware conocidos para los desarrolladores de dispositivos. La generación de subprocesos y gestión de memoria reducida es desarrollada por el kernel de Linux.

1.3.5. Kit de desarrollo de Android

Software Development Kit (SDK) [10] por sus siglas en inglés, es un conjunto de herramientas cuya finalidad es facilitar el desarrollo de aplicaciones en Android. Entre las herramientas utilizadas se tiene un depurador de código, biblioteca, un simulador de un dispositivo Android basado en QEMU, documentación, tutoriales y acceso a ejemplos de código. Estas herramientas pueden ser utilizadas en Linux, Mac y Windows.

Características SDK

- Android hace uso de un puente de depuración llamado Android Debug Bridge que es un conjunto de herramientas incluidas en el SDK utilizadas para la depuración de las aplicaciones cuya función es comunicar al cliente con el servidor.
- El SDK de Android usa el protocolo Fastboot cuya función es modificar el sistema de ficheros flash que permite generar instrucciones para optimizar el arranque del sistema. Estas instrucciones son enviados por comandos dentro de los más usados se encuentran los siguientes:
    - Flash: sobrescribe una partición especifica insertando una imagen binaria y la almacena en un computador.
    - Erase: elimina una partición específica.
    - Reboot: reinicia el dispositivo con el sistema principal, en la partición de recuperación del sistema.
    - Devices: indica el número de serie de todos los dispositivos que se encuentran conectados al ordenador.
    - Format: da formato a una partición en específico.

- El SDK contiene un Native development kit que permite instalar librerias desarrolladas en los lenguajes C,C++, entre otros, esta librerias pueden ser utilizadas en la aplicación que estamos utilizando para ahorrar tiempo y líneas de código.

1.3.6. Base de datos

Una base de datos es una serie de datos organizados de tal manera que se facilite su gestión [11].

Las bases de datos brindan las siguientes características:

a. **Redundancia de datos:** Los datos y varias copias de estos se almacenan en distintos ficheros generando redundancia, pero se podría generar inconsistencias en los datos debido a que no todas las copias redundantes contienen la misma información.

b. **Compartir datos:** los datos de un fichero pueden ser compartido con todos los clientes que tengan los permisos.

c. **Consistencia en los datos:** para mantener consistencia en los datos, toda transacción debe pasar de un estado válido a otro, es decir que debe cumplir todas las reglas definidas.

d. **Integridad en los datos:** una base de datos garantiza validez y consistencia en los datos mediante el uso de restricciones que se deben cumplir en la información que se encuentra almacenada.

e. **Seguridad:** las bases de datos brindan protección a la información impidiendo el acceso a personas no autorizadas.

1.3.6.1. Modelo de datos

El modelado de datos genera una estructura que permite almacenar los datos de un usuario final. Un modelo de datos generalmente permite representar de forma gráfica situaciones reales complejas.

Los elementos básicos de los modelos de datos son: entidades, atributos, relaciones y restricciones.

a. **Entidad:** es una persona, lugar, cosa o hecho que se identifica de manera única en la cual se almacenan sus datos.

b. **Atributo:** es una característica de una entidad.

c. **Relación:** describe la forma que se asocian las entidades. Los modelos de datos usan tres tipos de relaciones: uno a uno, uno a mucho y muchos a muchos.

   - **Relación 1:1:** cada fila de una tabla está asociada a una fila de otra tabla.
   - **Relación 1:M:** cada fila de una tabla está asociada a muchas filas de otra tabla.
   - **Relación M:N:** muchas filas de una tabla están asociadas con muchas filas de otra tabla. Para su representación en el modelo se crea una nueva tabla que se asocia con relaciones de 1:M con las tablas originales.

d. Restricciones: ayudan a proteger la integridad de los datos.

1.3.7. Metodología Kanban

Kanban es una metodología que permite organizar las tareas de un proyecto manteniendo la secuencia de las mismas, con la finalidad de optimizar la ejecución en un tiempo previamente establecido [12].

Para llevar a cabo dicha metodología se utiliza el tablero denominado Kanban, conformado por un grupo de tarjetas que representan las tareas o módulos de un proyecto o sistema. Dichas tarjetas son ubicadas dentro de la tabla de acuerdo al avance o estado de trabajo en el que se encuentran. Estos estados se encuentran establecidos en las columnas del tablero Kanban como se muestra en la figura 1.3.

Figura 1.2. Ejemplo de Tablero Kanban [13]

Hay varios ejemplos de tableros Kanban que varían según el objetivo del trabajo, teniendo diferentes diseños de tableros según el alcance del trabajo. Para alcanzar el objetivo de este trabajo se va a utilizar cuatro columnas: pendiente, en proceso, pruebas y finalizado.

Las ventajas de esta metodología son:

a. Visualizar el estado de las tareas lo que permite planificar y evitar el retraso de las mismas, ya que esta metodología accede a una buena organización y distribución del trabajo.

b. Establecer la cantidad de tareas para evitar sobrecargar el trabajo y cumplir con los objetivos establecidos.

c. Permite la fácil aplicación de indicadores visuales como tarjetas de colores para distinguir tareas con más prioridad que otras y permitir una lectura fácil.

d. Identifica los problemas existentes a tiempo permitiendo revisar el avance del trabajo para aumentar el rendimiento.

1.3.8. Cifrado Advanced Encryption Standard (AES)

Es un algoritmo de cifrado simétrico, cuyo análisis requiere de una gran cantidad de textos cifrados y gran procesamiento, características que lo hacen un algoritmo bastante seguro ya que para poder descifrarlo es necesario el uso de un billón de ordenadores y que cada uno pueda probar un millón de claves por segundo.

Características adicionales al algoritmo se muestran a continuación:

- Su cifrado se basa en una matriz de estado.
- Tiene tamaños de 128, 192 y 156 bits.
- Se realiza con el operador “exclusive or” (XOR).
- Su operación se conoce como “AddRoundKey”.

El cifrado de un AES tiene 3 etapas:

1 Confusión: para esta etapa se hace uso del cifrado “César” que permite ocultar la relación entre el mensaje real y el cifrado.

2 Difusión: se realiza una transposición de columnas que permite esparcir el mensaje.

3 Clave secreta: el método puede ser descubierto en cualquier momento, pero se podría complicar con el uso de una clave diferente en cada comunicación cliente – servidor.

El uso de este algoritmo aumenta la seguridad en el intercambio de datos entre el cliente y servidor, especialmente en los registros de usuarios y los inicios de sesión en los perfiles de “Administrador” y “Vocal”, encriptando sus respetivos nombres de usuarios y contraseñas.

1.3.9. Herramientas de desarrollo

Las herramientas utilizadas para el desarrollo del sistema que permitieron la codificación de la base de datos y del servidor WCF son:

- **Android Studio**: es un entorno de desarrollo de aplicaciones para Android y está basado en IntelliJ IDEA, que genera respuestas en menor tiempo en el flujo de codificación y ejecución. Cada aplicación de Android Studio contiene diversos módulos con archivos de códigos y recursos que permiten almacenar en la base de datos. Además, ofrece funciones como: editor de código inteligente, al pulsar instant run se aplicarán los cambios realizados, el emulador se instala e inicia la aplicación en menor tiempo, etc.
- **Microsoft Visual Studio:** permite el desarrollo de aplicaciones web y sitios web en cualquier entorno compatible con la plataforma .NET, es decir, que permite que las aplicaciones se comuniquen con dispositivos móviles, páginas web, estaciones de trabajo, entre otras. Además, este entorno permite trabajar con sistemas operativos como Linux, Mac OS y Windows [15] que en este caso será utilizado para el desarrollo del prototipo que permite crear, editar y depurar códigos, así como publicar una aplicación.
- **SQL Server Management Studio(SSMS):** este sistema permite gestionar la base de datos, es decir, permite crear, leer, modificar y eliminar la información que se encuentre almacenada en la base de datos. Además, contiene un medio gráfico para el uso de comandos del Lenguaje de Base de Datos (DDL) y Lenguaje de Manipulación de Datos (DML). SSMS permite administrar datos de otros servidores.

1.3.10. Investigación Aplicada

Investigación que da solución a problemas específicos, resolviendo problemas de una sociedad o una organización. Adquiere conocimiento para llevarlo a la práctica dando solución a aspectos del mundo moderno.

Objetivo de la investigación aplicada: su objetivo es predecir el comportamiento específico de un determinado sector, con el fin de poner en práctica el conocimiento teórico y ser capaz de proyectarlo y desarrollarlo para su uso en la vida real.

2. METODOLOGÍA

El presente proyecto técnico se desarrollará empleando una investigación de tipo aplicada, basada en llevar a la práctica los conocimientos adquiridos con respecto al desarrollo de servicios WCF basados en JSON, desarrollo de bases de datos relacionales y aplicaciones

móviles Android. Se contará con un prototipo que permitirá gestionar los partidos de la Liga Deportiva Barrial “La Armenia”.

La recolección de los requerimientos de usuario se la realizará mediante encuestas, las mismas que serán analizadas para establecer los requerimientos del usuario. Este proceso permitirá establecer los requerimientos funcionales y no funcionales.

En el desarrollo del sistema se utilizará la metodología Kanban por la fácil planificación, control y distribución de cada proceso, por lo cual las tareas se desarrollan en función a los requerimientos del usuario. Este tablero estará formado por 4 columnas que son: entrada, implementación, pruebas y salida.

En la columna entrada se colocarán los módulos que tendrá el prototipo, en implementación se ubicarán las tareas que se requieren para la construcción de los módulos. En la columna pruebas se indicarán las tareas esenciales para validar el funcionamiento de los módulos y finalmente en salida se colocará los entregables de cada módulo del sistema.

Es importante mencionar que el sistema prototipo se diseñará considerando dos partes. La primera es la base de datos para la cual se realizará un diagrama de entidad – relación, con la finalidad de identificar las entidades, atributos y sus respectivas relaciones. La segunda es la aplicación web, para el cual se elaborará un diagrama de clases el cual describe la estructura de un sistema.

Además, para las interfaces del cliente Android se utilizarán sketches y wireframes que permitan una fácil comprensión de la aplicación web.

En cuanto a la implementación, se llevará a cabo en 3 etapas: la primera consiste en un procesamiento de información la misma que es obtenida de las encuestas realizadas a los miembros de la liga barrial. La segunda etapa se refiere a la elaboración de la base de datos para lo cual se utilizará Microsoft SQL Server Management Studio 17 en el cual se codificará con scripts en SQL para la creación de tablas. En la tercera etapa consiste en el desarrollo del prototipo web que utilizará la tecnología .Net Framework 4.6 y el lenguaje de programación C#.

2.1. Arquitectura del prototipo

La arquitectura del prototipo es del tipo cliente – servidor, es decir, el cliente requiere de un dispositivo móvil de sistema operativo Android, que, al contar con la aplicación instalada y acceso a internet, se conectará al servidor web [16] desde cualquier parte del mundo. En la figura 2.1 se muestra la interacción entre el cliente Android y el servicio WCF. El cliente Android envía una solicitud por medio del protocolo HTTP que permite conectarse a la red

de internet, dicha petición llega al servicio WCF es procesada y enviada hacia el servidor de base de datos por medio del protocolo ODBC, este realiza la consulta solicitada y devuelve la respuesta al servidor para posteriormente enviarla al cliente Android.

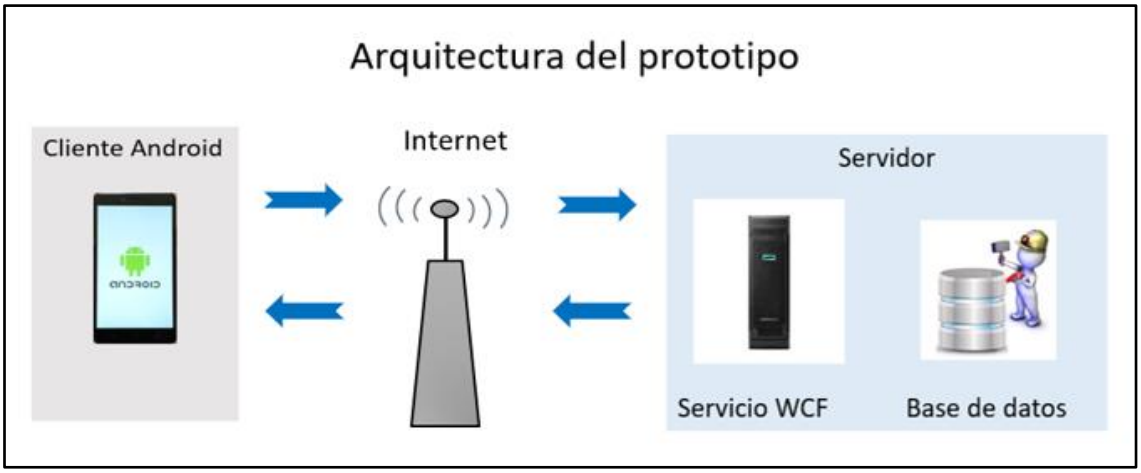


Figura 2.1. Arquitectura del prototipo

El sistema web está conformado por el cliente Android y el servidor web. Este último a su vez está compuesto por el servicio WCF y la base de datos como se muestra en la figura 2.1. El cliente al ingresar en el prototipo empezará a enviar peticiones al servidor web, en el cual se aloja el servicio WCF. Estas peticiones se envían en forma de mensajes encapsulados en formato JSON a través del protocolo HTTP, viajan por la red de internet llegan al servidor WCF y posteriormente se realiza la consulta en la base de datos del servidor SQL como se muestra en la figura 2.2.

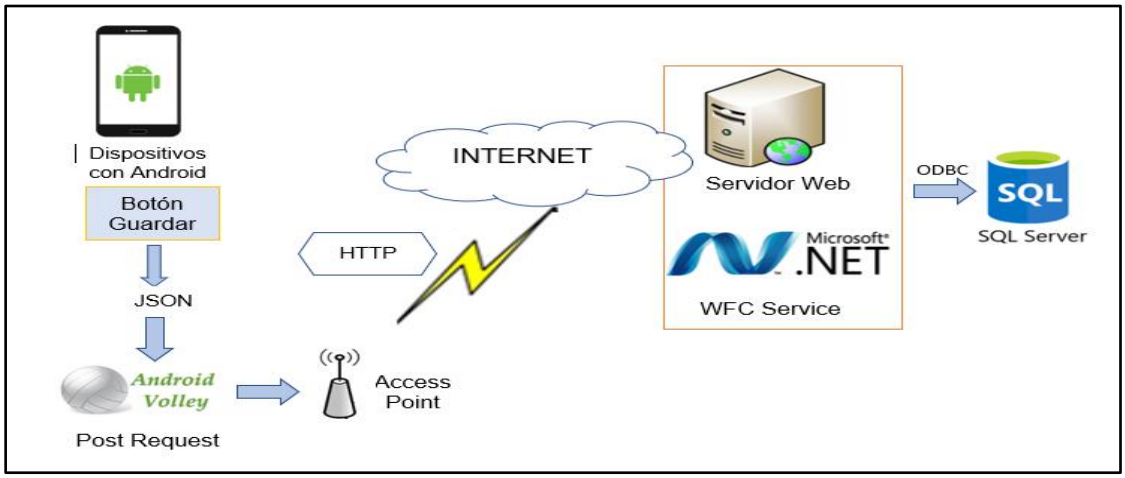


Figura 2.2. Envío de Post Request

Una vez procesada la petición por parte del servidor[5], este le responderá al usuario con la información solicitada realizando el proceso inverso al mostrado en la figura anterior como se muestra en la figura 2.3.

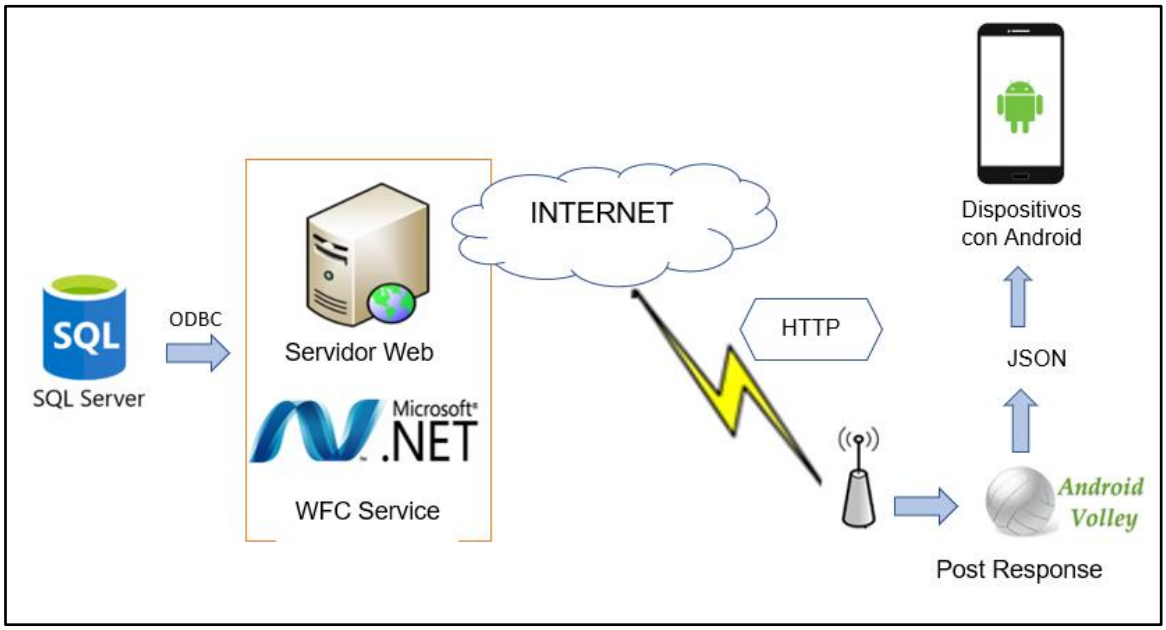


Figura 2.3 Respuesta Post Response

2.1. Entrevistas

En el presente proyecto se realizaron 15 entrevistas mediante encuestas a los administradores, deportistas e hinchas que forman parte de la Liga Deportiva Barrial "La Armenia", con la finalidad de establecer las historias de usuario y los requerimientos funcionales y no funcionales.

Las entrevistas se realizaron de forma personal en las instalaciones de la liga barrial, cada una de ellas contienen 9 preguntas que tuvieron como objetivo sacar los requerimientos de usuario. El formato de dichas entrevistas se establece en el anexo A y a continuación se indican los datos obtenidos:

a. La primera pregunta permite saber si los usuarios pertenecen a un equipo de una liga barrial. En la figura 2.4 se muestra que el 92% de los encuestados forman parte de un equipo de una Liga barrial, resultado que permite determinar que es un área interesada en el fútbol de la liga barrial.

[5] Ordenador que permite tener acceso a los recursos compartidos entre las estaciones de trabajo.

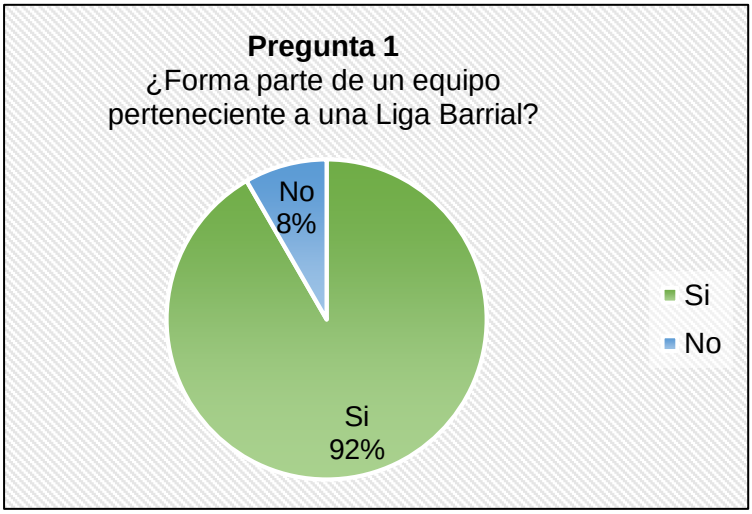


Figura 2.4. Respuesta a la primera pregunta

La segunda pregunta permite saber si los usuarios tienen un dispositivo móvil Android con acceso a Internet. En la figura 2.5 se muestra que el 83% de los encuestados disponen de un dispositivo móvil con Android, resultado que permite determinar la viabilidad de elaborar un prototipo con el sistema operativo Android.

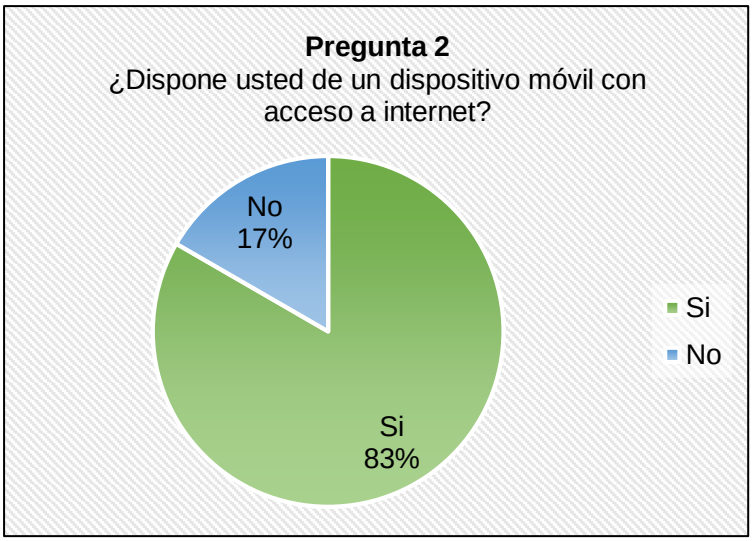


Figura 2.5. Respuesta a la segunda pregunta

b. La tercera pregunta permite saber si los usuarios han utilizado las aplicaciones 365 Scores[6] o OneFootball[7] o similares. En la figura 2.6 se indica que el 79% de los encuestados han usado una aplicación de resultados de fútbol, resultado que permite determinar el gran uso e interés que tiene el sector por aplicaciones relacionadas con el fútbol.

[6] 365Score: [En línea]. Disponible en: https://www.365scores.com/es-mx
[7] OneFootball: [En línea]. Disponible en: https://onefootball.com/co/inicio

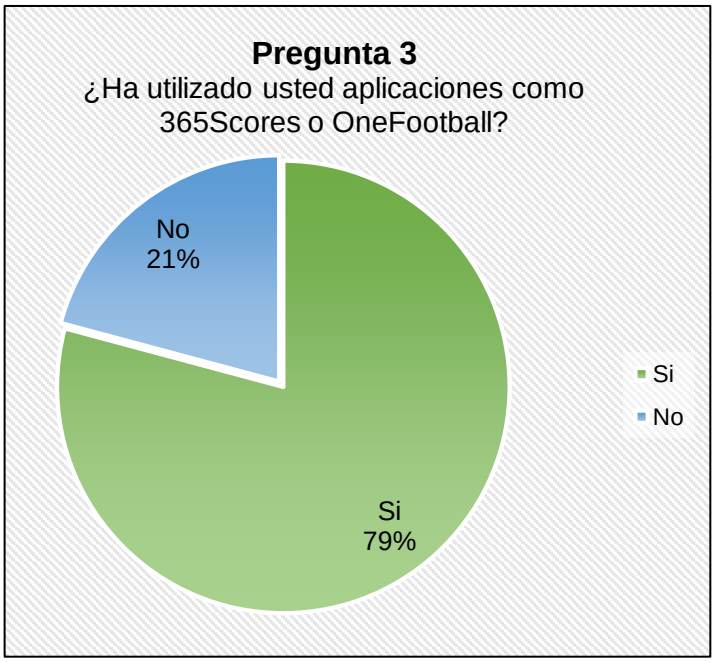


Figura 2.6. Respuesta a la tercera pregunta

c. La cuarta pregunta permite saber si a los usuarios les interesa estar informados de los partidos que disputa su equipo. En la figura 2.7 se indica que el 100% de los encuestados están interesados en tener información de sus equipos, resultado que confirma que es factible el desarrollo del presente trabajo.

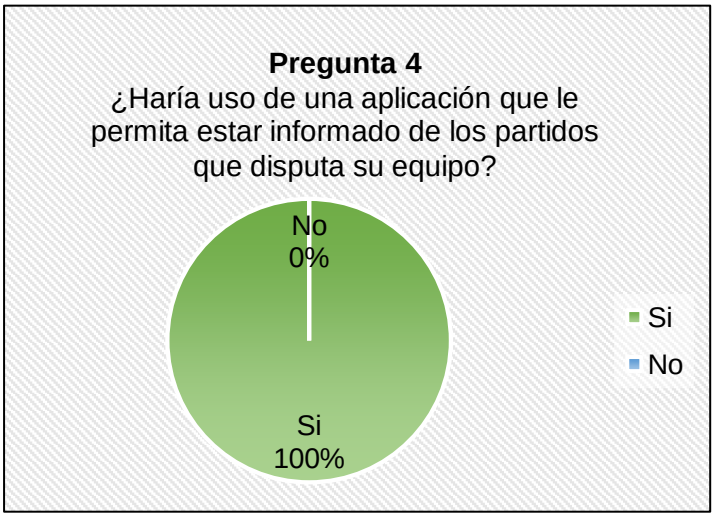


Figura 2.7. Respuesta a la cuarta pregunta

d. La quinta pregunta permite saber cuáles de las opciones presentadas son las más relevantes. En la figura 2.8 se indica que las opciones más relevantes son: Tabla de posiciones con un 29%, Resultados de los partidos con un 29% y Calendario de los partidos con un 28%, resultado que permite determinar que los tres módulos (Tabla de posiciones, resultados y calendario) deben estar en el prototipo.

Figura 2.8. Respuesta a la quinta pregunta

e. La sexta pregunta permite saber si para el usuario es necesario que la aplicación tenga un manual incluido. En la figura 2.9 se indica que el 96% de los encuestados harían uso de dicho manual, resultado que obliga a realizar un manual de usuario para facilitar el uso del prototipo al cliente.

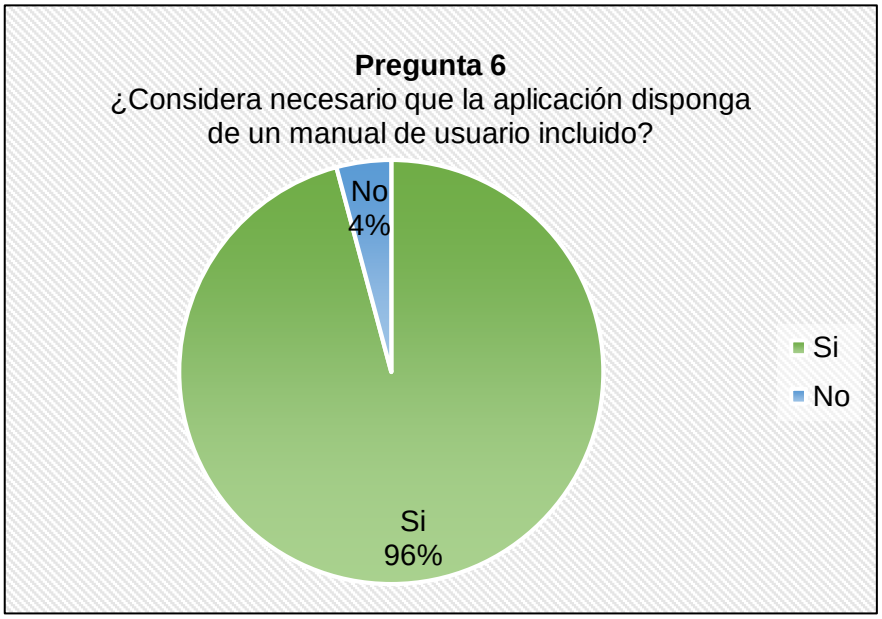


Figura 2.9. Respuesta a la sexta pregunta

f. La séptima pregunta permite saber si el usuario acepta recibir notificaciones de la aplicación. En la figura 2.10 se indica que el 100% de los encuestados no les molesta recibir notificaciones, resultado que determina la importancia de las notificaciones acerca de los eventos de la liga barrial.

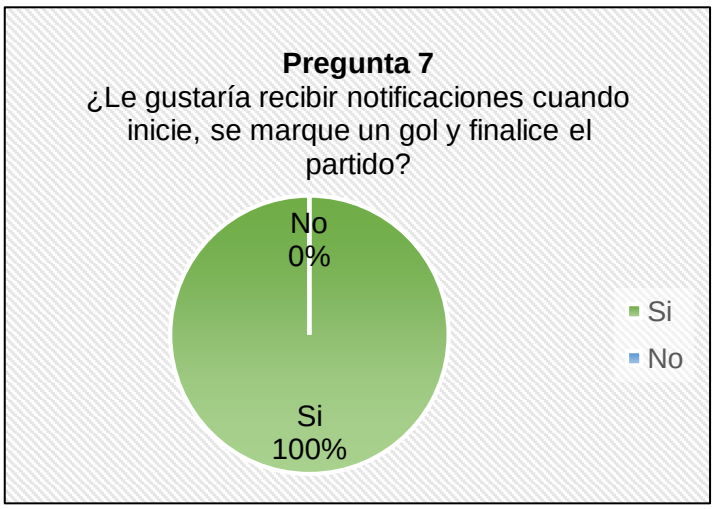


Figura 2.10. Respuesta a la séptima pregunta

g. La octava pregunta permite saber si al usuario le gustaría tener la opción de predecir el marcador de los partidos a disputarse. En la figura 2.11 se indica que el 96% de los encuestados aceptan tener la opción de predecir los partidos, resultado que permite agregar una seccion para predecir el partido.

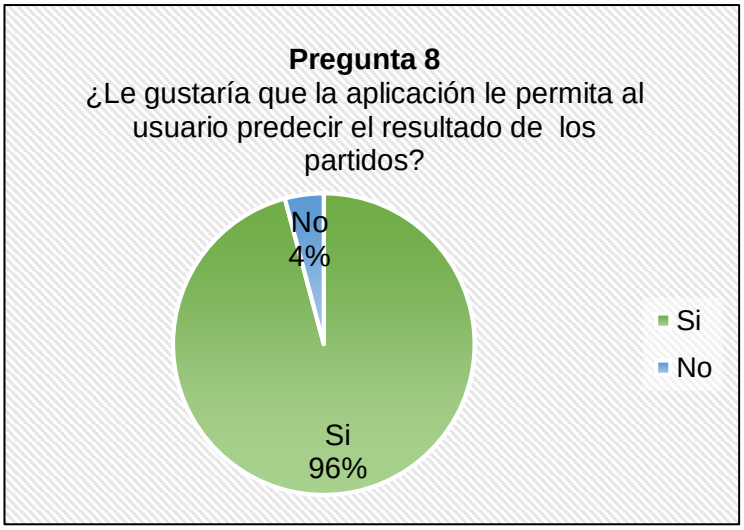


Figura 2.11. Respuesta a la octava pregunta

h. La novena pregunta permite saber si a los usuarios les gustaría compartir con la aplicación videos relacionados con sus equipos. En la figura 2.12 se indica que el 67% de los encuestados compartirían videos con la aplicación, resultado que permite agregar una sección de videos compartidos por el usuario.

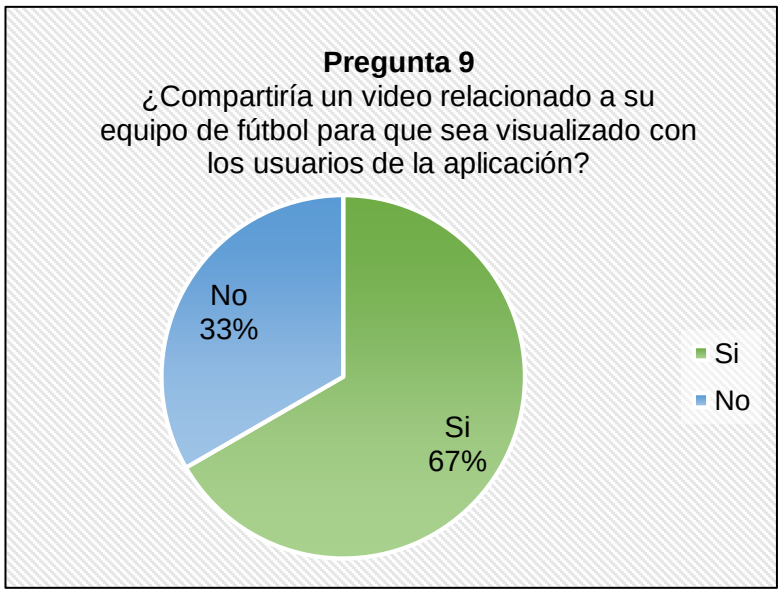


Figura 2.12. Respuesta a la novena pregunta

A continuación, se muestra el resumen de los requerimientos obtenidos en las entrevistas:

Tabla 2.1. Resumen de entrevistas

N°	Pregunta	Resultados
1.	¿Forma parte de un equipo perteneciente a una Liga Barrial?	
	• Si	92%
	• No	8%
2.	¿Dispone usted de un dispositivo móvil con acceso a Internet?	
	• Si	83%
	• No	17%
3.	¿Ha utilizado usted aplicaciones como 365 Scores o OneFootball?	
	• Si	79%
	• No	21%
4.	¿Haría uso de una aplicación que le permita estar informado de los partidos que disputa su equipo?	
	• Si	0%
	• No	100%
5.	De las opciones que se presentan a continuación ¿Cual o cuales considera que son más relevantes?	
	• Alineación	6%
	• Goleadores	8%
	• Calendario	28%
	• Posiciones	29%
	• Resultados	29%
6.	¿Considera necesario que la aplicación disponga de un manual de usuario incluido?	
	• Si	96%
	• No	4%
7.	¿Le gustaría recibir notificaciones cuando inicie, se marque un gol y finalice el partido?	
	• Si	100%
	• No	0%
8.	¿Le gustaría que la aplicación le permita al usuario predecir el resultado de los partidos?	
	• Si	96%
	• No	4%

9.	¿Compartiría un video relacionado a su equipo de fútbol para que sea visualizado con los usuarios de la aplicación?	
	• Si	67%
	• No	33%

2.2. Historias de usuario

A partir de las historias de usuario se obtienen los requerimientos establecidos por parte del usuario, se establecieron mediante las encuestas y el estudio de la aplicación 365 Scores.

Las historias de usuario serán descritas mediante el siguiente formato e indicando su nivel de prioridad:

- ID: permite identificar a la historia de usuario el mismo que empieza con las letras HU y seguido de un número.
- Título: permite identificar a la HU mediante el uso de nombres
- Descripción: detalla las funciones que realiza la HU.
- Prioridad: indica la importancia de cada HU establecida según el porcentaje obtenido en las preguntas realizadas en las entrevistas. La calificación se muestra en la tabla 2.2.

Tabla 2.2. Nivel de prioridades de HU

Prioridad baja	Prioridad media	Prioridad alta
1	2	3

A continuación, se establecen las historias de usuario en la tabla 2.3.

Tabla 2.3. Historia de usuario

Id	Título	Descripción	Prioridad
HU01	Pantalla principal	El usuario visualizará la pantalla principal y tendrá dos pestañas: Partidos y Noticias.	3
HU02	Calendario de partidos	Se mostrarán todos los partidos que están por disputarse y se resaltarán los que se están jugando en tiempo real.	3
HU03	Noticias	Se indicará a los jugadores que hayan acumulado tarjetas rojas o amarillas, los mismos que no estarán disponibles para el próximo encuentro.	2

HU04	Detalles de partidos	El usuario al escoger un partido del calendario podrá visualizar los detalles del partido, la tabla de posiciones y las estadísticas de los equipos.	3
HU05	Fechas de partidos	Se visualizará el tiempo transcurrido del partido o la hora de inicio del mismo, la alineación de los equipos, y una opción para predecir los partidos.	2
HU06	Tabla de posiciones	Se visualizará la tabla de posiciones de todos los equipos detallando la cantidad de puntos, goles a favor, goles en contra y gol diferencia.	3
HU07	Estadísticas	Se visualizará el rendimiento de los equipos según la cantidad de puntos conseguidos.	1
HU08	Videos	Se permite al usuario compartir videos con la aplicación los mismos que serán mostrados en la sección de videos.	1
HU09	Detalle de equipos	Al seleccionar un equipo se enviará a un nuevo *activity* que indicará todos los jugadores que conforman el equipo, y los resultados de los últimos 5 partidos.	2
HU10	Detalle de jugadores	Al seleccionar un jugador se enviará a un nuevo *activity* en el que se muestra el nombre, edad, número de camiseta, tarjetas amarillas y rojas, número de goles marcados y el equipo en el que juega.	2
HU11	Vocalía	El tipo de usuario “Vocal” ingresará las actualizaciones que se presenten durante el desarrollo del partido en tiempo real.	3
HU12	Administración Campeonato	El tipo de usuario “Administrador” puede editar la información de los Campeonatos.	2
HU13	Administración Equipo	El tipo de usuario “Administrador” puede editar la información de los Equipo.	2
HU14	Administración Jugador	El tipo de usuario “Administrador” puede editar la información de cada Jugador.	2
HU15	Administración Partidos	El tipo de usuario “Administrador” puede editar la información de los Partidos.	2
HU16	Administración Noticias	El tipo de usuario “Administrador” puede editar la información de las Noticias.	2
HU17	Administración Posiciones	El tipo de usuario “Administrador” puede editar la información de las Posiciones	2

2.3. Requerimientos de usuario

Para determinar los módulos que va a tener el prototipo se consideró las encuestas mencionadas con anterioridad, en las que se encuentra incluida la as solicitudes realizadas por parte de la administración de la Liga Deportiva Barrial “La Armenia” y se tuvo como ejemplo la aplicación 365 Scores publicada en el Play Store de Google en la fecha 18/10/2018. Estos datos permitieron establecer los requerimientos funcionales y no funcionales.

Estos requerimientos se planificaron y organizaron en base a la metodología Kanban que permitió el desarrollo del prototipo considerando las tareas que se requieren para el diseño y la construcción de los módulos del prototipo hasta la implementación del mismo.

2.3.1. Requerimientos funcionales

Los requerimientos funcionales especifican los servicios que los módulos del sistema deben cumplir [17], los cuales se presentan a continuación:

- El sistema en pantalla principal tendrá dos pestañas una de los partidos a disputar y otra de noticias.
- El sistema permitirá visualizar en la pestaña de partidos todos aquellos que están por disputar con su respectivo horario.
- Se tendrá una sección de noticias en la que se mostrará información de las tarjetas amarillas y rojas que un jugador ha acumulado para saber si está suspendido para el próximo encuentro.
- Los partidos que se están jugando serán resaltados para identificarlos de los partidos que no lo están.
- Al seleccionar un partido nos enviará hacia un nuevo *activity*[8] el mismo que tendrá 3 pestañas: Partido, Tabla de posiciones y Estadísticas.
- Se tendrá una sección Partido en la que se mostrará el tiempo transcurrido y el marcador del encuentro, o a su vez la hora en la que se lo va a disputar.

[8] Componente que contiene una pantalla que permite interactuar con el usuario para que la aplicación realice una acción.

- Se tendrá la opción de predecir el encuentro indicando al equipo ganador o si el partido termina en empate. Una vez elegido al equipo se muestra en porcentaje las votaciones de otros usuarios.
- Se podrá visualizar la alineación de jugadores que cada uno de los equipos ha dispuesto.
- Al seleccionar un equipo se enviará hacia un nuevo *activity* en el cual se debe enlistar los jugadores que forman parte del equipo y los resultados de los últimos 5 partidos.
- Al seleccionar un jugador se enviará hacia un nuevo *activity* en el cual se detallará la información del jugador como el equipo en el que juega, número de camiseta, edad, goles marcados, asistencias, tarjetas amarillas y rojas.
- Se tendrá la opción de compartir videos creados por el usuario que sean relacionados con los partidos que se disputan en la liga barrial.
- Al seleccionar videos se enviará hacia un nuevo *activity* el mismo que tendrá los videos subidos por los usuarios y las instrucciones a seguir para enviar un video.
- Al enviar un video nos pedirá que ingresemos los siguientes datos: nombre, apellido y equipo del cual es hincha.
- Se tendrá la opción de tabla de posiciones se muestran los puntos, goles a favor, goles en contra, gol diferencia y partidos jugados los mismos que permitirán definir la posición en la que se encuentran cada uno de los equipos.
- Se tendrá la sección de Estadísticas en la que se encontrará un gráfico en el cual se indicará el rendimiento de los equipos según los puntos conseguidos.
- Se mostrará una tabla en la cual se indicará la cantidad de goles que han marcado cada uno de los jugadores de sus respectivos equipos.
- Se podrá crear, modificar, ver y eliminar los campeonato, equipos, jugadores, partidos, noticias y posiciones con el tipo de usuario “Administrador”.

2.3.2. Requerimientos no funcionales

Los requerimientos no funcionales se refieren a las características generales del sistema e implementación de la aplicación [18] como se menciona a continuación:

- La aplicación se desarrollará en el sistema operativo Android.

- Los usuarios deberán disponer de Internet para acceder a la aplicación.
- La información se almacenará en una base de datos relacional.
- Se obtendrán los datos por medio de un servicio REST WCF.

2.3.3. Módulos del Prototipo

La agrupación de los requerimientos de usuarios permite determinar los módulos que va a tener el prototipo, cada uno de los módulos va a tener su propia interfaz de usuario. El prototipo contará con los siguientes módulos:

a. Partidos en vivo

Este módulo permite estar actualizado en cuanto a los avances desarrollados durante el partido, indicando el tiempo transcurrido y el marcador de cada partido. Se podrá visualizar los cambios realizados y los goles marcados por cada uno de los equipos. Cada vez que surja un evento (inicio del partido, gol, sustitución, fin de partido) se enviará una notificación para informar al usuario de dicho evento. Dentro de este módulo hay una opción que permite realizar un pronóstico del equipo que se considere ganador.

b. Calendario

Este módulo permite obtener información de los partidos que va a disputar cada uno de los equipos indicando la categoría (infantil, femenino, senior) a la que pertenecen, así como la fecha y hora de inicio del partido.

c. Tabla de posiciones

Este módulo se va a ir actualizando en tiempo real según varíe los resultados de cada uno de los partidos, indicando la posición, partidos jugados, goles a favor, goles en contra y los puntos que tiene cada uno de los equipos.

d. Noticias

Este módulo se indicará los jugadores que se encuentren sancionados por acumulación de tarjetas amarillas o por tarjeta roja mostrada en el partido previo. Se indicará cualquier novedad que se presente en cuanto a la liga barrial corresponde.

e. Alineaciones

En este módulo se presentará las alineaciones que tiene cada uno de equipos para el partido que se va a disputar, indicando cada una de las posiciones (arquero, defensa, volante, delantero) que va a tener cada uno de los jugadores dentro del campo de juego.

Se enviará una notificación al usuario indicando que ya está disponible la alineación del partido.

f. Estadísticas

En este módulo se indicará mediante un gráfico de barras el rendimiento de cada uno de los partidos según la cantidad de puntos alcanzados. Se mostrará la tabla de goleadores a los jugadores que hayan realizado la mayor cantidad de anotaciones ordenados de mayor a menor de cada uno de los equipos que conforman el campeonato actual.

g. Videos

En este módulo se tiene la opción de compartir videos por parte del usuario , para la cual deberá ingresar sus datos, indicar el equipo del cual es hincha y el enlace del video subido a YouTube, este video será mostrado dentro de la aplicación para que pueda ser disfrutado por otros usuarios.

Finalmente, el prototipo contará con una sección de correo electrónico que permitirá enviar de forma recurrente los resultados de los partidos, la tabla de posiciones y la próxima fecha a jugarse a cada uno de los usuarios registrados.

En la tabla 2.4 se muestra las historias de usuario que componen cada uno de los módulos del sistema.

Tabla 2.4 Historias de usuario que componen el cada uno de los módulos del sistema

Módulo	Historias de usuario
Módulo Partidos	HU01
	HU02
	HU04
	HU05
Módulo Interfaz de usuario	HU09
	HU10
Módulo Calendario	HU01
Módulo Tabla de Posiciones	HU04
	HU06
Módulo Noticias	HU03
Módulo Alineaciones	HU05
	HU04

Módulo Estadísticas	HU07
Módulo Videos	HU08
Módulo Usuario	HU11
	HU12
	HU13
	HU14
	HU15
	HU16
	HU17

2.4. Tablero Kanban

El tablero Kanban contiene tres grupos principales, en los que se clasificaron las tareas previamente definidas [14]. Estos grupos se describen a continuación:

- **Funcionalidad básica:** en este grupo se encontrarán las tareas que permitirán cumplir con la mayoría de los requerimientos funcionales, así como para el desarrollo de la base de datos y la conexión con la aplicación web. Entre estas tareas se encuentran: inserción, modificación, lectura, eliminación de elementos y métodos que realicen cálculos, carga y descarga de datos.
- **Validación de datos:** en este grupo se encuentran las tareas que revisan el área funcional tales como: tipos de datos, estructura, entre otros.
- **Gráficos:** se encontrará las tareas relacionadas con el desarrollo de gráficas.

Este tablero contará con 4 columnas que corresponden a: entrada, implementación, prueba y salida como se muestran a continuación:

Tabla 2.5. Tablero Kanban de funcionalidad básica

Funcionalidad Básica			
Entrada	**Implementación**	**Pruebas**	**Salida**
Creación de la base de datos.	Basándose en los diagramas Entidad – Relación se codificará el script para la creación de la base de datos en SQL Server Management Studio.	N/A	Base de datos

Creación del servicio WCF y conexión con la base de datos mediante Volley.	En Visual Studio se creará una Aplicación de servicios WCF. Uso de Data Access Object(DAO) para la comunicación con la base de datos.	Comprobar la conexión con la base de datos	Sistema prototipo web conectado con la base de datos
Administración de usuario.	Codificar los métodos que permiten realizar las funciones Crear, Leer, Actualizar y Eliminar (CRUD).	Comprobar que los métodos permiten gestionar la información ingresada a la base de datos.	Gestión de usuarios
Ingreso al sistema.	Se codifica los métodos que permiten a un usuario iniciar sesión en la aplicación.	Comprobar que el usuario inició sesión exitosamente.	Entrada al sistema.
Carga de información.	Se codifica los métodos que permiten subir los datos de campeonatos, equipos, jugadores y videos de hinchas.	Comprobar que los métodos creados funcionan correctamente.	Carga de información.
Cálculos del puntaje de equipos.	Se codifica los métodos que permiten realizar los cálculos para determinar la cantidad de puntos que tiene cada equipo.	Comprobar que los métodos creados funcionen correctamente.	Tabla de posiciones.
Creación de partidos en vivo.	Se codifica un método que permita resaltar a los equipos que se encuentren jugando en tiempo real.	Visualizar partidos en vivo.	Partidos en tiempo real.
Creación de noticias.	Se codifican los métodos que permitan mostrar los jugadores que se encuentran sancionados.	Visualizar las noticias generadas.	Noticias.
Creación de pronósticos.	Se codifica el método que permita pronosticar al equipo ganador.	Comprobar que el pronóstico se realizó correctamente.	Pronósticos.
Descarga de información de los equipos.	Se crean los métodos que permitan obtener la información de los	Visualizar información de los equipos.	Información de equipos.

	jugadores que forman parte de un equipo y los últimos resultados de sus partidos.		

En la tabla 2.6 Se presenta el tablero Kanban de validaciones que permite limitar la información según las tareas que éstas realicen.

Tabla 2.6. Tablero Kanban de Validación

Tareas de Validación			
Entrada	**Implementación**	**Pruebas**	**Salida**
Inicio de sesión únicamente para usuarios registrados.	Codificar los métodos que permitan iniciar sesión únicamente a los usuarios registrados.	Comprobar inicio de sesión.	Inicio de sesión para usuarios registrados.
Carga de videos por parte de los hinchas.	Codificar el método que permita subir videos a hinchas previamente registrados.	Comprobar la carga de videos.	Visualización de los videos.
Carga de imágenes.	Codificar los métodos que permitan la carga de imágenes en formato jpg.	Comprobar la carga de imágenes.	Visualización de imágenes.

En la siguiente tabla 2.7 se indica el tablero con sus respectivas tareas:

Tabla 2.7. Tablero Kanban de gráficos

Tareas de Gráficos			
Entrada	**Implementación**	**Pruebas**	**Salida**
Gráficos estadísticos de los equipos	Codificación de los métodos que permitan desarrollar gráficos estadísticos según el rendimiento de los equipos.	Visualizar el gráfico creado.	Gráfico estadístico.
Creación de alineaciones.	Se codifica el método que permita visualizar las alineaciones de los equipos.	Visualizar las alineaciones de los equipos.	Alineación.

2.5. DISEÑO DE LA CAPA DE DATOS

En esta sección se realiza el diseño del diagrama entidad relación de la base de datos que se va a utilizar, el diagrama tendrá sus entidades con cada uno de sus respectivos atributos. Cada una de las entidades se van a relacionar con otras a través del campo que tengan en común. Este diseño será desarrollado para la creación de la base de datos mediante el uso de SSMS de SQL Server 2017.

2.5.1. Diagrama Entidad – Relación

El diagrama entidad – relación como su nombre lo indica permite relacionar entre si entidades dentro de un sistema. Estas entidades pueden ser objetos, elementos, conceptos, con los que se establece un enlace o líneas que conectan las entidades generando una relación [20].

En este diagrama se considera que la liga barrial necesita saber a qué equipo pertenece un jugador, director técnico o hincha para tener registrado en su base de datos, adicionalmente se necesita saber los partidos que va a disputar cada uno de los equipos registrados en un campeonato. Una vez iniciado el campeonato se requiere saber la posición en la que se encuentra cada uno de los equipos, las estadísticas y noticias que se vayan generando en cada uno de los partidos.

Una vez se haya marcado un gol en cualquiera de los partidos que se estén disputando se debe registrar el nombre del jugador que marcó el gol y el equipo del cual forma parte. Se debe tener información de cada uno de los equipos como su escudo y toda la lista de los jugadores que forman parte de dicho equipo. Adicional se debe mostrar información de cualquier jugador escogido por el cliente para que pueda saber al equipo del cual forma parte, edad, nombres, nacionalidad, posición, goles marcados, tarjetas amarillas y tarjetas rojas recibidas.

Por este motivo se ha realizado un análisis estructural[9] basado en la interacción de cada una de las entidades establecidas en la propuesta presentada en reuniones con la administración de la liga en la que se determinó que la entidad Equipos se relaciona con las entidades Partidos, Campeonato, Estadísticas, Noticias, Posiciones, Jugadores, que fueron obtenidas en los requerimientos de usuario. La entidad Hincha fue creada únicamente para la sección de Videos en la cual se permite registrar los datos del hincha que compartirá contenido multimedia con la aplicación. Mediante este diagrama se

[9] Se enfoca en la estructura de un análisis tomando en cuenta los requerimientos, parámetros y medidas que condicionan un resultado.

representará la base de datos que almacenará la información referente a los requerimientos del usuario como se muestra en la figura 2.31.

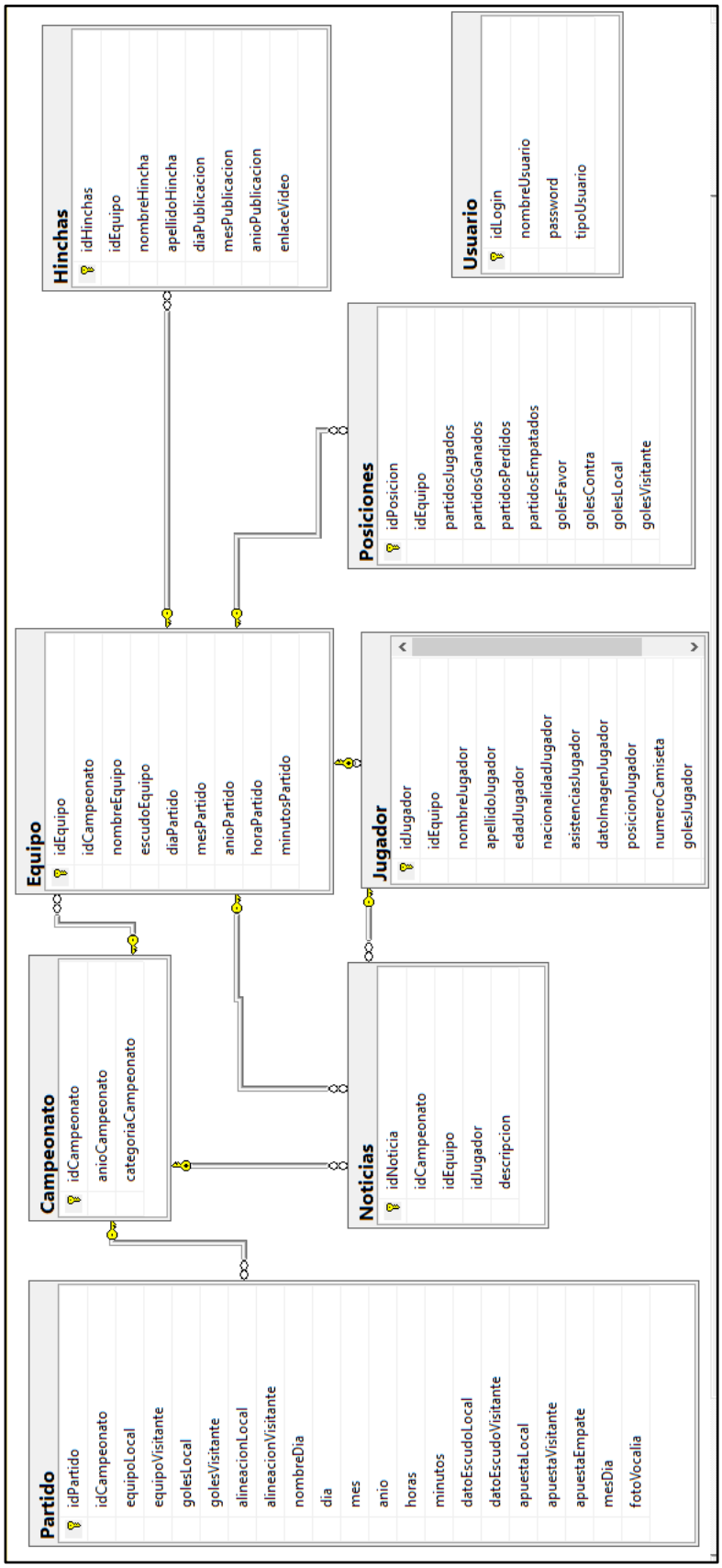


Figura 2.13. Diagrama Entidad - Relación

A continuación, se describe cada una de las entidades establecidas en el diagrama:

- **Tabla Equipo:** almacenará la información de todos los equipos como su nombre y su escudo. Esta tabla tendrá una clave primaria llamada idEquipo y una clave foránea de nombre idCampeonato que la recibe de la tabla Campeonato por su relación de uno a muchos.

- **Tabla Campeonato**: almacenará el nombre y año en el que se está jugando el campeonato. Tendrá una clave primaria llamada idCampeonato.

- **Tabla Jugador**: guardará toda la información de cada uno de los jugadores como nombre, apellido, edad, nacionalidad, asistencias, foto, posición en la que juega, número de camiseta, goles marcados, tarjetas amarillas y rojas. Esta tabla tendrá una clave primaria llamada idJugador y dos claves foráneas idEquipo e idAlineación que son recibidas de las tablas Equipo y Alineación respectivamente.

- **Tabla Partido:** contendrá la información de todos los partidos que se van a disputar dentro de la Liga Deportiva Barrial “La Armenia” como los equipos que hacen de local y visitante, goles marcados y las alineaciones de cada uno de los equipos, fecha y hora del encuentro, los escudos de los equipos y las predicciones del equipo ganador en cada uno de los partidos que han realizado los usuarios. Esta tabla tendrá una clave primaria llamada idPartido.

- **Tabla Posiciones**: guarda la información de los partidos para ubicar a cada equipo según su desempeño, almacenando la cantidad de puntos acumulados, los partidos jugados, ganados, empatados, perdidos, goles a favor, en contra y gol diferencia. Esta tabla tiene una clave primaria llamada idPosicion y una clave foránea que recibe de la tabla Equipo debido a la relación uno a muchos existente entre ellas.

- **Tabla Hinchas**: almacenará los datos del cliente que comparte sus videos con la aplicación como su nombre, apellido y fecha de la publicación. Esta tabla tendrá una clave primaria llamada idHinchas y una clave foránea idEquipo que la recibe de la tabla Equipo debido a la relación de uno a muchos existente entre ellas.

- **Tabla Noticias**: guardará cualquier evento que deba ser comunicado a los miembros de la liga barrial. Esta tabla tendrá una clave primaria llamada idNoticia y dos claves foráneas idEquipo e idJugador que son heredadas de las tablas Equipo y jugador debido a la relación uno a muchos existente entre ellas.

- **Tabla Usuarios:** almacena la información de los dos tipos de usuario: Administrador y Vocal. Esta tabla tendrá una clave primaria llamada idUsuario y un atributo tipoUsuario para diferenciar si el usuario es administrador o vocal.

2.6. DISEÑO DE LA CAPA DE NEGOCIO

En esta capa se desarrolla la parte lógica del prototipo realizando los diagramas UML (Lenguaje Unificado de modelado) que requieren de la combinación de diversos elementos gráficos, con la finalidad de generar una representación simplificada de un sistema que en este caso es el prototipo web [19].

Es importante mencionar que estos diagramas describen como se construirá el prototipo, pero no describe como se implementará.

2.6.1. Diagramas de casos de uso

Una vez establecidos los requerimientos del cliente se establecen los actores que participan en el sistema que en este caso son: el Administrador, Vocal y el Usuario final.
El Administrador crea, edita o elimina la información almacenada en cada uno de los módulos del sistema. El Vocal puede editar la información en tiempo real de los partidos que se están jugando. Este usuario interactúa con la aplicación para ingresar a todos los partidos, visualizar el calendario de programación y resultados de cada uno de ellos. Las siguientes actividades del diagrama de casos de uso fueron obtenidas de los requerimientos de usuario detallados en el alcance del proyecto: navegación de partidos, sección de noticias, tabla de posiciones, estadísticas y videos. Las secciones de información de los equipos y jugadores fueron obtenidas por semejanzas con la aplicación 365 Scores. En la figura 2.13 se muestra el caso de uso del cliente Android.

Figura 2.14. Diagrama de caso de uso Cliente Android

Cada una de estas actividades mencionadas en el diagrama de casos de uso del cliente se describen a continuación:

- Navegar por partidos, permitirá al usuario observar los partidos que se encuentren disponibles, inclusive los que se encuentran disputando en ese momento.
- Sección de noticias, el usuario podrá mirar todos los jugadores que se encuentran suspendidos para la siguiente fecha a disputarse, ya sea por una tarjeta roja o acumulación de tarjetas amarillas.
- Predecir el resultado del partido, el usuario podrá predecir al equipo ganador o si el partido termina en empate, después de su elección se mostrará en porcentaje las votaciones realizadas por otros usuarios.
- Visualizar la alineación de los equipos, el usuario podrá escoger la alineación que desea observar ya sea del equipo local o visitante.
- Información de los equipos, el usuario podrá observar a todos los jugadores que forman parte del equipo y el resultado de sus últimos partidos jugados.
- Observar la información de los jugadores, el cliente podrá ver información del jugador como su edad, goles marcados, asistencias realizadas, número de camiseta, equipos al que pertenece, tarjetas amarillas y rojas.

- Tabla de posiciones, el usuario podrá ver la ubicación en la tabla de todos los equipos del campeonato organizados según su puntaje y gol diferencia.
- Estadísticas, el usuario podrá visualizar el rendimiento de los equipos según el número de puntos conseguidos y los jugadores que han marcado uno o más goles.
- Compartir videos, el usuario deberá poner sus datos para poder compartir sus videos previamente subidos a la plataforma YouTube.

En la figura 2.14 se muestra el diagrama de casos de uso del administrador.

Figura 2.15. Diagrama de caso de uso Administrador

Cada una de estas actividades mencionadas en el diagrama de casos de uso del cliente se describen a continuación:

- Gestionar calendario, permite al administrador realizar las operaciones CRUD para el calendario.
- Gestionar equipos, permite al administrador realizar las operaciones CRUD para el cada uno de los equipos.
- Gestionar jugadores, permite al administrador realizar las operaciones CRUD para todos los jugadores de cada uno de los equipos.
- Gestionar partidos, permite al administrador realizar las operaciones CRUD para el los partidos existentes en un determinado campeonato.

- Gestionar posiciones, permite al administrador realizar las operaciones CRUD que permite tener un control de la ubicación en la tabla de posiciones de cada uno de los equipos.
- Envío de correo electrónico, permite al administrador enviar correo electrónico con la información de la tabla de posiciones y calendario de los partidos a los usuarios de la aplicación.

2.6.2. Diagrama de actividades

El diagrama de actividades que se muestra en la figura 2.32 corresponde al caso de uso que permite registrar la información de un partido.

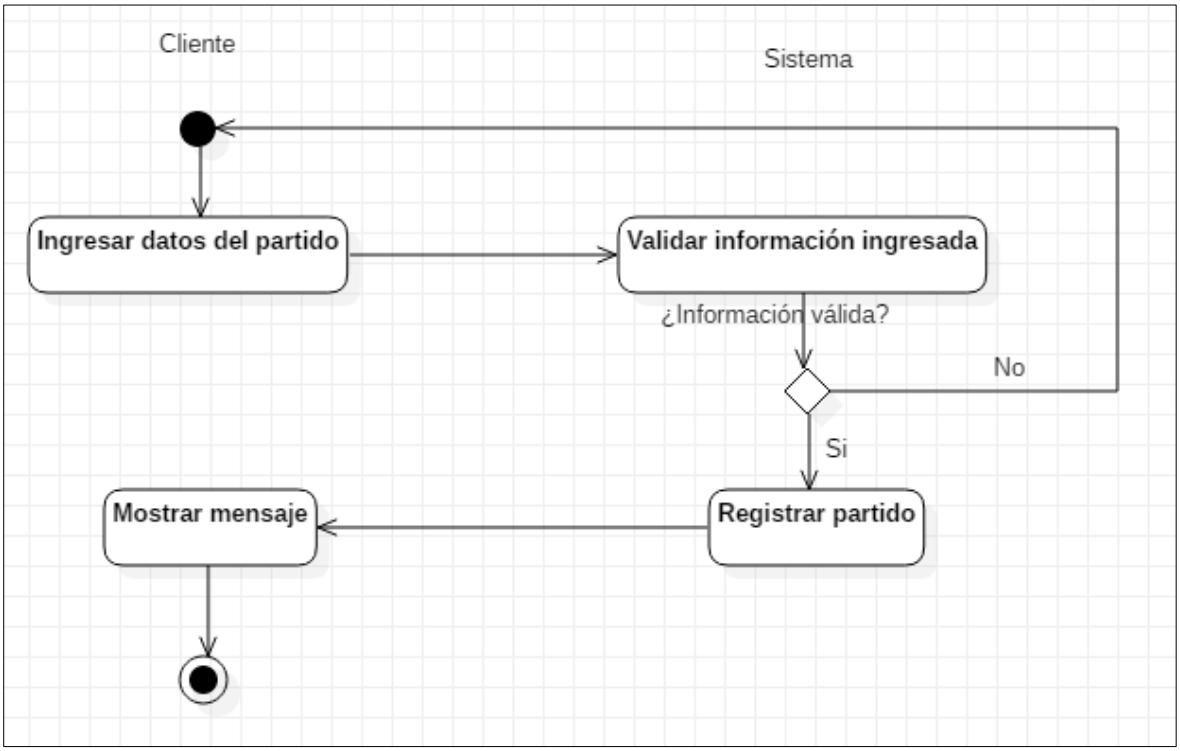


Figura 2.16. Diagrama de actividades registro de un partido

El cliente ingresará los datos y luego se los entrega al servidor para que éste se encargue de validarlos. Si la información ingresada es válida se registra en la base de datos y el cliente recibirá un mensaje para informar el resultado de la transacción. Si la información ingresada no es válida el cliente podrá volver a iniciar el proceso.

En la figura 2.33 se muestra el diagrama de actividades que permite editar la información de un campeonato correspondiente al caso de uso “Editar información de campeonato”.

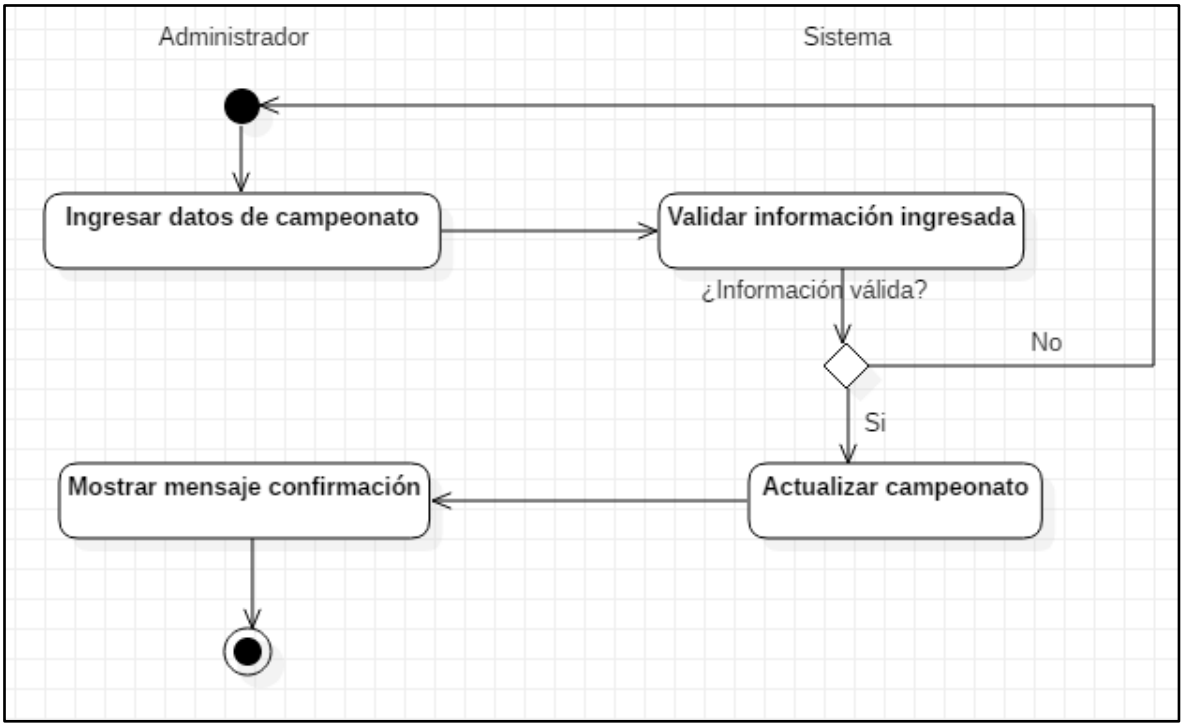


Figura 2.17. Diagrama de actividades actualizar un campeonato

El cliente en la sección del detalle de partidos tendrá la opción de pronosticar al equipo ganador eligiendo a uno de los dos. La votación del cliente se enviará al servidor permitiendo votar una sola vez por partido. Se realiza los cálculos que permitan determinar en porcentaje las votaciones realizadas por los usuarios para posteriormente presentarlos al cliente.

2.6.3. Diagrama de clases

El diagrama de clases permitirá indicar las clases que tiene el sistema con sus respectivos atributos, métodos y la forma en la que se relacionan [21].

En este diagrama las clases Equipo, Partido, Posiciones, Hinchas, Jugador, Estadísticas, Campeonato y Noticias se muestran con sus respectivos atributos como se indica en la figura 2.34.

La cantidad de clases de este diagrama fue determinada por los requerimientos establecidos por el usuario y su diseño en base a un análisis estructural como se mencionó en el diagrama E-R, siendo la clase central Equipo debido a que tiene relación con la mayoría de las clases del sistema.

La clase Partido no tiene relación con ninguna otra debido a que tiene atributos únicos con información recolectada por parte del administrador.

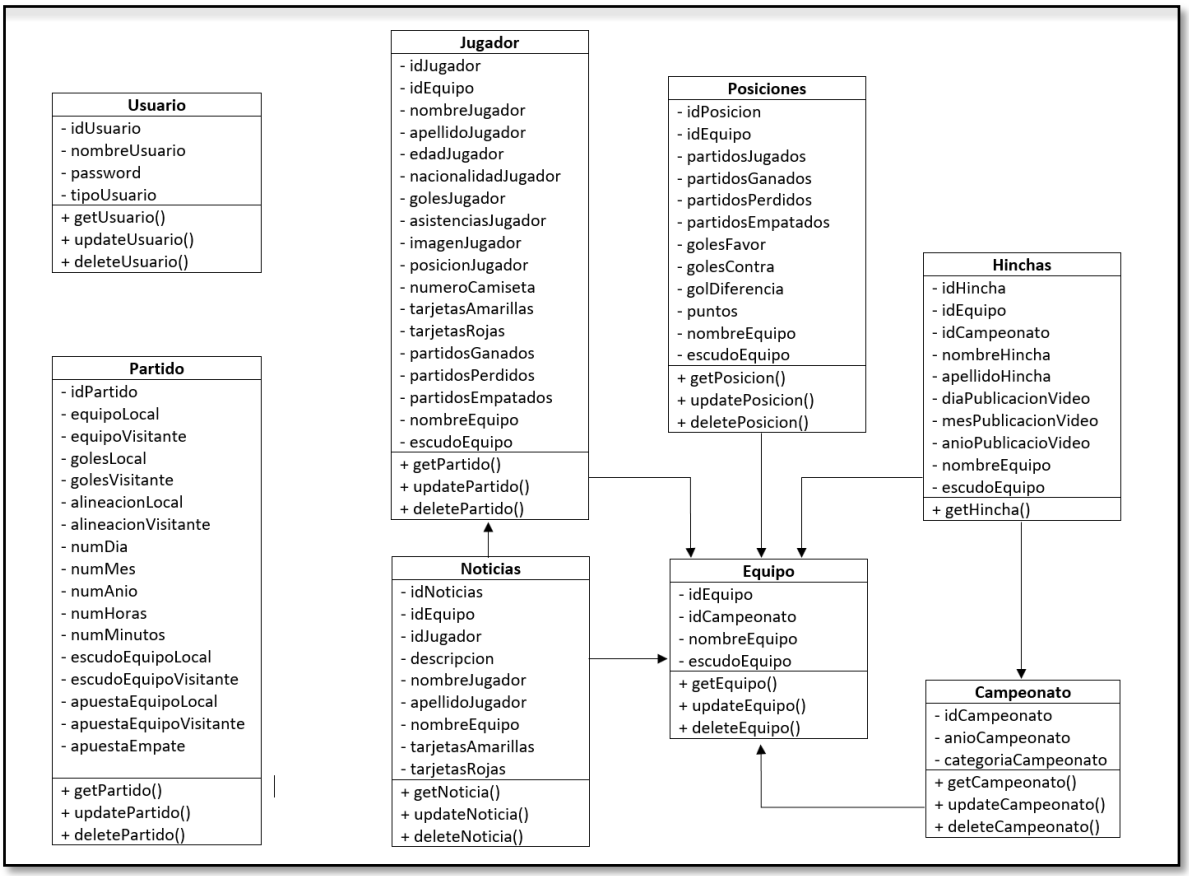


Figura 2.18. Diagrama de clases

El servicio WCF está compuesto por el contrato del servicio IWebService y la clase WebService que implementa a mencionado contrato. El IWebService define las operaciones del servicio mediante el uso de las direcciones URL. El WebService controla las funcionalidades que tiene el sistema.

Los métodos utilizados en el WebService se detallan a continuación:

- **getJugador:** retorna la lista de jugadores que forman parte de un equipo.
- **getUltimosPartidos:** recibe el parámetro nombreEquipo que permite retornar los resultados de los últimos partidos de un determinado equipo.
- **getHincha:** retorna una lista con la información del hincha y del equipo al que está apoyando.
- **getHinchaVideo:** retorna una lista con la información del hincha y con el enlace que permitirá compartir su video.

- **saveHincha:** recibe como parámetros los datos entregados por el hincha para ser registrados, devuelve un mensaje indicando que el registro ha sido exitoso.
- **saveJugador:** recibe como parámetros los datos provenientes de un jugador para ser registrado. Retorna un mensaje indicando el éxito del registro.
- **updateJugador:** recibe como parámetro el identificador de un jugador y los datos que necesiten ser actualizados. Retorna un mensaje informando que la transacción se ha ejecutado correctamente.
- **deleteJugador:** recibe como parámetro el identificador de un jugador proveniente del administrador del sistema para proceder a eliminar al jugador.
- **getEquipo:** retorna una lista con todos los equipos participantes y las posiciones que ocupan en la tabla de resultados.
- **saveEquipo:** recibe como parámetros los datos de un equipo para ser registrado. Retorna un mensaje de confirmación del registro.
- **updateEquipo:** recibe como parámetro el identificador de un equipo y los datos que necesiten ser actualizados. Retorna un mensaje confirmando la transacción.
- **deleteEquipo:** recibe como parámetro el identificador de un equipo que necesite ser eliminado de los registros. Retorna un mensaje informando que el equipo seleccionado ha sido eliminado exitosamente.
- **getDirTecnico:** retorna una lista con todos los directores técnicos de los equipos participantes en la liga.
- **saveDirTécnico:** recibe como parámetro los datos provenientes de un director técnico para ser registrados. Retorna un mensaje informando que los datos han sido registrados exitosamente.
- **updateDirTécnico:** recibe como parámetro el identificador del director técnico y los datos que necesiten ser actualizados. Retorna un mensaje confirmando el éxito de la transacción.
- **deleteDirTécnico:** recibe como parámetros el identificador de un director técnico y los datos que van a ser actualizados. Retorna un mensaje informando que la transacción se realizó con éxito.

- **getCampeonato:** retorna una lista con la información del campeonato que se está disputando en el año en curso.
- **saveCampeonato:** recibe como parámetros los datos proporcionados del campeonato. Retorna un mensaje informando que la información se ha guardado con éxito.
- **UpdateCampeonato:** recibe como parámetros el identificador del campeonato y los campos que necesitan ser actualizados.
- **deleteCampeonato:** recibe como parámetro el identificador del campeonato que permita eliminar el registro seleccionado.
- **getEstadísticas:** retorna una lista con la información recolectada de los equipos a lo largo del campeonato.
- **saveEstadísticas:** recibe como parámetros toda la información recolectada de los equipos según su desempeño. Retorna un mensaje informando que los datos se guardaron correctamente.
- **updateEstadisticas:** recibe como parámetro el identificador y los datos necesarios de actualización. Retorna un mensaje informando que los datos se actualizaron correctamente.
- **deleteEstadísticas:** recibe como parámetro el identificador de una estadística que permitirá eliminarla. Retorna un mensaje informando que el registro se eliminó correctamente.
- **getNoticia:** retorna una lista de todas las noticias pertenecientes a un jugador con su respectivo equipo.
- **saveNoticia:** recibe como parámetro la información que necesite ser publicada con respecto a las novedades que se presenten con los equipos. Retorna un mensaje informando que los datos se han guardado exitosamente.
- **updateNoticia:** recibe como parámetros el identificador de una noticia y los datos de los campos que necesitan ser actualizados. Retorna un mensaje informando que los datos se han actualizado correctamente.
- **deleteNoticia:** recibe como parámetro el identificador de una noticia que necesita ser eliminada. Retorna un mensaje informando el resultado de la transacción.

- **getPartido:** retorna una lista con los datos de los partidos del campeonato.
- **savePartido:** recibe como parámetros los datos de un partido que se desarrolló en el campeonato. Retorna un mensaje informando que los datos se guardaron exitosamente.
- **updatePartido:** recibe como parámetro el identificador del partido y los datos que van a ser actualizados. Retorna un mensaje informando el resultado de la transacción.
- **deletePartido:** recibe como parámetro el identificador del partido a ser eliminado. Retorna un mensaje informando el resultado de la transacción.
- **getPosición:** retorna una lista que permite conocer la posición de cada uno de los equipos que participan en el campeonato.
- **savePosición:** recibe como parámetro los datos de los resultados de los partidos. Retorna un mensaje informando que los datos se guardaron exitosamente.
- **updatePosición:** recibe como parámetros el identificador y los datos que necesitan ser actualizados. Retorna un mensaje informando que los datos fueron actualizados correctamente.
- **deletePosición:** recibe como parámetro el identificador que permite eliminar el registro. Retorna un mensaje informando el resultado de la transacción.

2.7. DISEÑO DE LA CAPA PRESENTACIÓN

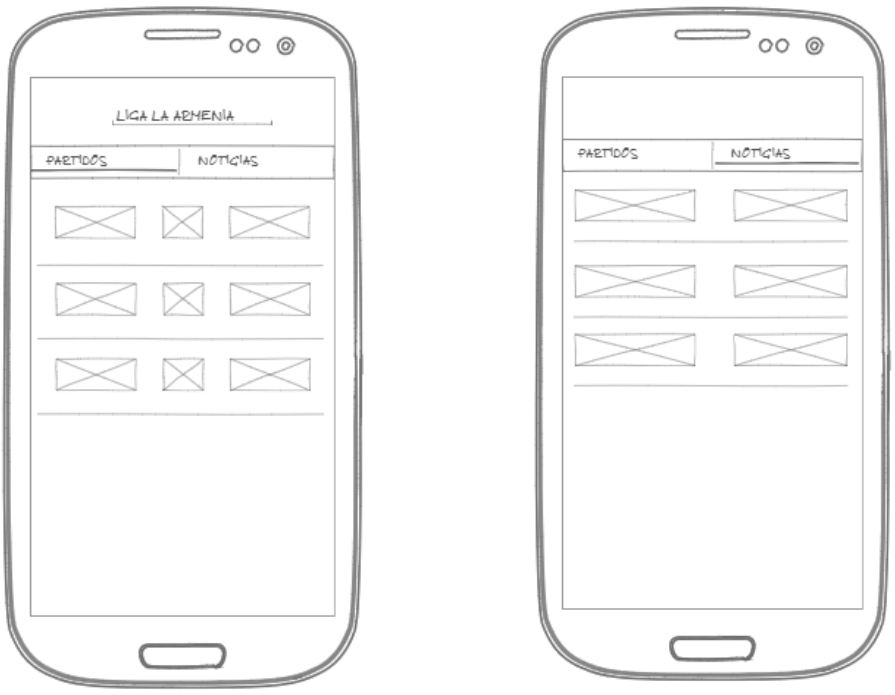
En esta capa se desarrollan las interfaces gráficas que va a tener el sistema, las mismas que se realizó de una forma amigable para su fácil uso del cliente final.

2.7.1. Sketches

Los requerimientos funcionales del usuario fueron tomados para realizar sketches que permitirán realizar las interfaces que va a tener la aplicación.

En la figura 2.35 se muestran los sketches de la pantalla principal del usuario donde se encuentran todos los partidos que están por jugarse en las próximas fechas (a) y la sección de noticias en las que se informa todos los jugadores que están sancionados con tarjetas debido a un mal comportamiento (b). Se creó estas dos pestañas en la parte incial de la

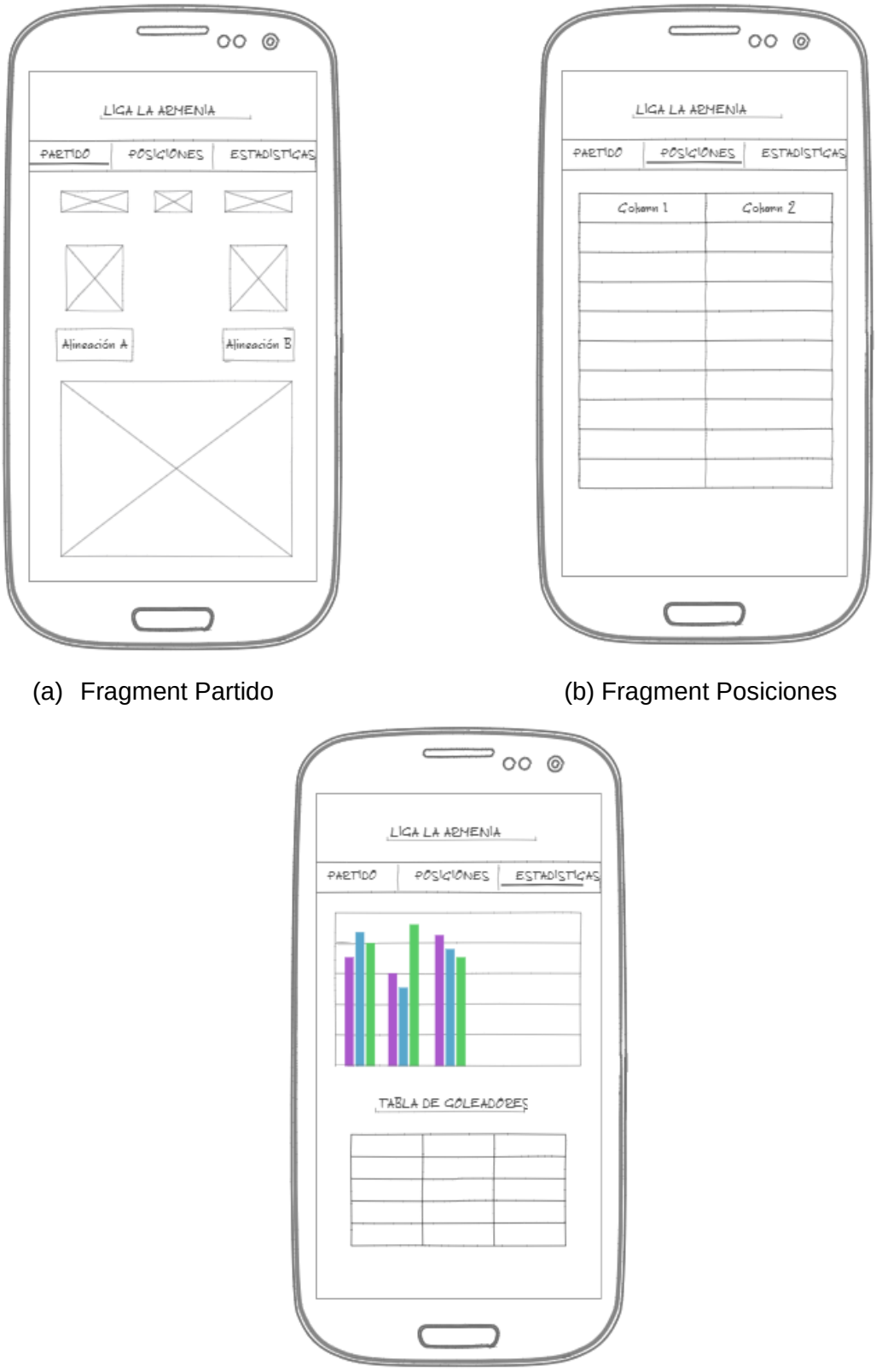
aplicación por ser la información central de la cual se derivan las otras secciones del aplicativo.



(a) Calendario de Partidos (b) Sección Noticias

Figura 2.19. Sketches de la pantalla principal de usuario

En la figura 2.36 se muestran 3 sketches que representan las pestañas que tiene este activity. La primera pestaña tiene la información del partido seleccionado por el usuario en el que se encuentra el tiempo transcurrido, marcador, alineaciones y sección de predicción del partido (a). En la segunda pestaña se muestra la tabla de posiciones de cada uno de los equipos ordenada según la cantidad de puntos que hayan acumulado (b). En la tercera pestaña se indican las estadísticas de los equipos representadas en un gráfico según el rendimiento que hayan tenido a lo largo del campeonato(c). En este activity se colocaron 3 pestañas las mismas que se basaron en la aplicación 365 Scores, obteniendo la información más relevante en el momento de seleccionar un partido como la fecha, hora, pronósticos, alineaciones, estadísticas y tabla de posiciones.



(a) Fragment Partido

(b) Fragment Posiciones

(c) Fragment Estadísticas

Figura 2.20. Sketches de la UI que detallan los partidos

En la figura 2.37 se muestra el sketch donde se detalla el nombre, escudo, los resultados de los últimos 5 partidos y la lista de jugadores que conforman un equipo. En esta sección el componente más adecuado es el uso de un recyclerView que permite utilizar varios componentes que generan información dentro de una misma fila y es independiente de la cantidad de ítems registrados en un arraylist.

Figura 2.21. Sketch que muestra la información de los equipos

En la figura 2.38 se indica la información de cada uno de los jugadores como su nombre, foto, equipo en el que juega, número de camiseta, edad, goles marcados, asistencias realizadas, tarjetas rojas y amarillas. Para mostrar esta información se hace uso de dos imageView que permiten mostrar la foto del jugador y el escudo del equipo al que pertenece, para la información restante se hace uso de textview ya que son campos con poca información.

Figura 2.22. Sketch que muestra la información del jugador

En la figura 2.39 se muestra los videos compartidos por los hinchas de los equipos que conforman la liga barrial. Para mostrar los videos se hace uso de YouTubeAndroidPlayerApi en su versión 1.2.2, el mismo que nos permite reproducir cualquier video que se encuentre en los servidores de la plataforma YouTube.

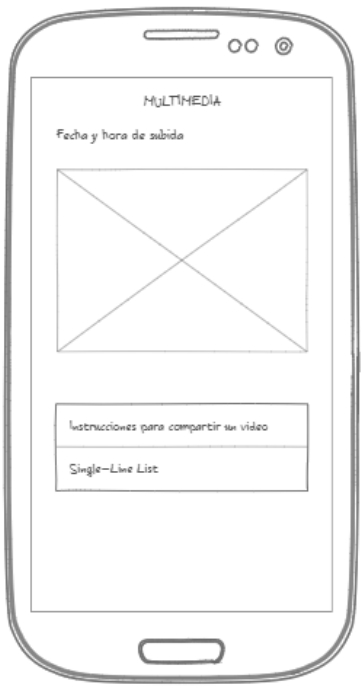


Figura 2.23. Sketch que indica los videos de aficionados

En la figura 2.40 se tiene la información como: el nombre, equipo y el enlace que requiere la plataforma para que el hincha pueda compartir sus videos, estos datos se piden para saber quién sube el video y de que equipo es hincha.

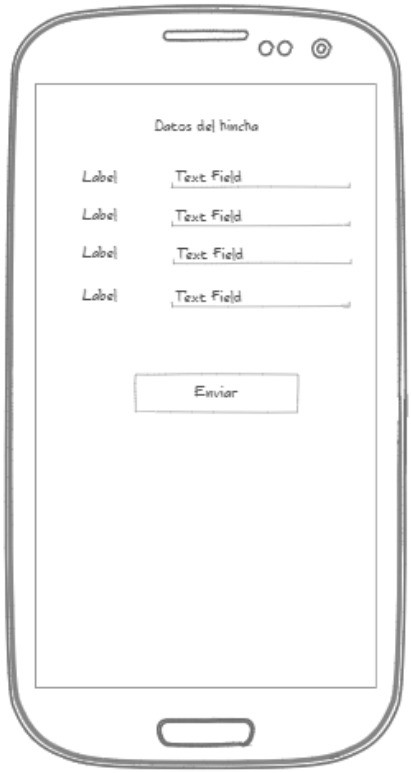


Figura 2.24. Sketch de la información del hincha

2.7.2. Wireframes

En las figuras 2.41 y 2.42 se muestran uno de los principales wireframes que permite entender de mejor manera la transición entre vistas de la aplicación.

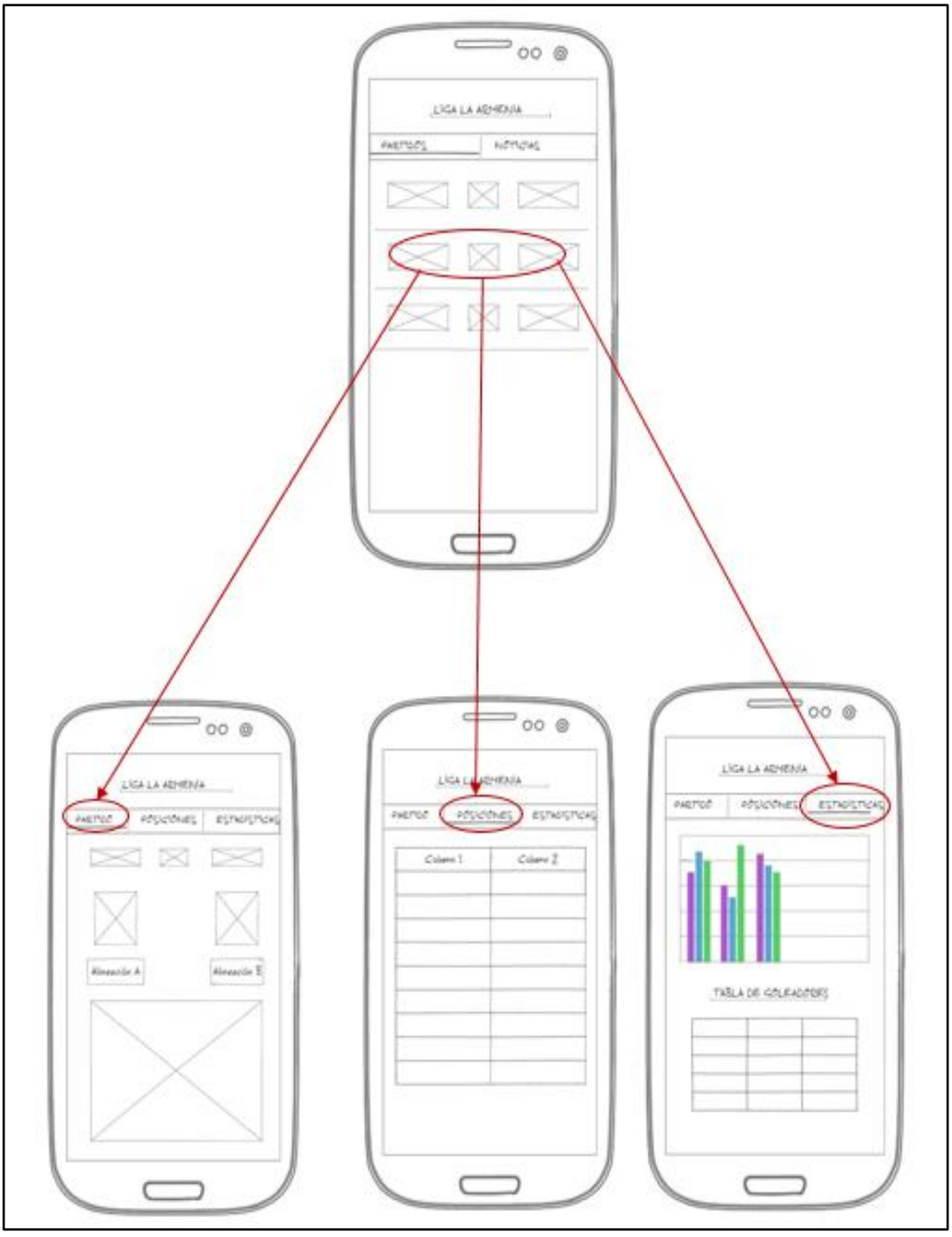


Figura 2.25. Wireframe del prototipo (Parte 1 de 2)

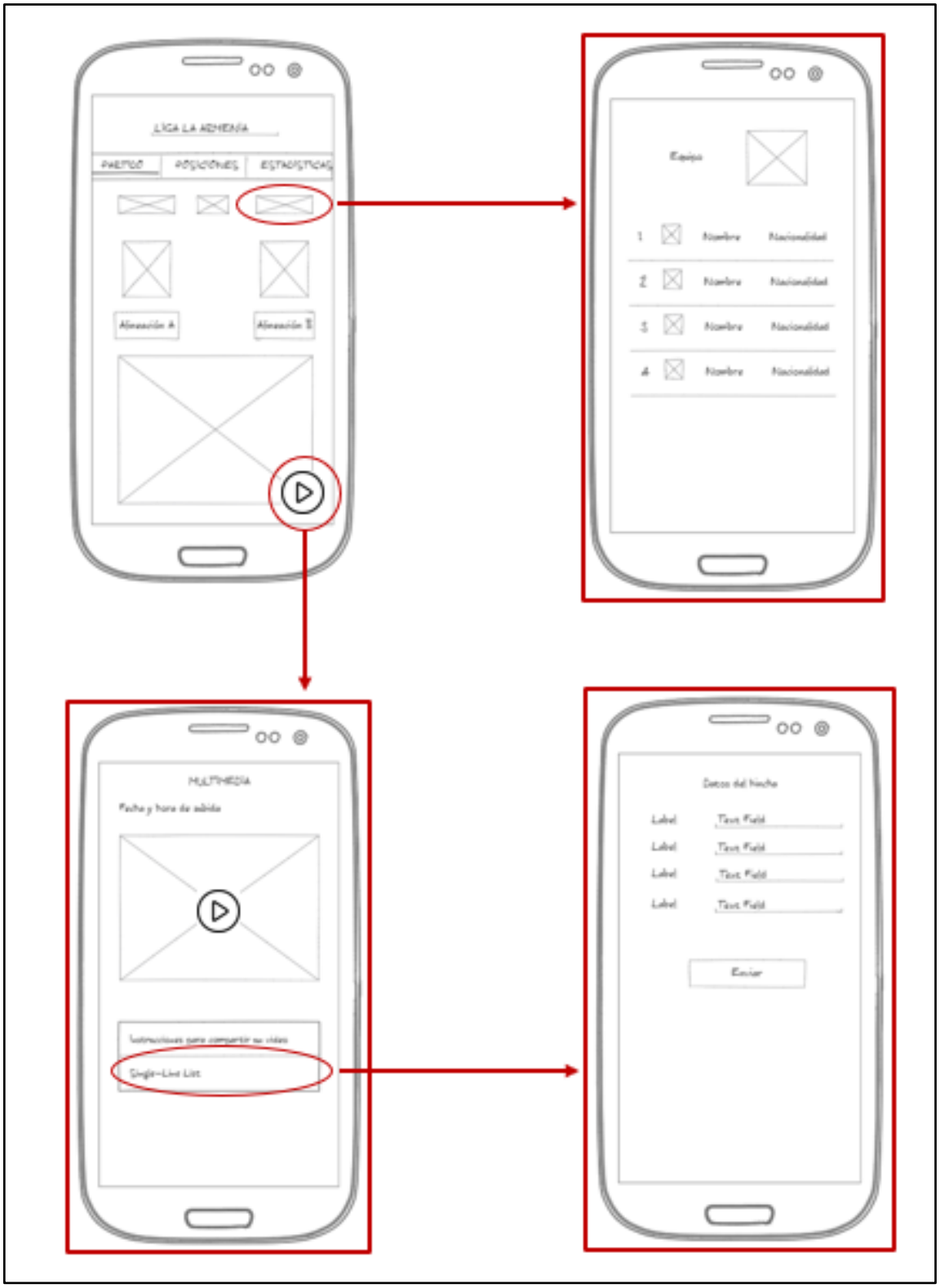


Figura 2.26. Wireframe del prototipo (Parte 2 de 2)

2.8. Implementación de la capa de base de datos

Para el desarrollo del código de la base de datos se utilizó el sistema de gestión de base de datos SSMS de SQL Server 2017. En el código 2.1 en la línea 8 se muestra la creación de la base de datos llamada DB_LigaArmenia en donde se almacenará toda la información del sistema. En las líneas 9 y 10 se establece la base de datos que se va a usar para la ejecución de consultas.

```
8  CREATE DATABASE DB_LigaArmenia
9  USE DB_LigaArmenia
10 GO
```

Código 2.1 Creación de la base de datos DB_LigaArmenia

En el código 2.2 desde la línea 41 a la 60 se muestra la creación de la tabla Partido, las demás tablas utilizadas para la creación de la base de datos se encuentran en el anexo B. En la línea 41 se especifica el nombre de la tabla la misma que tendrá las columnas establecidas entre las líneas 43 a 59. En la línea 41 se establece a la columna idPartido como llave primaria, la cual será auto incremental y no puede ser nula. En las líneas 56 a la 59 se indican las tablas con campos opcionales debido a su tipo NOT NULL.

```
41  CREATE TABLE [Partido]
42  (
43   [idPartido] Int IDENTITY(1,1) NOT FOR REPLICATION NOT NULL PRIMARY KEY,
44   [equipoLocal] Varchar(max) NOT NULL,
45   [equipoVisitante] Varchar(max) NOT NULL,
46   [golesLocal] Int NOT NULL,
47   [golesVisitante] Int NOT NULL,
48   [alineacionLocal] Int NULL,
49   [alineacionVisitante] Int NULL,
50   [nombreDia] Varchar(max) NULL,
51   [dia] Int NOT NULL,
52   [mes] Varchar(max) NOT NULL,
53   [anio] Int NOT NULL,
54   [horas] Int NOT NULL,
55   [minutos] Int NOT NULL,
56   [datoEscudoLocal] Varchar(max) NULL,
57   [datoEscudoVisitante] Varchar(max) NULL,
58   [apuestaLocal] Int NULL,
59   [apuestaVisitante] Int NULL
60  )
```

Código 2.2 Creación de la tabla Partidos

2.9. IMPLEMENTACIÓN DE LA CAPA DE NEGOCIO

En esta sección se describe la implementación de la capa de negocio, que incluye el desarrollo del Servicio WCF, la publicación en IIS (Internet Information Service) y la publicación en los servidores de AWS(Amazon Web Services). Para la publicación en un servidor de internet se escogió a AWS debido a que tiene una mayor cantidad de características sobre los servidores de Microsoft Azure como es la configuración automática en el acceso de los recursos establecidos en un endpoint.

2.9.1. Servicio WCF

El servicio WCF se desarrolló en el Entorno de Desarrollo Integrado (IDE) de Visual Studio y en el lenguaje de programación C# creando una solución y un proyecto llamado WS_LaLiga.

- `WSLaLiga`: este proyecto contendrá al servicio WCF y está compuesto por la interfaz `IWebService` donde se establecen las operaciones del servicio. La clase `WebService` en la cual se definen los métodos que tiene la interfaz. El archivo `Web.config` donde se establece la conexión con la base de datos y se configura los endpoints. En la figura 2.43 se muestra la creación de este proyecto.

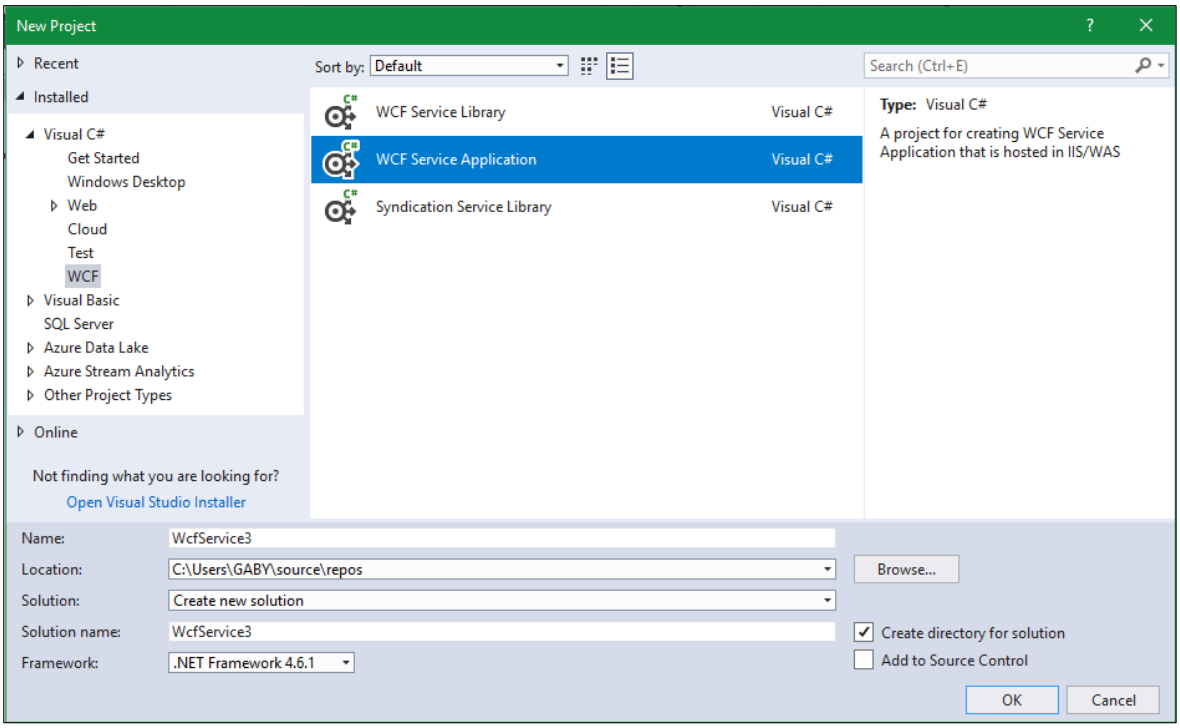


Figura 2.27. Creación del proyecto WS_LaLiga

En el código 2.3 se indica en la línea 11 mediante el uso de [ServiceContract] que la interfaz es el contrato del servicio. En el código 2.4 se muestra la clase que define la interfaz, en la línea 17 se puede observar que la clase WebService se deriva de la interfaz IWebService.

```
 9   namespace WS_Laliga
10   {
11       [ServiceContract]
12       public interface IWebService
```

Código 2.3 Interfaz IWebService

```
15   namespace WS_Laliga
16   {
17       public class WebService : IWebService
```

Código 2.4 Clase WebService

Las clases que se utilizaron en el proyecto son las siguientes: Campeonato, DirTecnico, Equipo, Estadísticas, Hincha, Jugador, Noticias, Partidos y Posiciones.

En el código 2.5 se muestra un ejemplo de los métodos que van a ser utilizados en el servicio REST. El método `getHincha(nombreEquipo)` permite obtener a los hinchas de un determinado equipo y para ello se envía un atributo llamado `nombreEquipo` el mismo que permite saber el equipo del cual el usuario es hincha para traer sus datos. En la línea 21 se usa el atributo `[OperationContract]` que permite al método ser usado por el servicio WCF. En la línea 22 se utiliza el atributo WebGet que nos permite hacer uso del método Get y se identifica de manera única al método haciendo uso de su URI. En la línea 23 se establece el formato JSON que será utilizado para la respuesta del servicio. En la línea 25 se define la firma del método que acepta un objeto del tipo `string` como parámetro de entrada y retorna una lista de objetos del tipo `Hincha`. La firma del método debe coincidir cada uno de sus caracteres con el atributo que va entre llaves dentro de su URI.

```
21          [OperationContract]
22          [WebGet(UriTemplate = "/getHincha/nombreEquipo/{nombreEquipo}",
23              ResponseFormat = WebMessageFormat.Json,
24              BodyStyle = WebMessageBodyStyle.Wrapped)]
25          List<Hincha> getHincha(string nombreEquipo);
```

Código 2.5 Definición del método *`getHincha(nombreEquipo)`*

En el código 2.6 se crea una variable del tipo `string` que nos permite obtener la cadena de conexión establecida en el archivo `Web.config` la misma que nos permite conectarnos con la base de datos.

```
string cadenaConexion = ConfigurationManager.ConnectionStrings["myConnection"].ConnectionString;
```

Código 2.6 Definición de la variable cadenaConexión

En el código 2.7 se muestra el elemento `<connectionStrings>` que establece la cadena de conexión que posee la base de datos para poder acceder a ella. Este método se encuentra en el archivo `Web.config`.

```
<connectionStrings>
  <add name="myConnection" connectionString="Data Source=GABY-PC\SQLEXPRESS;Initial Catalog=DB_LigaArmenia;Integrated
       providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Código 2.7 Elemento <*connectionStrings*>

2.9.1.1. Registro de datos en el servidor WCF

En el código 2.8 se muestra el registro de un hincha cuya función es compartir sus videos con los demás usuarios de la aplicación. Este es un ejemplo de los métodos que permiten registrar datos en el servidor, el resto de código se encuentra en el anexo B. El método `saveHincha()`, lleva como parámetros todos los datos solicitados al hincha como su nombre, apellido y enlace. En la línea 118 se define la consulta que se va a realizar a la base de datos, la misma que lleva todos los parámetros que se definieron en el método. En la línea 122 se realiza una instancia que permite comunicarse con el servidor de base de datos de SQL, lleva como parámetro la cadena de conexión. En la línea 123 se crea una instancia que lleva como parámetros la consulta que se va a realizar y la instancia de conexión con la base de datos. Entre las líneas 126 a 132 se establecen los valores que va a recibir la base de datos en cada una de sus columnas. En la línea 133 y 134 se ejecuta la consulta a la base de datos y se cierra la conexión con la misma. Cabe mencionar que los hinchas son gestionados bajo el perfil de Administrador.

```
112         public int saveHincha(string idEquipo, string nombreHincha, string apellidoHincha, string diaPublicacion,
113             string mesPublicacion, string anioPublicacion, string enlaceVideo)
114         {
115             int res = 0;
116             try
117             {
118                 string query = "insert into Hinchas (idEquipo, nombreHincha, apellidoHincha, diaPublicacion, " +
119                     "mesPublicacion, anioPublicacion , enlaceVideo) values (@idEquipo, @nombreHincha, " +
120                     "@apellidoHincha, " + "@diaPublicacion, @mesPublicacion, @anioPublicacion, @enlaceVideo)";
121 
122                 SqlConnection cnn = new SqlConnection(cadenaConexion);
123                 SqlCommand cmm = new SqlCommand(query, cnn);
124                 cnn.Open();
125                 cmm.CommandType = CommandType.Text;
126                 cmm.Parameters.AddWithValue("idEquipo", idEquipo);
127                 cmm.Parameters.AddWithValue("nombreHincha", nombreHincha);
128                 cmm.Parameters.AddWithValue("apellidoHincha", apellidoHincha);
129                 cmm.Parameters.AddWithValue("diaPublicacion", diaPublicacion);
130                 cmm.Parameters.AddWithValue("mesPublicacion", mesPublicacion);
131                 cmm.Parameters.AddWithValue("anioPublicacion", anioPublicacion);
132                 cmm.Parameters.AddWithValue("enlaceVideo", enlaceVideo);
133                 res = cmm.ExecuteNonQuery();
134                 cnn.Close();
135             }
136             catch (Exception e)
137             {
138                 throw new Exception("Error al guardar hincha", e);
139             }
140             return res;
141         }
```

Código 2.8 Crear un hincha

2.9.1.2. Obtener datos desde el servidor WCF

En el código 2.9 se crea el método `getJugador()`, éste es un ejemplo de los métodos que permiten traer datos desde el servidor el resto de código similar se encuentra en el anexo B adjunto a este documento. En la línea 146 se define una variable del tipo `string` con la consulta que permite obtener datos desde el servidor. Se crea un bucle que permite iterar la información recibida. En la línea 154 se añade a los atributos del objeto Jugador cada una de sus correspondientes columnas que vienen de la tabla Jugador de la base de datos. En la línea 176 se retorna la lista de jugadores que va a ser enviada hacia el cliente Android.

```
144         public List<Jugador> getJugador()
145         {
146             string query = "select * from Posiciones p, Jugador j, Equipo e where p.idEquipo = j.idEquipo
147             SqlConnection cnn = new SqlConnection(cadenaConexion);
148             SqlCommand cmm = new SqlCommand(query, cnn);
149             cnn.Open();
150             SqlDataReader leer = cmm.ExecuteReader();
151 
152             while (leer.Read())
153             {
154                 listaJugador.Add(new Jugador
155                 {
156                     IdJugador = leer.GetInt32(10),
157                     NombreJugador = leer.GetString(13),
158                     ApellidoJugador = leer.GetString(14),
159                     EdadJugador = leer.GetInt32(15),
160                     DatoImagenJugador = leer.GetString(18),
161                     NacionalidadJugador = leer.GetString(16),
162                     PosicionJugador = leer.GetString(19),
163                     NumeroCamiseta = leer.GetInt32(20),
164                     GolesJugador = leer.GetInt32(21),
165                     AsistenciasJugador = leer.GetInt32(17),
166                     TarjetasAmarillas = leer.GetInt32(22),
167                     TarjetasRojas = leer.GetInt32(23),
168                     PartidosGanados = leer.GetInt32(3),
169                     PartidosPerdidos = leer.GetInt32(4),
170                     PartidosEmpatados = leer.GetInt32(5),
171                     NombreEquipo = leer.GetString(26),
172                     EscudoEquipo = leer.GetString(27)
173 
174                 });
175             }
176             return listaJugador;
177         }
```

Código 2.9 Obtención de datos del jugador

2.9.2. Publicación en IIS

El servicio WCF se lo alojará en IIS de Windows 10. Para su publicación se da clic derecho sobre el proyecto `WS_LaLiga` se elige la opción de `Publish` y nos aparece la pantalla de la figura 2.44 Se escoge la opción IIS, FTP, etc. Y se da clic en Publish.

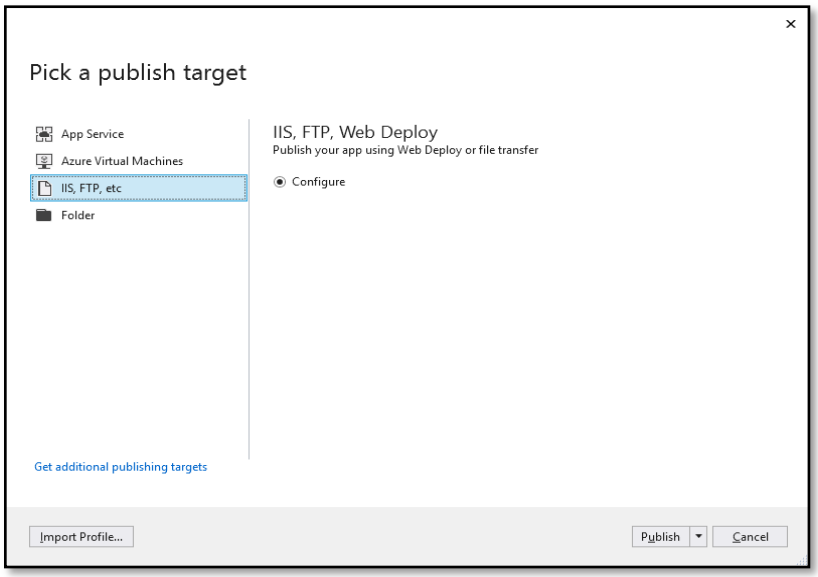


Figura 2.28. Publicar servicio WCF (parte 1)

En la figura 2.45 se debe elegir como método de publicación *FileSystem* y se debe especificar la ruta en la cual va a estar nuestro servidor.

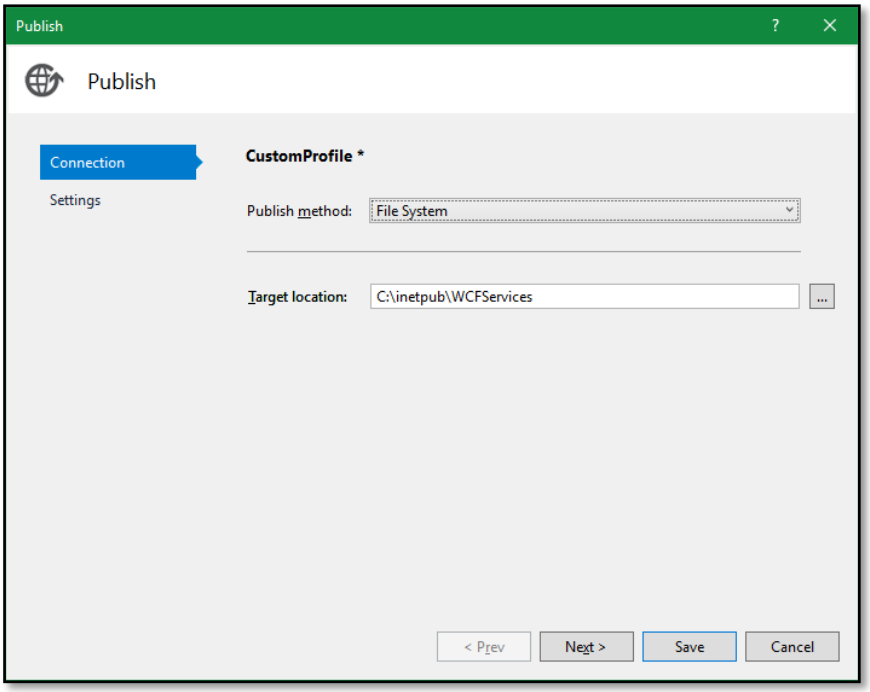


Figura 2.29. Publicar servicio WCF (parte 2)

Finalmente se configura la publicación como Release y se da clic al botón Save como se indica en la figura 2.46.

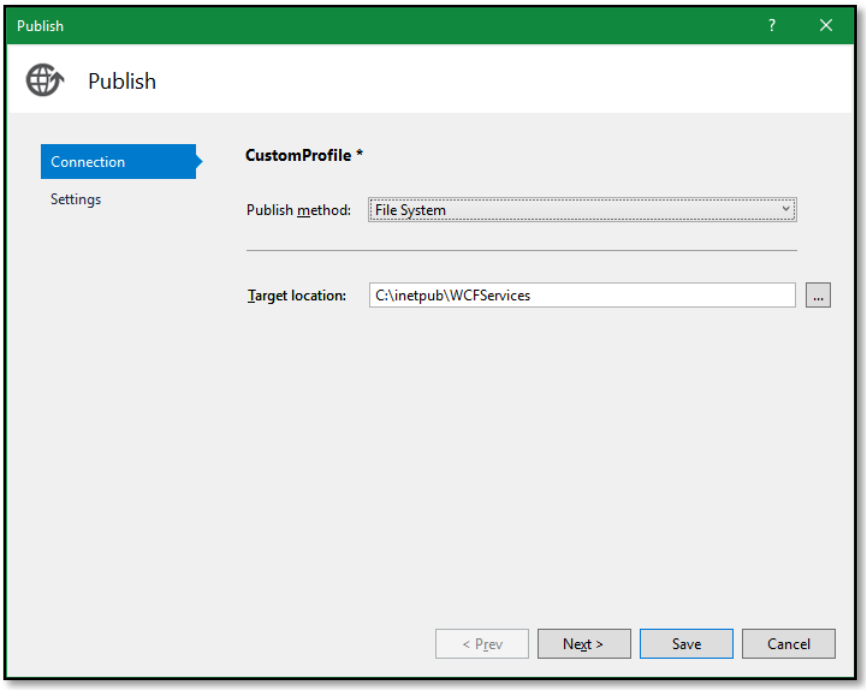


Figura 2.30. Publicar servicio WCF (parte 3)

Para publicar el servicio en IIS se copiaron los archivos publicados en la dirección C:\inetpub como se muestra en la figura 2.47.

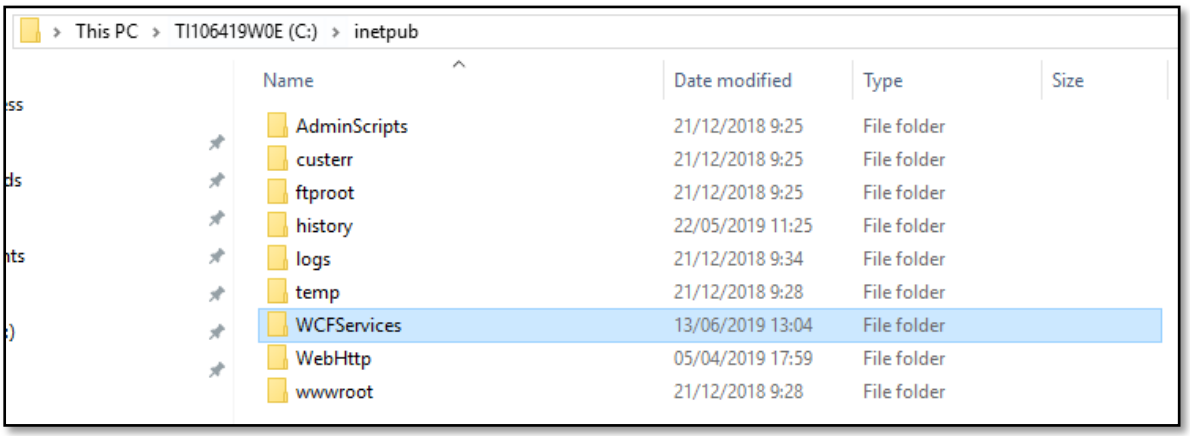


Figura 2.31. Ubicación de los archivos del servicio

En el administrador IIS de Windows 10 se creó un nuevo *WebSite* de nombre *WCFServices,* en la dirección física se estableció `C:\inetpub\WCFServices` establecida en el paso anterior. La dirección IP asignada fue `192.168.100.180` con el puerto `80` habilitado como se muestra en la figura 2.48.

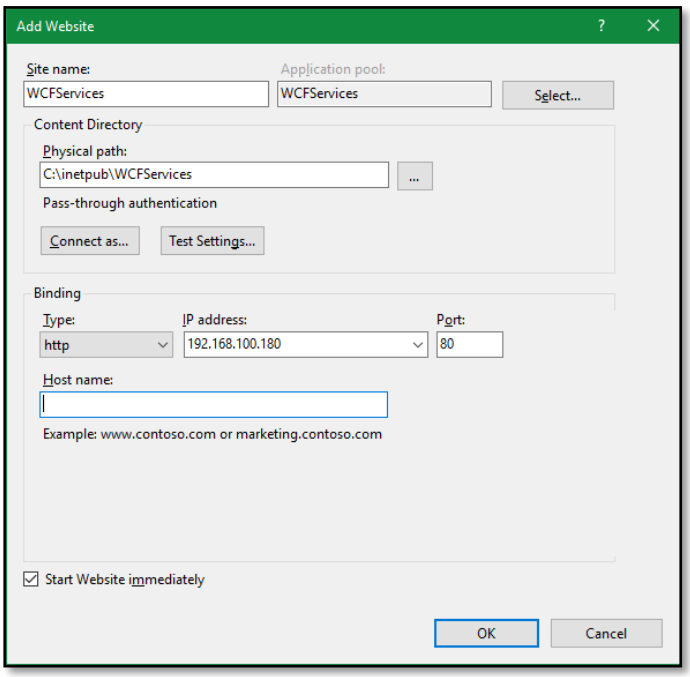


Figura 2.32. Publicar en el servidor IIS

2.9.3. Publicación en AWS

A continuación, se publica el servicio WCF creado en los servidores de Amazon Web Services, para ello se debe instalar el paquete de herramientas de Amazon que permite subir los archivos a sus servidores.

Para publicar el servicio WCF se debe dar clic derecho en el proyecto y luego en *Publish to AWS Elastic Beanstalk*, como se muestra en la figura 2.49.

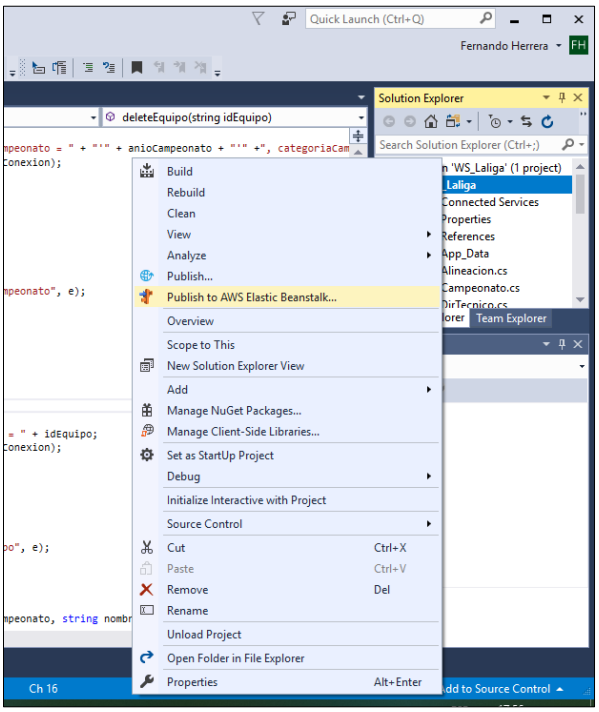


Figura 2.33. Publicar proyecto en AWS

En la figura 2.50 se deben llenar los campos de ID de clave de acceso y Clave de Acceso Secreta que se los obtiene de la página: *http://aws.amazon.com/developers/access-keys/*

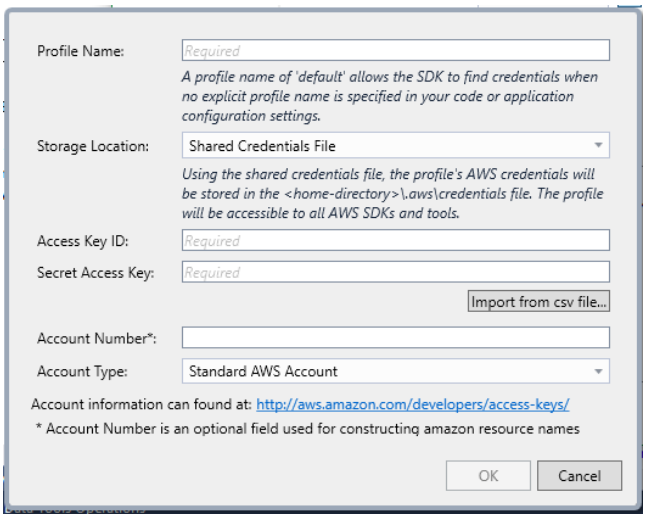


Figura 2.34. Claves de acceso

En la figura 2.51 se ingresa un nombre con el cual se identifica a la aplicación, además se establece una dirección URL con la que se tiene acceso al servicio WCF.

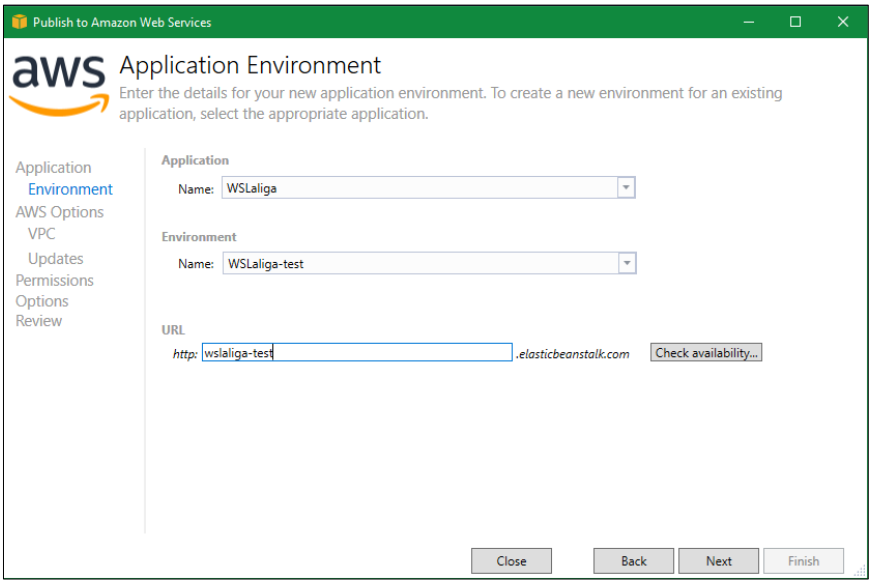


Figura 2.35. Dirección URL

Finalmente se verifican todas las opciones seleccionadas y se da clic en Deploy para que la publicación del servicio comience a subir a los servidores de Amazon, como se muestra en la figura 2.52.

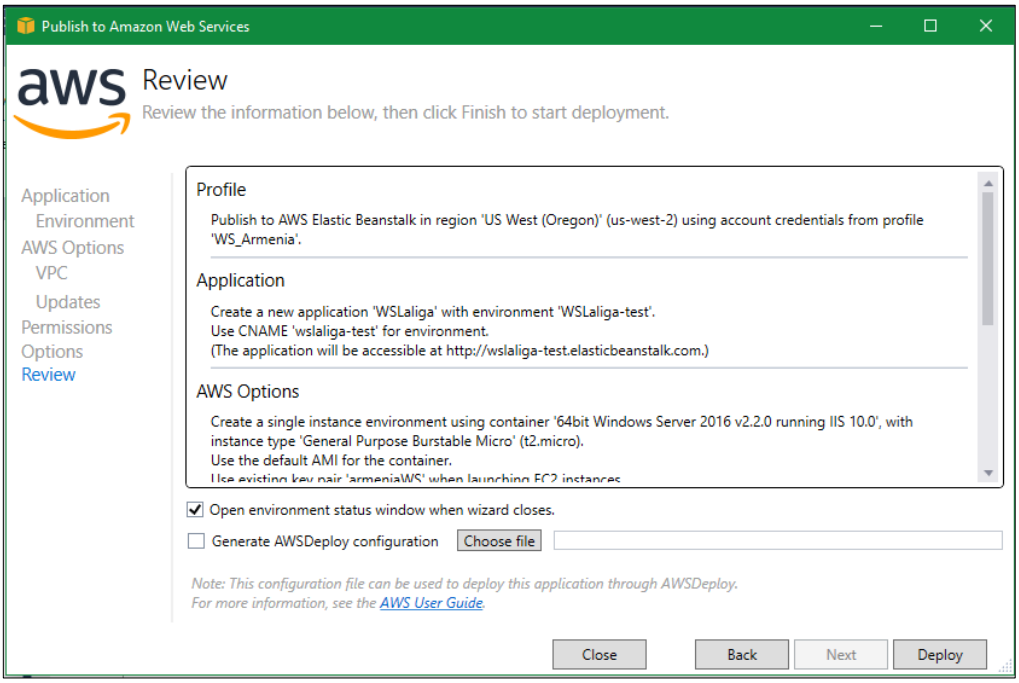


Figura 2.36. Finalizar la publicación

2.10. IMPLEMENTACIÓN DE LA CAPA DE PRESENTACIÓN

La aplicación en Android se desarrolló con el IDE de Android Studio 2.2.1. El lenguaje Java fue utilizado para la lógica del cliente y las interfaces gráficas se desarrollaron con el lenguaje XML. Para la comunicación entre el cliente Android y el servicio Rest se utilizó la librería Volley, para implementarla se necesita escribir en el *gradle*[10] la línea: `compile 'com.android.volley:volley:1.1.1'` permitiendo hacer uso de las clases de la librería.

2.10.1. Cliente Android

En el código 2.10 se realiza un request al servidor que permite obtener los datos de noticias de los equipos. En las líneas 51 y 52 se define la URI que permite identificar el recurso que se desea obtener. En la línea 54 se crea una instancia del tipo JsonObjectRequest definiendo el cuerpo de la solicitud en formato JSON, lleva como parámetros el método Get la URI y el cuerpo del request. En la línea 60 en JsonArray se almacena todos los objetos del tipo JSON que se reciben desde el servidor y lleva como parámetro el nombre del URI establecido en el contrato. Entre las líneas 63 a 73 se realiza un bucle *for* que permite iterar todos los objetos que se reciben desde el servidor y se los almacena en un Arraylist llamado listaNoticias que se encuentra en la línea 73 del código 2.10.

[10] Gradle: Sistema de compilación basado en JVM (Java Virtual Machine) que traduce el código fuente de la aplicación y recursos a código binario y los empaqueta en APK (Android Package). Facilita la actualización y exportación de la aplicación.

En las líneas 75 y 76 el arraylist se envía a una clase llamada AdaptadorNoticias que permite colocar los datos de forma organizada dentro de un RecyclerView[11].

```
50      private void getNoticia() {
51          String ip = getString(R.string.ip);
52          String url = ip +"/WebService.svc/getNoticia";
53
54          JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url, null,
55                  new Response.Listener<JSONObject>() {
56                      @Override
57                      public void onResponse(JSONObject response) {
58                          listaNoticias = new ArrayList<>();
59                          Noticia noticia = null;
60                          JSONArray json = response.optJSONArray("getNoticiaResult");
61
62                          try {
63                              for (int i = 0; i < json.length(); i++){
64                                  noticia = new Noticia();
65                                  JSONObject jsonObject = null;
66                                  jsonObject = json.getJSONObject(i);
67                                  noticia.setNombreEquipo(jsonObject.optString("nombreEquipo"));
68                                  noticia.setNombreJugador(jsonObject.optString("nombreJugador"));
69                                  noticia.setApellidoJugador(jsonObject.optString("apellidoJugador"));
70                                  noticia.setDescripcion(jsonObject.optString("descripion"));
71                                  noticia.setTarjetasAmarillas(jsonObject.optInt("tarjetasAmarillas"));
72                                  noticia.setTarjetasRojas(jsonObject.optInt("tarjetasRojas"));
73                                  noticia.setDatoImagenEquipo(jsonObject.optString("imagenEquipo"));
74                                  listaNoticias.add(noticia);
75                                  Toast.makeText(getContext(), "Noticia leida", Toast.LENGTH_SHORT).show();
76
77                                  AdaptadorNoticias adapter = new AdaptadorNoticias(listaNoticias);
78                                  recyclerNoticias.setAdapter(adapter);
79                              }
80                          } catch (JSONException e) {
81                              e.printStackTrace();
82                          }
```

Código 2.10 Recibir datos desde el servidor WCF con Volley (parte 1)

En la línea 85 mediante el uso de Toast se informa al cliente la existencia de un error en el momento de recibir los datos. En la línea 89 se utiliza RequesQueue[12] que utiliza la red para realizar el transporte de los datos a la cual se le agrega la petición con los objetos en formato JSON realizada previamente.

```
82                      }, new Response.ErrorListener() {
83                  @Override
84                  public void onErrorResponse(VolleyError error) {
85                      Toast.makeText(getContext(), "Error al leer la noticia", Toast.LENGTH_SHORT).show();
86                  }
87              });
88
89              request = Volley.newRequestQueue(getContext());
90              request.add(jsonObjectRequest);
```

Código 2.11 Recibir datos desde el servidor WCF con Volley (parte 2)

[11] RecyclerView: conjunto de componentes que permite renderizar los ítems de una lista para mostrar sus datos de una forma eficiente.

[12] RequestQueue: Clase que permite transportar las solicitudes hacia el servidor.

Para hacer uso del componente RecyclerView se debe agregar en el gradle la siguiente línea de código `compile 'com.android.support:recyclerview-v7:26.0.0-alpha1'` que permite instalar la librería para poder hacer uso de sus clases.

En el código 2.12 se crea la clase AdaptadorNoticias que permite enlazar los datos que se recibe desde el servidor para mostrarlos dentro de un RecyclerView en el cliente. En la línea 24 se crea un constructor que nos permitirá hacer uso de la lista que viene desde el servidor. Dentro de la clase *DetallePartidosViewHolder* se define los atributos que permiten relacionar con los componentes de la interfaz gráfica que conforma el RecyclerView.

```
24      ArrayList<Noticia> listaNoticias;
25      public AdaptadorNoticias(ArrayList<Noticia> listaNoticias) {
26          this.listaNoticias = listaNoticias;
27      }
28
29      public class DetallePartidosViewHolder extends RecyclerView.ViewHolder {
30          TextView txtNombreEquipo, txtNombreJugador, txtApellidoJugador, txtTarjetasRojas, txtTarjetasAmarillas, txtDescripcion;
31
32          public DetallePartidosViewHolder(View itemView) {
33              super(itemView);
34
35              txtNombreEquipo = itemView.findViewById(R.id.txtNombreEquipo);
36              txtNombreJugador = itemView.findViewById(R.id.txtJugador);
37              txtTarjetasRojas = itemView.findViewById(R.id.txtTarjetaRoja);
38              txtTarjetasAmarillas = itemView.findViewById(R.id.txtTarjetasAmarillas);
39              txtDescripcion = itemView.findViewById(R.id.txtDescripcion);
40          }
41      }
```

Código 2.12 Adaptador noticias (parte 1)

En el código 2.13 línea 45 se indica el layout que tendrá los componentes que van a ser utilizados dentro del RecyclerView. Entre las líneas 53 y 57 se agregan a los componentes los datos provenientes de la lista de noticias. En la línea 62 se indica la cantidad de elementos que van a ser mostrados en los ítems del RecyclerView.

```
43      @Override
44      public DetallePartidosViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
45          View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_list_noticias, null, false);
46          view.setOnClickListener(this);
47          return new DetallePartidosViewHolder(view);
48      }
49
50      @Override
51      public void onBindViewHolder(DetallePartidosViewHolder holder, int position) {
52
53          holder.txtNombreEquipo.setText(listaNoticias.get(position).getNombreEquipo());
54          holder.txtNombreJugador.setText("Jugador: " + listaNoticias.get(position).getNombreJugador() + " " + listaNoticias.get(position)
55          holder.txtTarjetasAmarillas.setText("T. Amarillas: " + String.valueOf(listaNoticias.get(position).getTarjetasAmarillas()));
56          holder.txtTarjetasRojas.setText("T. Roja: " +String.valueOf(listaNoticias.get(position).getTarjetasRojas()));
57          holder.txtDescripcion.setText(listaNoticias.get(position).getDescripcion());
58      }
59
60      @Override
61      public int getItemCount() {
62          return listaNoticias.size();
63      }
```

Código 2.13 Adaptador noticias (parte 2)

Para el desarrollo de las interfaces gráficas se hace uso de la herramienta *ConstraintLayout* la misma que se la instala escribiendo en el *gradle* dentro de *dependencies* la siguiente línea que se muestra en el código 2.14.

```
compile 'com.android.support.constraint:constraint-layout:1.1.3'.
```

Código 2.14 Dependencia Constraint Layout

En el código 2.15 se muestra el *layout* correspondiente al *fragment PartidosFragment.* En las líneas 3 a la 6 se definen los *namespaces* que serán utilizados. En las líneas 7 a la 9 se establecen los límites del componente. En la línea 10 se establece el color de fondo que va a tener el *fragment*. En las líneas 12 a la 26 se muestra un ejemplo de los componentes que se va a utilizar dentro de este *fragment*. Entre estas líneas se define un componente con la etiqueta `<TextView>` el cual nos muestra la información que le asignemos. En la línea 13 se le asigna un identificador que permitirá hacer uso del componente dentro de las clases Java. Entre las líneas 15 a la 24 se define el tamaño y los límites que va a tener el componente en relación con los otros elementos. En la línea 25 y 26 se establece el tamaño y el color que va a tener la fuente respectivamente.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:orientation="vertical"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9      android:layout_centerInParent="true"
10     android:background="@color/cardview_dark_background">
11
12     <TextView
13         android:id="@+id/txtFecha"
14         android:text="TextView"
15         android:layout_width="wrap_content"
16         android:layout_height="wrap_content"
17         android:layout_marginEnd="16dp"
18         app:layout_constraintRight_toRightOf="parent"
19         android:layout_marginRight="16dp"
20         android:layout_marginStart="16dp"
21         app:layout_constraintLeft_toLeftOf="parent"
22         android:layout_marginLeft="16dp"
23         android:layout_marginTop="8dp"
24         app:layout_constraintTop_toTopOf="parent"
25         android:textSize="10dp"
26         android:textColor="@android:color/white"/>
```

Código 2.15. Layout del fragment *PartidosFragment*

En el código 2.16 se define el método `horaActual()` permitiendo obtener la fecha y hora actual. En la línea 95 se crea una instancia de la clase *Time* obteniendo la hora según la zona horaria que se encuentre. En las líneas 97 a 102 se usa las variables `día, mes, anio, horaActual y minutos` igualando a los tiempos obtenidos con la clase *Time*.

```
94      private void horaActual(){
95          Time today = new Time(Time.getCurrentTimezone());
96          today.setToNow();
97          dia = today.monthDay;
98          mes = today.month;
99          mes++;
100         anio = today.year;
101         horaActual = today.hour;
102         minutos = today.minute;
```

Código 2.16. Método *horaActual()*

En el código 2.17 se define el método `iniciarCronometro()` permitiendo mostrar al cliente el tiempo transcurrido del partido seleccionado. En la línea 115 se crea la variable tiempoTranscurrido que se define de la resta entre los minutos actuales y los minutos en los que empieza el partido. Entre las líneas 118 a la 123 se definen variables que establecen los límites para determinar si el partido se encuentra en el primer tiempo, descanso o en el segundo tiempo. En la línea 125 se realiza el condicional que permitirá analizar si la fecha y hora actual coinciden con la fecha del partido. En la línea 129 se establece un condicional que permitirá identificar si el partido se encuentra en el primer tiempo, en el descanso o si el partido ya ha terminado. En la línea 134 se hace uso del método *postDelayed()* que permite ejecutar una acción después de un tiempo determinado.

```
113     private void iniciarCronometro(){
114
115         int tiempoTranscurrido = minutos - partidos.getMinutos();
116         tiempoTranscurrido = tiempoTranscurrido*60000;
117
118         int diaPartido = partidos.getDia();
119         int horaPartido = partidos.getHoras();
120         int horaMaxima = horaPartido+2;
121         int diaMes = partidos.getMesDia();
122         int anioPartido = partidos.getAnio();
123         int partidoIniciado = horaActual - partidos.getHoras();
124
125         if (tiempoTranscurrido < 0 && partidoIniciado == 1 && diaPartido == dia && diaMes == mes && anioPartido == anio){
126             int tiempoJugado = 3600000 + tiempoTranscurrido;
127             int primerTiempo = 2700000 - tiempoJugado;
128
129             if (primerTiempo < 0 && primerTiempo > -900000){
130                 int tiempoDescanso = 900000 + primerTiempo;
131
132                 cronometro.setText("MT");
133                 txtHora.setText(partidos.getGolesLocal() + " - " + partidos.getGolesVisitante());
134                 cronometro.postDelayed(new Runnable() {
135                     @Override
136                     public void run() {
137                         segundoTiempo();
138                         cronometro.postDelayed(() -> {
141                                 cronometro.stop();
142                                 cronometro.setText("PT");
143                                 txtHora.setText(partidos.getGolesLocal() + " - " + partidos.getGolesVisitante());
144                         },2700000);
146                     }
```

Código 2.17. Método `iniciarCronometro()`

En el código 2.18 se muestra los métodos `pausarCronometro()` y `reiniciarCronometro()`. En la línea 252 se hace uso del método `stop()` para detener el cronómetro y en la línea 253 se captura el tiempo en el que fue detenido el cronómetro.

En la línea 261 se captura el tiempo transcurrido en el cronómetro y se le asigna el valor de 0 a la variable `detenerse` para reiniciar el cronómetro.

```
250     private void pausarCronometro(){
251         if (correr){
252             cronometro.stop();
253             detenerse = SystemClock.elapsedRealtime() - cronometro.getBase();
254             String tiempoCronometro = cronometro.getText().toString();
255             Toast.makeText(getContext(), "Tiempo: " + tiempoCronometro, Toast.LENGTH_SHORT).show();
256             correr = false;
257         }
258     }
259     private void reiniciarCronometro(){
260         cronometro.setBase(SystemClock.elapsedRealtime());
261         detenerse = 0;
262     }
```

Código 2.18. Métodos *pausarCronometro() y reiniciarCronometro()*

En el Código 2.19 se crea el método `convertidorImagen()` que permite mostrar las imágenes dentro del *fragment*. En la línea 282 se almacena la imagen en un arreglo de bytes recibiendo los datos en formato base64. En la línea 283 se convierte la imagen en formato Bitmap para poder ser mostrado en los ImageView.

```
281     private void convertidorImagen(){
282         byte[] byteCode = Base64.decode(datoEscudoLocal, Base64.DEFAULT);
283         IMG_EQUIPOLOCAL = BitmapFactory.decodeByteArray(byteCode, 0,byteCode.length);
284
285         byte[] byteCodeVisitante = Base64.decode(datoEscudoVisitante, Base64.DEFAULT);
286         IMG_EQUIPOVISITANTE = BitmapFactory.decodeByteArray(byteCodeVisitante, 0,byteCodeVisitante.length);
287     }
```

Código 2.19. Método *convertidorImagen()*

En el código 2.20 se crea el método `sumaContador()` que administra la cantidad de votaciones realizadas por los clientes en la sección de pronósticos. En la línea 290 se almacena el identificador del partido seleccionado por el cliente. En la línea 291 se llama al método `enviarApuestas()` que permite enviar las apuesta al servidor. Entre las líneas 293 y 303 se realizan los cálculos del porcentaje según las votaciones de los clientes y se muestran los resultados en los componentes respectivos.

```
289     private void sumaContador(int apuestaLocal, int apuestaVisitante, int apuestaEmpate){
290         idPartido = partidos.getIdPartido();
291         enviarApuestas(idPartido, apuestaLocal, apuestaVisitante, apuestaEmpate);
292
293         suma = apuestaLocal + apuestaVisitante + apuestaEmpate;
294         porcentajeLocal = (100 * apuestaLocal)/suma;
295         porcentajeVisitante = (100 * apuestaVisitante)/suma;
296         porcentajeEmpate = 100 - porcentajeLocal - porcentajeVisitante;
297
298         txtApuestaLocal.setText(String.valueOf(porcentajeLocal)+"%");
299         txtApuestaLocal.setVisibility(View.VISIBLE);
300         txtApuestaVisitante.setText(String.valueOf(porcentajeVisitante)+"%");
301         txtApuestaVisitante.setVisibility(View.VISIBLE);
302         txtPorcentajeEmpate.setText(String.valueOf(porcentajeEmpate)+"%");
303         txtPorcentajeEmpate.setVisibility(View.VISIBLE);
304     }
```

Código 2.20. Método `sumaContador()`

En el código 2.21 se pasan datos entre formularios y se muestran las imágenes de un jugador que forma parte de un equipo. En la línea 434 se crea una instancia de la clase *Intent* que indica el *activity* actual y el *activity* al cual se envían los datos. En la línea 435 se crea una instancia de la clase *Bundle* que permite enviar los datos, en este caso se envía el objeto `partidos`. En la línea 439 se inicia el nuevo *activity* con los datos a enviar. En la línea 444 se agrega la imagen del jugador a su respectivo *ImageView*. Entre las líneas 449 y 453 se muestra el código que permite pasar al *activity* en donde se muestra la información del jugador seleccionado.

```
431     imgVEquipoVisitante.setOnClickListener((view) → {
434             Intent intent = new Intent(getActivity(), DetalleEquipoActivity.class);
435             Bundle bundleEnvio = new Bundle();
436             bundleEnvio.putSerializable("EQUIPO", partidos);
437             bundleEnvio.putSerializable("BOTON", "visitante");
438             intent.putExtras(bundleEnvio);
439             startActivity(intent);
440     });
442
443
444     imgVJugador1.setImageBitmap(jugador1.getImagenJugador());
445     imgVJugador1.setOnClickListener((view) → {
448             Toast.makeText(getContext(), "Tu has seleccionado a: " + jugador1.getNombreJugador(), Toast.LENGTH_SHORT).show();
449             Intent intent = new Intent(getActivity(), DetalleJugadorActivity.class);
450             Bundle bundleEnvio = new Bundle();
451             bundleEnvio.putSerializable("JUGADOR", jugador1);
452             intent.putExtras(bundleEnvio);
453             startActivity(intent);
454
455     });
```

Código 2.21. Uso de ImageView

En el código 2.22 se muestra el método `alineacionEquipo()` que lleva como parámetros la alineación y el identificador del equipo donde se ubica a los jugadores en la posición establecida por su director técnico. En la línea 713 se usa el condicional *if* para identificar si la alineación es del equipo local o visitante. En la línea 714 se establece la alineación que va a presentar el equipo. En las líneas 715 a la 736 se posiciona a los jugadores en la posición que les corresponde.

```
711     private void alineacionEquipoLocal(int alineacion, int id) {
712
713         if (id == idBtnAlineacionLocal){
714             if (alineacion == 352) {
715                 imgVJugador1.setX(480f);
716                 imgVJugador1.setY(920f);
717                 imgVJugador2.setX(320f);
718                 imgVJugador2.setY(720f);
719                 imgVJugador3.setX(480f);
720                 imgVJugador3.setY(740f);
721                 imgVJugador4.setX(640f);
722                 imgVJugador4.setY(700f);
723                 imgVJugador5.setX(240f);
724                 imgVJugador5.setY(460f);
725                 imgVJugador6.setX(360f);
726                 imgVJugador6.setY(500f);
727                 imgVJugador7.setX(480f);
728                 imgVJugador7.setY(520f);
729                 imgVJugador8.setX(600f);
730                 imgVJugador8.setY(500f);
731                 imgVJugador9.setX(720f);
732                 imgVJugador9.setY(460f);
733                 imgVJugador10.setX(400f);
734                 imgVJugador10.setY(150f);
735                 imgVJugador11.setX(560f);
736                 imgVJugador11.setY(150f);
737             }
```

Código 2.22.Método *alineacionEquipos*

Para el módulo Posiciones se crea el código 2.23 dentro de la clase Posiciones que permite comparar los puntos que tienen cada uno de los equipos como se muestra en la línea 139.

```
137     @Override
138     public int compareTo(Posiciones posiciones) {
139         return puntos.compareTo(posiciones.getPuntos());
140     }
```

Código 2.23. Comparar puntos tabla de posiciones

En el código 2.24 se ordenan los puntajes de los equipos de mayor a menor mediante el uso de la clase *Collections.sort* como se muestra en la línea 127. En la línea 128 se envía las posiciones al *Adaptador_Posiciones* para ser mostrados en un *RecyclerView*.

```
127     Collections.sort(listaPosiciones, Collections.<Posiciones>reverseOrder());
128     Adaptador_Posiciones adapter = new Adaptador_Posiciones(listaPosiciones);
129     recyclerPosiciones.setAdapter(adapter);
```

Código 2.24. Ordenar puntajes de equipos

Para el módulo de Noticias se trae la información desde el servidor de los jugadores que se encuentran con tarjetas amarillas o rojas, el código utilizado es muy similar al código 2.10 mostrado anteriormente.

En el módulo Estadísticas se analiza el rendimiento de los jugadores según la cantidad de puntos conseguidos. Los resultados se muestran en un gráfico de barras que representa en el eje X a los equipos y en el eje Y la cantidad de puntos conseguidos. Para el uso de gráficos en Android Studio es necesario instalar su librería correspondiente. El código 2.25 indica las líneas que se deben agregar dentro del *gradle* para hacer uso de las clases de `MPAndroidChart`.

```
maven {
    url "https://jitpack.io"
}

dependencies {
    compile 'com.github.PhilJay:MPAndroidChart:v3.0.2'
```

Código 2.25. Clase *MPAndroidChart*

En el código 2.26 se define el contenedor `<ScrollView>` que permite a los elementos que se encuentren dentro de él puedan ser desplazados. En las líneas 11 a la 20 se establecen los límites que tendrá el componente.

```
10      <ScrollView
11          android:layout_width="0dp"
12          tools:layout_constraintRight_creator="1"
13          app:layout_constraintRight_toRightOf="parent"
14          tools:layout_constraintLeft_creator="1"
15          app:layout_constraintLeft_toLeftOf="parent"
16          android:layout_height="0dp"
17          tools:layout_constraintTop_creator="1"
18          tools:layout_constraintBottom_creator="1"
19          app:layout_constraintBottom_toBottomOf="parent"
20          app:layout_constraintTop_toTopOf="parent">
```

Código 2.26. Layout del fragment *fragmentEstadisticas* (Parte 1/1)

En el código 2.27 línea 63 se define el tipo de gráfico que se va a utilizar, en este caso se usó el `BarChart`. En la línea 64 se define un id que permitirá identificar al componente gráfico. En las líneas 65 a la 74 se establecen los límites que tendrá el grafico.

```
63          <com.github.mikephil.charting.charts.BarChart
64              android:id="@+id/barChart"
65              android:layout_height="300dp"
66              android:layout_width="0dp"
67              android:layout_marginEnd="16dp"
68              app:layout_constraintRight_toRightOf="parent"
69              android:layout_marginRight="16dp"
70              android:layout_marginStart="16dp"
71              app:layout_constraintLeft_toLeftOf="parent"
72              android:layout_marginLeft="16dp"
73              app:layout_constraintTop_toBottomOf="@+id/txtEstadisticas"
74              android:layout_marginTop="24dp" />
```

Código 2.27. Layout del fragment *fragmentEstadisticas* (Parte 2/2)

En el código 2.28 se crea el método *getEquipos()* que permite obtener los datos con los nombres de los equipos que van en el eje de las X y su correspondiente cantidad de puntos obtenidos colocándolos en el eje de las Y del gráfico. En la línea 240 se almacenan los datos en el *ArrayList*<> llamado `listaEquipos`. En las líneas 243 a 246 se crea un lazo *for* que permitirá iterar cada uno de los ítems del *ArrayList* para luego pasarlos a un arreglo de *Strings* que es el tipo de elemento que aceptan las clases de la librería `MPAndroidChart`.

```
233                 try{
234                     for(int i=0; i < json.length(); i++){
235                         equipo = new Equipo();
236                         JSONObject jsonObject;
237                         jsonObject = json.getJSONObject(i);
238                         equipo.setNombreEquipo(jsonObject.optString("nombreEquipo"));
239                         equipo.setPuntos(jsonObject.optInt("puntos"));
240                         listaEquipos.add(equipo);
241                         arrayEquipo = new String[listaEquipos.size()];
242                         arrayPorcentaje = new String[listaEquipos.size()];
243                         for (int j = 0; j < arrayEquipo.length; j++ ){
244                             equipo = listaEquipos.get(j);
245                             String nombre = equipo.getNombreEquipo();
246                             arrayEquipo[j] = new String(nombre);
247                         }
248                         for (int k = 0; k < arrayPorcentaje.length; k++ ){
249                             equipo = listaEquipos.get(k);
250                             int puntos = equipo.getPuntos();
251                             arrayPorcentaje[k] = new String(String.valueOf(puntos));
252                         }
```

Código 2.28. Método *getEquipos()*

En el código 2.29 se crea el método `getSameChart()` que lleva como argumentos el tipo de gráfica, la descripción del gráfico, color de texto, color de fondo y la animación a utilizar. El código se muestra en las líneas 104, 105, 106, 107 y 108 respectivamente.

```
102     private Chart getSameChart(Chart chart, String descripcion, int textColor,
103                                int backgroundColor, int animacionY){
104         leyenda(chart);
105         chart.getDescription().setText(descripcion);
106         chart.getDescription().setTextSize(15);
107         chart.setBackgroundColor(backgroundColor);
108         chart.animateY(animacionY);
109         return chart;
110     }
```

Código 2.29. Método *getSameChart()*

En el código 2.30 se crean los métodos Leyenda. En la línea 112 se crea una instancia de la clase *Legend* que permite representar la leyenda que va a tener el gráfico. En la línea 113 se indica al círculo como tipo de forma que va a representar a cada elemento de la leyenda. En las líneas 116 a la 120 se crea un lazo *for* que permite iterar el arreglo *arrayEquipo* que tiene los nombres de cada uno de los equipos y para asignarles un color y colocarlos en la leyenda.

```
111     private void leyenda(Chart chart){
112         Legend legend = chart.getLegend();
113         legend.setForm(Legend.LegendForm.CIRCLE);
114         legend.setHorizontalAlignment(Legend.LegendHorizontalAlignment.CENTER);
115         ArrayList<LegendEntry> entriesLegend = new ArrayList<>();
116         for (int i = 0; i < arrayEquipo.length; i++){
117             LegendEntry entry = new LegendEntry();
118             entry.formColor = colors[i];
119             entry.label = arrayEquipo[i];
120             entriesLegend.add(entry);
121         }
122         legend.setCustom(entriesLegend);
123     }
```

Código 2.30. Método Leyenda()

En el código 2.31 se definen los métodos `getBarEntries()` que mediante el uso de un lazo for permite iterar a cada uno de los puntajes de los equipos. En la línea 128 se establece en el eje Y los puntos alcanzados y en el eje Y la posición de estos. En la línea 133 se establece en el eje X la separación de cada columna según la cantidad de valores que se tenga. En la línea 134 se establece la posición de los títulos de los equipos en la parte inferior del gráfico. En la línea 135 se indica el arreglo *arrayEquipos* que es donde se encuentra los nombres de los equipos. En la línea 139 se establece el espacio que queda en la parte superior del gráfico, para este caso se utiliza la cantidad máxima de puntos que posea un equipo y se le agrega un valor de 5 para que la barra mayor no se una con el límite superior de la gráfica.

```
125     private ArrayList<BarEntry> getBarEntries(){
126         ArrayList<BarEntry> entriesBar = new ArrayList<>();
127         for (int i = 0; i < arrayPuntos.length; i++){
128             entriesBar.add(new BarEntry(i, Float.parseFloat(arrayPuntos[i])));
129         }
130         return entriesBar;
131     }
132     private void ejeX(XAxis axis){
133         axis.setGranularityEnabled(true);
134         axis.setPosition(XAxis.XAxisPosition.BOTTOM);
135         axis.setValueFormatter(new IndexAxisValueFormatter(arrayEquipo));
136         axis.setEnabled(false);
137     }
138     private void ejeYLeft(YAxis axisy){
139         axisy.setSpaceTop(30);
140         axisy.setAxisMinimum(0);
141     }
```

Código 2.31. Métodos `getBarEntries()`, `ejeX()`, `ejeYLeft()`

En el código 2.32 se define el método `createCharts()` que permite crear la gráfica de barras. En la línea 147 se establece el tipo de gráfica a utilizar, la descripción, color de la fuente, el color de fondo y el tiempo de la animación de las barras. En la línea 148 se establece líneas de fondo dentro del gráfico para facilitar la visualización de las barras.

```
146     public void createCharts(){
147         barChart = (BarChart)getSameChart(barChart, "Equipos", Color.RED, Color.CYAN, 3000);
148         barChart.setDrawGridBackground(true);
149         barChart.setDrawBarShadow(true);
150         barChart.setData(getBarData());
151         barChart.invalidate();
152         ejeX(barChart.getXAxis());
153         ejeYLeft(barChart.getAxisLeft());
154         ejeYRigth(barChart.getAxisRight());
155     }
```

Código 2.32. Método `createCharts()`

Para el módulo Videos se utilizó la librería *YouTubeAndroidPlayerApi* que permite reproducir cualquier video de la plataforma YouTube dentro de la aplicación. Esta API se la descargó de la página de Desarrolladores de Google. Una vez descargado el API se lo mueve hacia la dirección del proyecto en la carpeta `\app\libs`. En el código 2.33 línea 60 se hace del API de YouTube. En las líneas 62 a 73 se establece los límites del componente.

```
60      <com.google.android.youtube.player.YouTubePlayerView
61          android:id="@+id/youtubePlayerView"
62          app:layout_constraintRight_toLeftOf="parent"
63          tools:layout_constraintRight_creator="1"
64          tools:layout_constraintLeft_creator="1"
65          app:layout_constraintLeft_toLeftOf="parent"
66          android:layout_marginTop="64dp"
67          app:layout_constraintTop_toBottomOf="@+id/txtVideo"
68          android:layout_marginStart="12dp"
69          android:layout_marginLeft="12dp"
70          android:layout_marginEnd="12dp"
71          android:layout_marginRight="12dp"
72          android:layout_height="280dp"
73          android:layout_width="0dp">
74      </com.google.android.youtube.player.YouTubePlayerView>
```

Código 2.33. Layout de la actividad *Videos*

En el código 2.34 se crea el método `onInitializationSucces` que se genera automáticamente después de heredar la librería *YouTubePlayer.OnInitializedListener* dentro de la clase*VideoActivity.* En la línea 104 se envía al reproductor de *YouTube* la extensión de la URL del video `"azxDhcKYku4"` que se genera una vez compartido el video con YouTube. El método `onInitiallizationFailure()` permite identificar la existencia de un error en el momento de reproducir un video. El método `onActivityResult()` envía una clave de YouTube la misma que se genera al momento de crear una cuenta en la página de Administrador de API de Google.

```
101     @Override
102     public void onInitializationSuccess(YouTubePlayer.Provider provider, YouTubePlayer youTubePlayer, boolean fueRestaurado) {
103         if (!fueRestaurado){
104             youTubePlayer.cueVideo("azxDhcKYku4");
105         }
106     }
107     @Override
108     public void onInitializationFailure(YouTubePlayer.Provider provider, YouTubeInitializationResult youTubeInitializationResult) {
109         if (youTubeInitializationResult.isUserRecoverableError()){
110             youTubeInitializationResult.getErrorDialog(this, 1).show();
111         }
112         else {
113             String error = "Error al iniciar Youtube" + youTubeInitializationResult.toString();
114             Toast.makeText(getApplicationContext(), error, Toast.LENGTH_SHORT).show();
115         }
116     }
117     protected void onActivityResult(int resquesCode, int resultCode, Intent data){
118         if (resultCode == 1){
119             getYoutubePlayerProvider().initialize(claveYoutube, this);
120         }
121     }
```

Código 2.34. Métodos *onInitializationSuccess, onInitiallizationFailure, onActivityResult*

En el código 2.35 se envía al servidor los datos del hincha que comparte el video con la aplicación. El método `enviarDatos()` lleva como parámetros el nombre, apellido del hincha y el enlace del video que va a compartir. En la línea 135 se envían los datos del

hincha hacia el servidor. En las líneas 142 y 147 se informa del resultado de la transacción si fue exitosa o no en la ejecución de la petición.

```
132     private void enviarDatos(String nombreHincha, String apellidoHincha, String enlaceYoutube)  {
133         String ip = getString(R.string.ip);
134         String url = ip +"/WebService.svc/saveHincha/idEquipo/"+ idEquipoHincha + "/nombreHincha/"+
135                 nombreHincha + "/apellidoHincha/" + apellidoHincha + "/diaPublicacion/"+ dia + "/mesPublicacion/" +
136                 mes + "/anioPublicacion/" + anio +"/enlaceVideo/" + enlaceYoutube;
137         JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url, null,
138             (response) → {
141                     Toast.makeText(getApplicationContext(), "Hincha guardado!!!", Toast.LENGTH_SHORT).show();
142             }, (error) → {
146                 Toast.makeText(getApplication(), "Error al enviar", Toast.LENGTH_SHORT).show();
147         });
```

Código 2.35. Método *enviarDatos*

Se crea la clase *FirebaseService* que permite configurar las notificaciones que se reciben desde el servidor. En el código 2.36 en el método `onMessageReceived()` en la línea 34 se condiciona la llegada del mensaje de notificación. En la línea 36 se llama al método `mostrarNotificacion()` el mismo que lleva como parámetros el cuerpo y el título del mensaje de notificación.

```
30     public void onMessageReceived(RemoteMessage remoteMessage) {
31
32         Log.d(TAG, "From: " + remoteMessage.getFrom());
33
34         if (remoteMessage.getNotification() != null){
35             Log.d(TAG, "Notification Message Body: " + remoteMessage.getNotification().getBody());
36             mostrarNotificacion(remoteMessage.getNotification().getTitle(), remoteMessage.getNotification().getBody());
37
38         }
```

Código 2.36. Método *`onMessageReceived()`*

En el código 2.37 se crea el método mostrar `notificación()` que recibe el título y el cuerpo del método `onMessageReceived()`. En la línea 46 se llama a la clase *Intent* que establece al *activity* que se desea abrir en el momento que el usuario de clic en la notificación recibida. En las líneas 50 a la 61 se construye la notificación que el usuario va a recibir, estableciendo el ícono que se va a mostrar, el título y el cuerpo de la notificación, el color, la cantidad de vibraciones que se van a realizar al recibir el mensaje y el sonido que se reproduce cuando la notificación se haya recibido.

```
44      private void mostrarNotificacion(String title, String body) {
45          Intent intent = new Intent(this , EquiposActivity.class);
46          intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
47          PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
48          Uri soundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
49
50          NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this)
51                  .setSmallIcon(R.drawable.ic_accessibility_black_24dp)
52                  .setContentTitle(title)
53                  .setContentText(body)
54                  .setColor(Color.BLUE)
55                  .setPriority(NotificationCompat.PRIORITY_DEFAULT)
56                  .setLights(Color.MAGENTA, 1000, 1000)
57                  .setVibrate(new long[] {1000, 1000, 1000, 1000})
58                  .setDefaults(Notification.DEFAULT_SOUND)
59                  .setSound(soundUri)
60                  .setAutoCancel(true)
61                  .setContentIntent(pendingIntent);
62
63
64          NotificationManager notificationManager = (NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
65          notificationManager.notify(0, notificationBuilder.build());
66
67      }
```

Código 2.37. Método mostrarNotificacion()

Para la creación de las notificaciones se hace uso de los servidores de Firebase de Google, para ello se dirige a la pestaña de Tools > Firebase como se muestra en la figura 2.53.

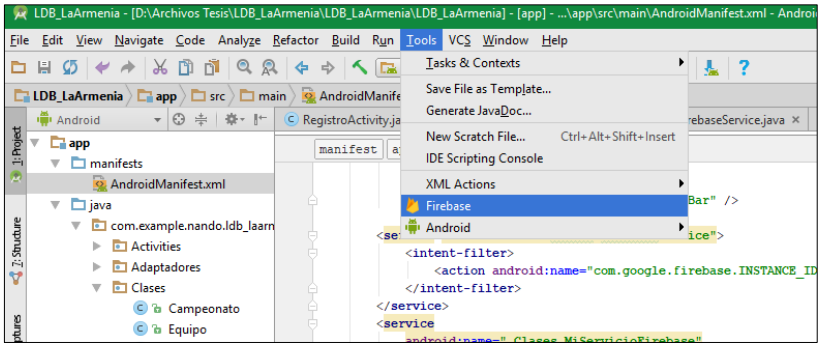


Figura 2.37. Notificaciones en Firebase

Se debe iniciar sesión con una cuenta de Google y posteriormente se selecciona la pestaña de *Cloud Messaging* y *Set up Firebase Cloud Messaging.* Como se muestra en la figura 2.54.

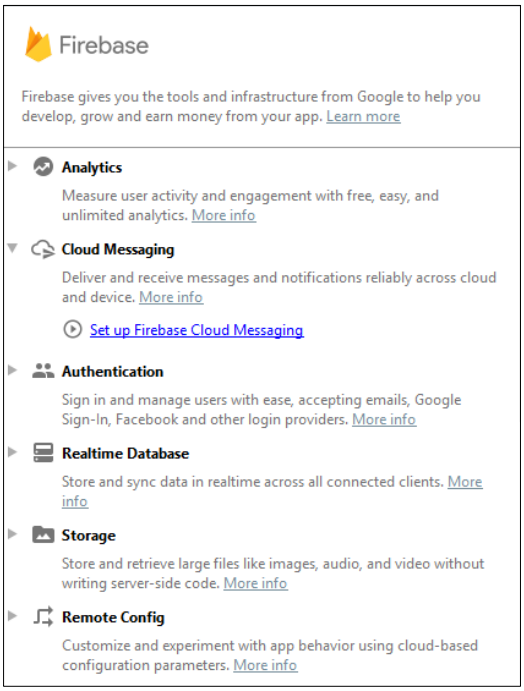


Figura 2.38. Firebase Cloud Messaging

Se siguen los pasos que nos indica el asistente dando los permisos necesarios dentro del AndroidManifest de la aplicación y creando las clases y métodos solicitados, como se muestra en las figuras 2.55 y 2.56.

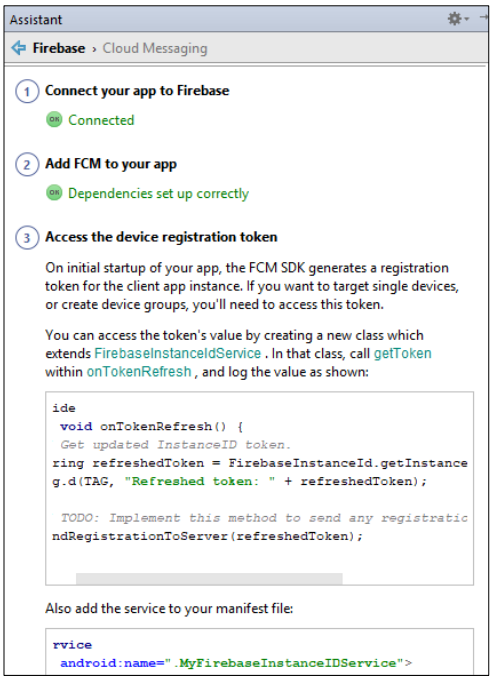


Figura 2.39. Pasos Firebase 1, 2 y 3.

Figura 2.40. Paso Firebase 4

Para el envío de las notificaciones se accede a la consola de *Firebase* desde el asistente y en la sección de *Cloud Messaging* se llenan los campos de cada uno de los pasos requeridos para enviar las notificaciones a todos o solo a un grupo de usuarios que tengan instalada la aplicación, como se indica en la figura 2.57.

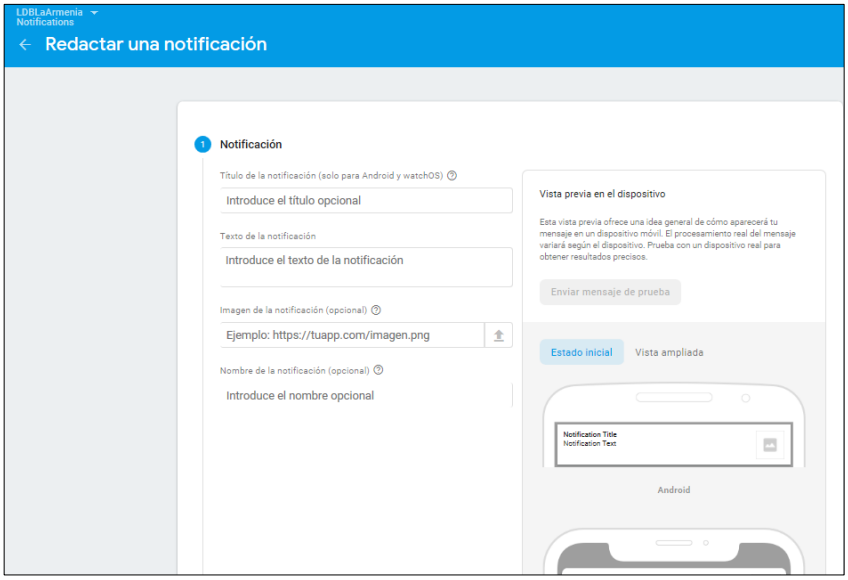


Figura 2.41. Envío de Notificaciones

El cifrado de la información en la transferencia de datos en el inicio de sesión se lo realizó mediante el uso del algoritmo AES como se muestre en el código 2.38. con la creación del método `SecretKeySpec` que permite generar la clave con la que varía el texto a cifrar. En la línea 137 se crea el objeto *MessageDigest*, en la línea 138 se pasa la contraseña a un arreglo de bytes con el estándar “UTF-8” en la línea 139 se agregan los datos al objeto *MessageDigest*, en la línea 140 se crea una instancia de la clase *SecretKeySpec* que lleva como firma la clave y el algoritmo usado para la encriptación.

```
136     private SecretKeySpec generateKey(String password) throws Exception {
137         MessageDigest sha = MessageDigest.getInstance("SHA-256");
138         byte[] key = password.getBytes("UTF-8");
139         key = sha.digest(key);
140         SecretKeySpec secretKey = new SecretKeySpec(key, "AES");
141         return secretKey;
142     }
```

Código 2.38 Método *`generateKey()`*

El método `encriptar()` lleva como firma el texto a encriptar y la clave como se muestra en el código 2.39. En la línea 128 se instancia la clase *SecretKeySpec* con la clave utilizada en la encriptación, en la línea 129 se crea una instancia de la clase *Cipher* indicando el algoritmo utilizado, en la línea 130 se inicia el modo de cifrado, en la línea 131 se indica el texto que se desea encriptar y se lo agrega en un arreglo de bytes y en la línea 132 se

codifica en un formato Base64 que permitirá almacenar el texto encriptado en una variable del tipo `string`.

```
127     private String encriptar(String nombre, String password) throws Exception {
128         SecretKeySpec secretKey = generateKey(password);
129         Cipher cipher = Cipher.getInstance("AES");
130         cipher.init(Cipher.ENCRYPT_MODE, secretKey);
131         byte[] datosEncriptadosBytes = cipher.doFinal(nombre.getBytes());
132         datosEncriptadosString = Base64.encodeToString(datosEncriptadosBytes, Base64.DEFAULT);
133         return datosEncriptadosString;
134     }
```

Código 2.39 Método `encriptar()`

El método `tomarFotografía()` permite al vocal tomar una foto del partido con el resultado de este como se muestra en el código 2.40. En la línea 288 se instancia a la clase *File* estableciendo el directorio de almacenamiento de la foto y la ruta de la misma, en las líneas 291 a la 296 se establece un condicional que permite crear el directorio en el cual se va a almacenar la foto tomada. En la línea 300 se crea un *Intent* que permite capturar la foto tomada para luego almacenarla en la dirección establecida como se muestra en la línea 301.

```
287     private void tomarFotografia() {
288         File fileImagen = new File(Environment.getExternalStorageDirectory(), RUTA_IMAGEN);
289         boolean isCreada = fileImagen.exists();
290         String nombre = "";
291         if (isCreada == false) {
292             isCreada = fileImagen.mkdirs();
293         }
294         if (isCreada == true) {
295             nombre = (System.currentTimeMillis() / 1000) + ".jpg";
296         }
297         path = Environment.getExternalStorageDirectory() + File.separator + RUTA_IMAGEN + File.separator + nombre;
298
299         File imagen = new File(path);
300         Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
301         intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(imagen));
302         startActivityForResult(intent, COD_FOTO);
303     }
```

Código 2.40 Método `tomarFotografía()`

El método `onActivityResult()` recibe la acción del objeto *Intent* establecido en el código anterior, gestionando la selección del usuario. En la línea 319 del código 2.41 se captura la foto tomada por parte del usuario, en la línea 321 se muestra la foto en el *ImageView* `imgVFotoVocalia` y en la línea 320 se llama al método `convertirBitmapTo64()` que permite pasar del formato *Bitmap* a *Base64* para posteriormente almacenarla en la base de datos.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (resultCode == RESULT_OK) {
        switch (requestCode) {
            case COD_FOTO:
                MediaScannerConnection.scanFile(this, new String[]{path}, null,
                        new MediaScannerConnection.OnScanCompletedListener() {
                            @Override
                            public void onScanCompleted(String s, Uri uri) {
                                Toast.makeText(getApplicationContext(), "Ruta" + path, Toast.LENGTH_SHORT).show();
                            }
                        });
                bitmap = BitmapFactory.decodeFile(path);
                convertirBitmapTo64();
                imgVFotoVocalia.setImageBitmap(bitmap);
                break;
        }
    }
}

private void convertirBitmapTo64() {
```

Código 2.41 Método `onActivityResult()`

2.10.2. Interfaces de usuario

A continuación, se muestran interfaces de usuario de la aplicación Android. En la figura 2.58 se muestran los partidos que están por jugarse con sus respectivas fechas y horas de inicio. En la figura 2.59 se muestran las noticias de los equipos cuyos jugadores tienen tarjetas amarillas o rojas.

Figura 2.42. UI de Partidos

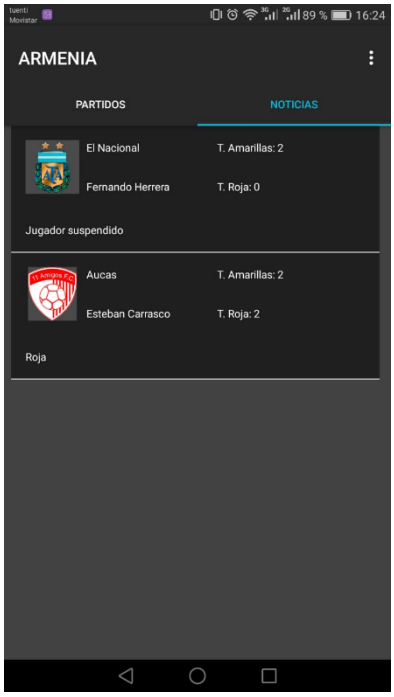


Figura 2.43. UI de Noticias

En la figura 2.60 se muestra la interfaz de usuario de inicio de sesión para el Administrador que permite gestionar los datos de los módulos de la aplicación Android.

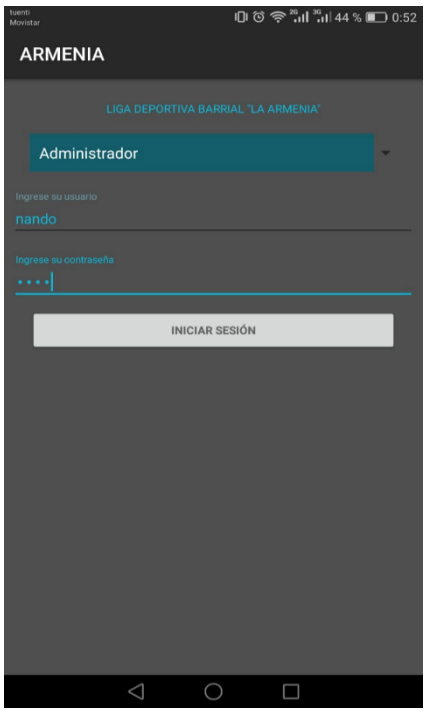
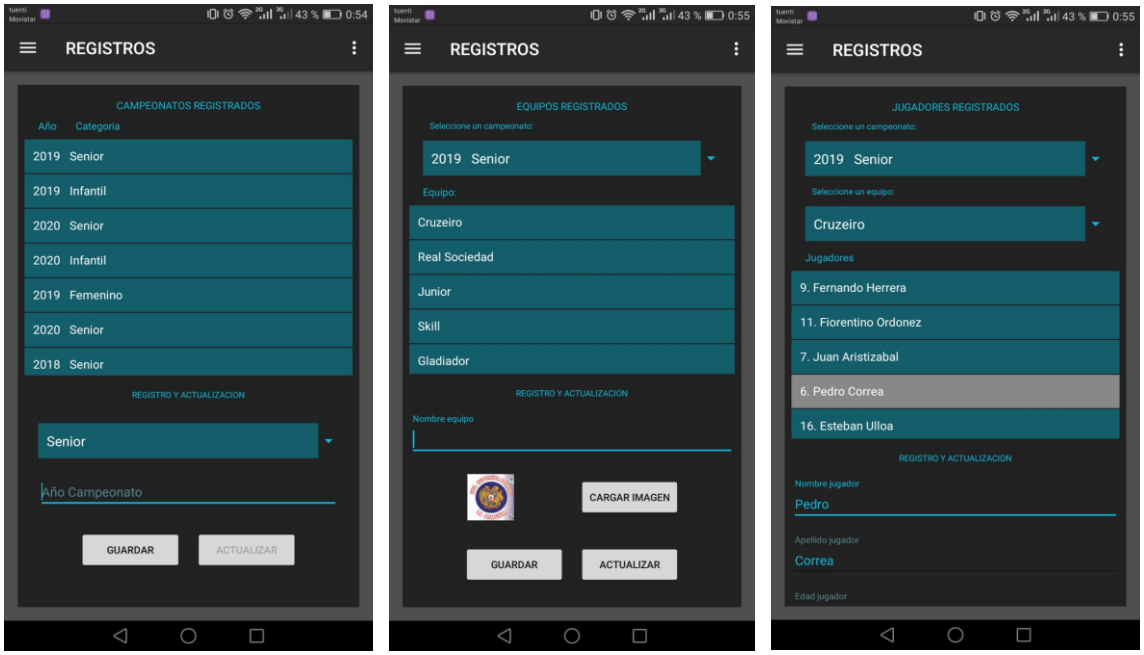


Figura 2.44. UI de Inicio de sesión

En la figura 2.61 se muestra la interfaz de usuario que gestiona los datos de la aplicación. La parte (a) permite guardar los datos de los campeonatos. La parte (b) almacena los datos de los equipos y la parte (c) guarda los datos de los jugadores.



(a) Registro *Calendario* (b) Registro *Equipos* (c) Registro *Jugadores*

Figura 2.45. UI Gestión de datos

En la figura 2.62 se muestra la información detallada de los partidos mostrando el tiempo de juego, marcador, alineaciones de los equipos y la opción de predecir al equipo ganador.

Figura 2.46. UI Detalle partidos

En la figura 2.63 se muestra la interfaz de usuario de la tabla de posiciones de los equipos según la cantidad de puntos que hayan acumulado y goles marcados.

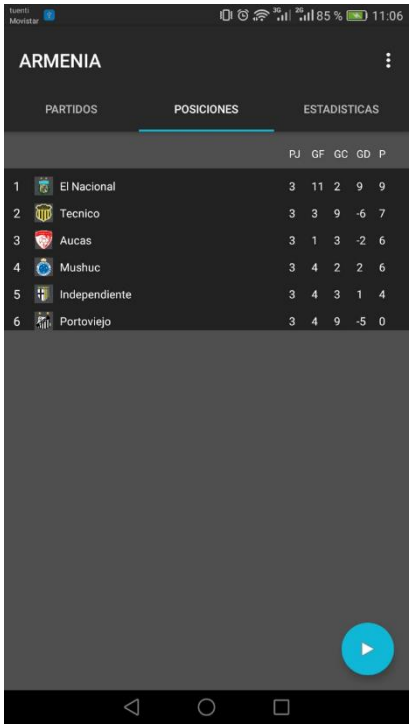


Figura 2.47. UI Tabla de Posiciones

En la figura 2.64 se muestra un gráfico indicando las estadísticas de cada uno de los equipos según la cantidad de puntos que haya conseguido.

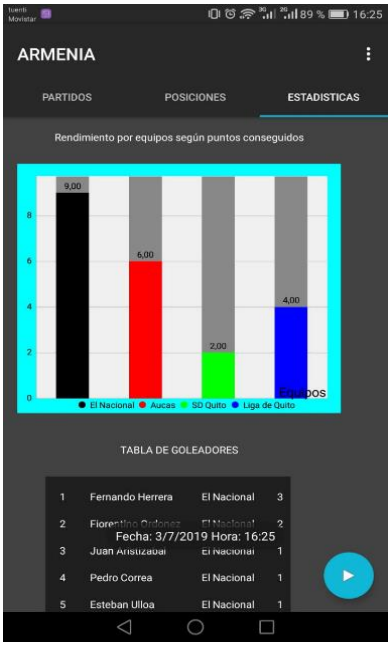
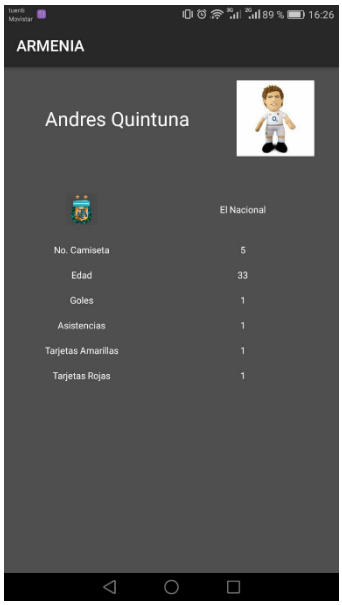


Figura 2.48. UI Estadísticas

En la figura 2.65 parte (a) se muestra la información de los equipos y en la parte (b) se indica la información de los jugadores.

(a) Detalle *Equipo*



(b) Detalle *Jugador*

Figura 2.49. UI detalles Equipo – Jugador

En la figura 2.66 se muestra la interfaz de usuario de los videos compartidos por los hinchas de cada uno de los equipos.

Figura 2.50. UI de Videos

En la figura 2.67 se muestra la interfaz de usuario en la que se guardan los datos de los hinchas que comparten sus videos con la aplicación.

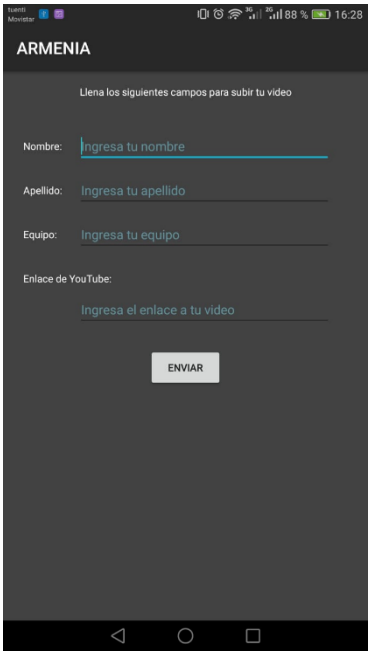


Figura 2.51. UI Datos de hinchas

3. RESULTADOS Y DISCUSIÓN

En este capítulo se realiza la verificación de la funcionalidad del prototipo tanto en la parte del servidor como del cliente Android, para lo cual se realizaron las siguientes pruebas:

- Pruebas de autenticación de usuario.
- Pruebas de creación de nuevos registros como campeonatos, equipos de fútbol, jugadores, partidos y usuarios.
- Pruebas de carga de imágenes como: fotos de jugadores y escudos de los equipos.
- Pruebas de actualización de registros de elementos como campeonatos, equipos, jugadores y partidos.
- Pruebas de eliminación de registros.
- Pruebas de carga de videos por parte del usuario.
- Pruebas de descarga de información de campeonatos, equipos, jugadores y partidos.

Adicionalmente, se realizaron 3 entrevistas a la administración, 4 a vocalía y 5 a los dirigentes de los equipos de la Liga Deportiva Barrial “La Armenia” con la finalidad de constatar la satisfacción con el prototipo desarrollado.

3.1. Prueba de funcionamiento

3.1.1. Pruebas de funcionamiento de la base de datos

Para comprobar el funcionamiento de la base de datos se realizarán sentencias SQL que permitan verificar la funcionalidad de la base de datos.

En la figura 3.1 se indican los resultados de la consulta a la tabla `Partido` mediante el uso de la sentencia `select * from Partido`. Aquí se indica a los equipos que van a disputar el partido, la hora y la fecha del encuentro, los goles que van marcando cada uno de los equipos, alineación con la que van a jugar cada uno de los equipos y las apuestas al equipo ganador realizadas por los usuarios.

	idPartido	idCampeonato	equipoLocal	equipoVisitante	golesLocal	golesVisitante	alineacionLocal	alineacionVisitante	nombreDia	dia	mes
1	1	1	Cruzeiro	Real Sociedad	3	2	442	352	Domingo	22	junio
2	2	1	Junior	Skill	1	3	352	451	Sabado	15	enero
3	3	1	Gladiador	Good Friends	3	2	442	352	Domingo	17	febrer

Figura 3.1 Resultados de la consulta a la tabla *Partido*

En la figura 3.2 se muestran los resultados de la consulta a la tabla `Jugador` mediante el uso de la sentencia `select * from Jugador` en la que se muestra el nombre, apellido el identificador que permite saber en que equipo se encuentra cada uno de los jugadores, edad, nacionalidad, goles, asistencias, posición en la que se ubica cada jugador, número de camiseta, tarjetas amarillas, tarjetas rojas y la foto del jugador.

	nombreJugador	apellidoJugador	edadJugador	nacionalidadJugador	asistenciasJugador	datoImagenJugador
1	Fernando	Herrera	28	ecuatoriana	3	/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAsICAgICAsICAsG
2	Fiorentino	Ordonez	21	ecuatoriana	2	/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAsICAgICAsICAsG
3	Juan	Aristizabal	23	ecuatoriana	3	/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAsICAgICAsICAsG
4	Pedro	Correa	22	ecuatoriana	0	/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAsICAgICAsICAsG
5	Esteban	Ulloa	19	ecuatoriana	2	/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAsICAgICAsICAsG
6	Ernesto	Gangotena	30	brasilena	1	/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAsICAgICAsICAsG
7	Bryan	Quishpe	26	argentina	8	/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAsICAgICAsICAsG

Figura 3.2. Resultados de la consulta a la tabla *Jugador*

En la figura 3.3 se muestran los resultados de la consulta a la tabla `Equipo` mediante el uso de la sentencia `select * from Equipo` en la que se muestra el nombre del equipo, el identificador que permite saber en qué campeonato está inscrito el equipo, el escudo que representa al equipo y la fecha y hora del próximo partido a disputarse.

	idCampeonato	nombreEquipo	escudoEquipo	diaPartido	mesPartido	anioPartido	horaPartido
1	1	Cruzeiro	iVBORw0KGgoAAAANSUhEUgAAAOEAAADhCAYAAAA+s9J6AAAAA...	7	8	2019	15
2	1	Real Sociedad	iVBORw0KGgoAAAANSUhEUgAAANYAAADrCAYAAAAVB2hdAAAA...	7	8	2019	15
3	1	Junior	/9j/4AAQSkZJRgABAQEAYABgAAD/4QAiRXhpZgAATU0AKgAAAAg...	13	7	2019	22
4	1	Skill	iVBORw0KGgoAAAANSUhEUgAAAOEAAADhCAYAAAA+s9J6AAAAA...	15	7	2019	22
5	1	Gladiador	/9j/4AAQSkZJRgABAQEAYABgAAD/4QAiRXhpZgAATU0AKgAAAAg...	16	7	2019	11
6	1	Good Friends	iVBORw0KGgoAAAANSUhEUgAAAOEAAADhCAYAAAA+s9J6AAAAA...	15	7	2019	11

Figura 3.3. Resultados de la consulta a la tabla *Equipo*

En la figura 3.4 se muestran los resultados de la consulta a la tabla `Posiciones` mediante el uso de la sentencia `select * from Posiciones` en la que se muestra el identificador que permite establecer la posición de cada uno de los equipos, partidos jugados, ganados, perdidos, empatados, goles a favor, goles en contra y la cantidad de puntos acumulados de cada uno de los equipos.

	idPosicion	idEquipo	partidosJugados	partidosGanados	partidosPerdidos	partidosEmpatados	golesFavor	golesContra
1	1	1	2	3	2	1	2	3
2	3	3	1	0	0	2	2	1
3	4	4	1	1	1	1	2	3
4	5	7	2	1	0	1	3	2
5	11	8	3	2	1	0	4	2

Figura 3.4. Resultados de la consulta a la tabla *Posiciones*

En la figura 3.5 se muestran los resultados de la consulta a la tabla Usuario mediante el uso de la sentencia `select * from Usuario` en la que se muestra el nombre de los administradores del prototipo y la contraseña con la cual pueden iniciar sesión.

	idLogin	nombreUsuario	password
1	1	nando	1234
2	2	jose	4321
3	3	caro	1234

Figura 3.5. Resultados de la consulta a la tabla Usuario

3.1.2. Prueba de funcionamiento del servicio WCF

Para comprobar la funcionalidad del servidor se utilizó Postman que es una herramienta que permite enviar peticiones y obtener respuestas HTTP.

En la figura 3.6 se muestra la petición realizada al servidor que es del tipo GET permitiendo obtener los datos desde el servidor realizando la consulta a la base de datos a través del servicio WCF, en la parte inferior se muestra el estado de la petición que en este caso es 200 OK indicando que la petición se la realizó exitosamente.

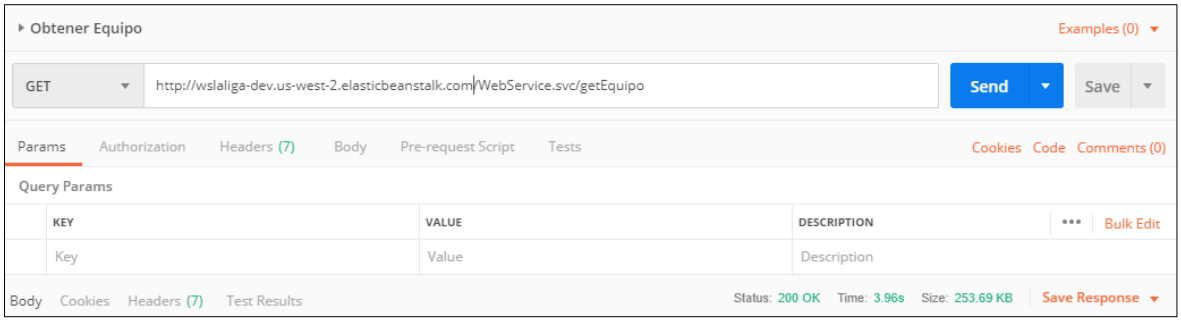


Figura 3.6. Petición GET al servidor WCF

En la figura 3.7 se muestra la respuesta a la petición GET realizada anteriormente.

Figura 3.7. Resultado a la petición GET

En la figura 3.8 se presenta una petición realizada al servidor del tipo POST que permite guardar los datos del Director Técnico como el nombre, apellido, edad, nacionalidad y un identificador que permite determinar al equipo que entrena. Al momento de realizar la petición se recibe como respuesta el mensaje *200 OK* indicando que la petición se realizó exitosamente.

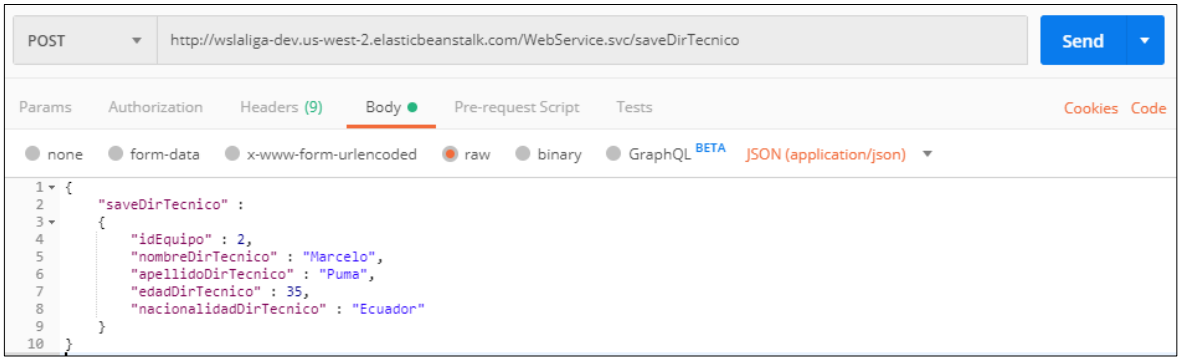


Figura 3.8. Petición POST guardar datos Director Técnico

En la figura 3.9 se indica que los datos del Director Técnico se registraron exitosamente en la base de datos del servidor con el identificador igual a 6 mediante el uso de la consulta: `select * from DirTecnico.`

	idDirTecnico	idEquipo	nombreDirTecnico	apellidoDirTecnico	edadDirTecnico	nacionalidadDirTecnico
1	1	1	JuanitoLimana	Martitegui	0	Colombia
2	6	2	Marcelo	Puma	35	Ecuador

Figura 3.9. Resultados de la consulta a la tabla *DirTecnico*

En la figura 3.10 se presenta una petición realizada al servidor del tipo POST que permite actualizar los datos del Director Técnico, los mismos que pueden ser todos los parámetros o solamente uno de ellos. Al momento de realizar la petición se recibe como respuesta el mensaje *200 OK* indicando que la petición se realizó exitosamente.

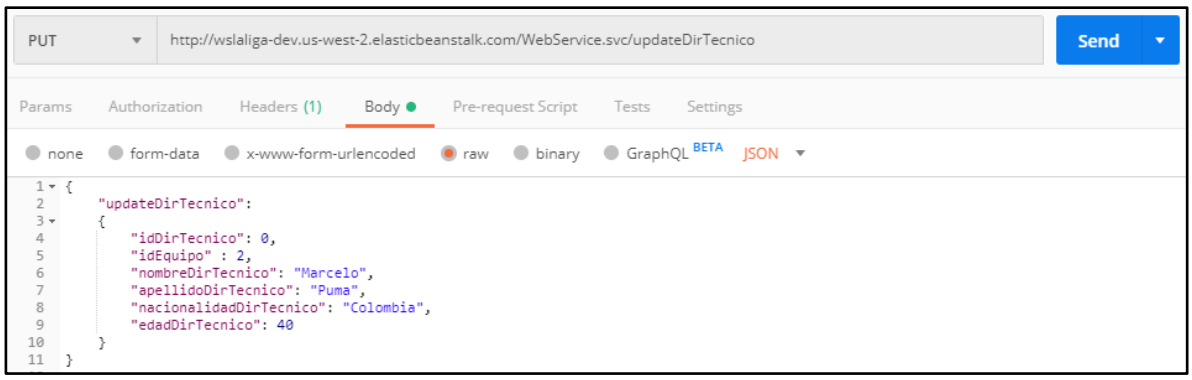


Figura 3.10. Petición POST actualizar datos de la tabla `DirTecnico`

En la figura 3.11 se indica que la edad del Director Técnico con identificador 6 se actualizaron correctamente cambiando la edad de 35 a 40 años, mediante el uso de la consulta `select * from DirTecnico.`

	idDirTecnico	idEquipo	nombreDirTecnico	apellidoDirTecnico	edadDirTecnico	nacionalidadDirTecnico
1	1	1	JuanitoLimana	Martitegui	0	Colombia
2	6	2	Marcelo	Puma	40	Ecuador

Figura 3.11. Resultados de la consulta a la tabla *DirTecnico*

En la figura 3.12 se presenta una petición realizada al servidor del tipo DELETE que permite eliminar un registro completo de la base de datos especificando el identificador de la columna que se desea eliminar. Al momento de realizar la petición se recibe como respuesta el mensaje *200 OK* indicando que la petición se realizó exitosamente.

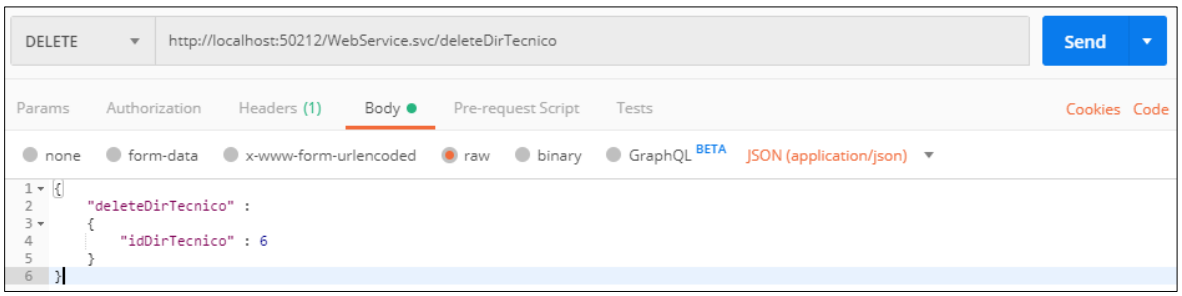


Figura 3.12. Petición DELETE eliminar registro de la tabla *DirTecnico*

En la figura 3.13 se indica que el Director Técnico con identificador 6 se ha eliminado exitosamente, mediante el uso de la consulta `select * from DirTecnico`.

	idDirTecnico	idEquipo	nombreDirTecnico	apellidoDirTecnico	edadDirTecnico	nacionalidadDirTecnico
1	1	1	JuanitoLimana	Martitegui	0	Colombia

Figura 3.13. Resultados de la consulta a la tabla DirTecnico

3.1.3. Prueba de funcionamiento del Cliente Android

Para validar el módulo de control de acceso se realizó la siguiente prueba:

3.1.3.1. Autenticación de usuario

En la pantalla principal de la aplicación aparecerá la vista que se muestra en la figura 3.14 en la cual se requiere llenar los campos `usuario` y `contraseña` del cliente, que se registró anteriormente.

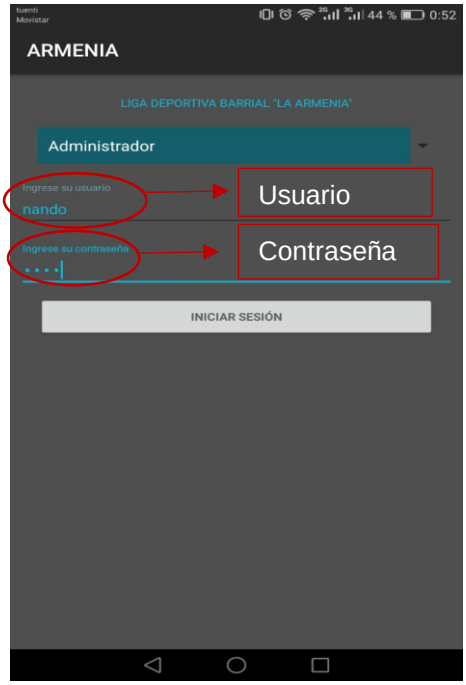


Figura 3.14. Ejemplo de inicio de sesión

Una vez se ingrese los datos de los campos de la pantalla de *login* y se presione el botón `iniciar sesión`, se redirigirá a una nueva vista en la que el usuario será notificado con el mensaje de `"acceso exitoso"`, como se muestra en la siguiente figura 3.15.

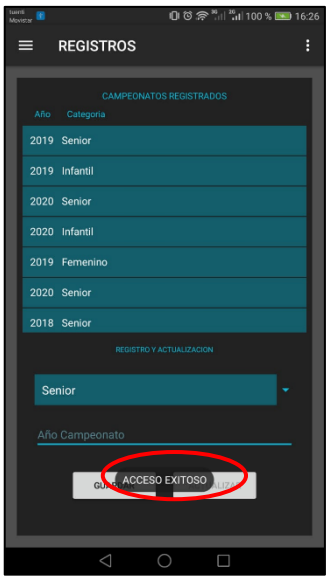


Figura 3.15. Ejemplo acceso exitoso

En caso de que el usuario con cuenta no sea del administrador o la contraseña sea incorrecta el inicio de sesión será fallido, lo cual se mostrará con una notificación como se muestran en la siguiente figura:

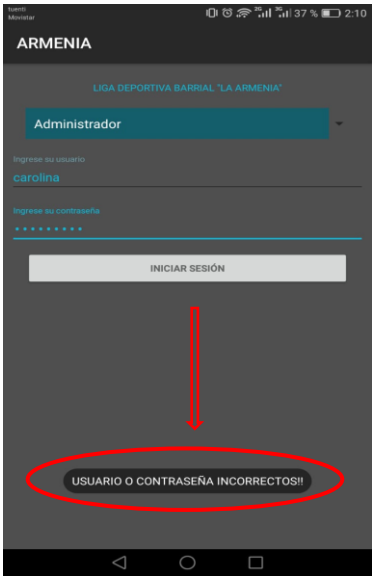


Figura 3.16. Inicio de sesión fallida

3.1.3.2. Creación de nuevos registros

La aplicación contiene varios módulos que requieren registrar información.
A continuación, se indica como ingresar dicha información en las distintas ventanas.

- **Registro Calendario de partidos**

El registro de calendario de partidos consiste en seleccionar en el spinner la categoría del campeonato e ingresa el año en el que se desarrollará el mismo y finalmente se almacenará la información al presionar el botón `guardar`, como se muestra en la figura 3.17.

Figura 3.17. Ejemplo Registro Campeonato

Una vez guardada la información registrada, se mostrará la notificación "`campeonato guardado con éxito`", como se muestra en la figura 3.18.

Figura 3.18. Ejemplo de Registro campeonato

- **Compartir videos de los usuarios**

El usuario podrá compartir videos en la aplicación, mediante los siguientes pasos:

1. Seleccionar un partido del cual se quiere obtener información.
2. Presionar el icono de videos.
3. Presionar el botón `Subir video`.
4. Ingresar los datos solicitados como se muestra en la figura 3.19
5. Presionar el botón `Enviar`

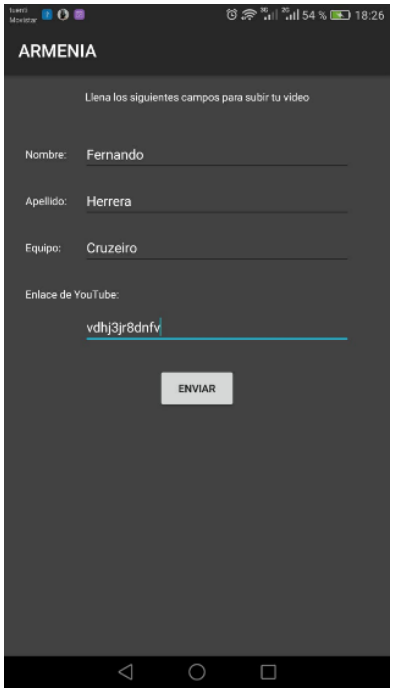


Figura 3.19. Ejemplo de subir videos

Una vez enviado el video con éxito se notificará al cliente con la notificación `"Hincha guardado"` como se muestra en la figura 3.20

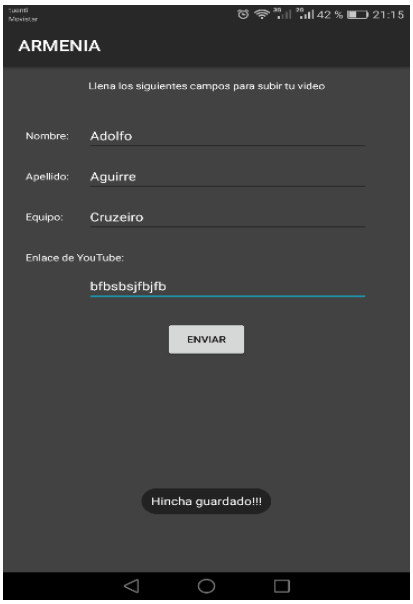


Figura 3.20. Resultado Subir video

3.1.3.3. Modificación de registros

La modificación de registros para cada módulo se llevará a cabo como se indica a continuación:

- **Actualizar datos de Campeonato**

La actualización de los datos del campeonato se realiza en la pantalla que se muestra en la figura 3.21, en la cual se cambian/ los campos requeridos al presionar el botón `Actualizar`.

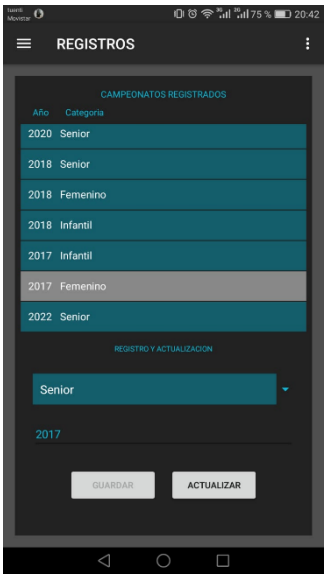


Figura 3.21. Ejemplo actualizar datos de campeonato

Finalmente, al realizarse los cambios con éxito aparecerá la notificación `"Campeonato actualizado con éxito"` como se muestra en la figura 3.22.

Figura 3.22. Resultado de actualizar campeonato

3.1.3.4. Eliminación de registros

Para eliminar un campeonato se requiere mantener presionado el registro a eliminar e inmediatamente aparecerá un cuadro de diálogo que solicita confirmar la actividad como se muestra en la figura 3.23.

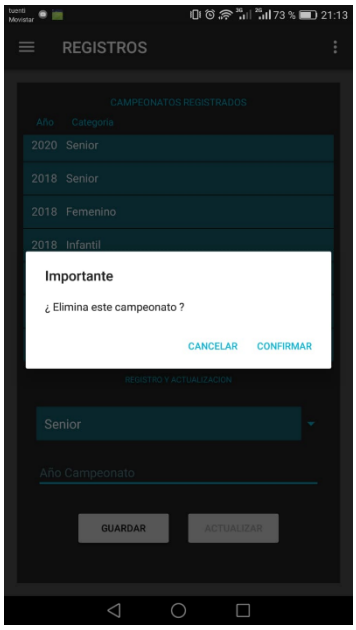


Figura 3.23. Ejemplo de eliminar un registro

Finalmente aparecerá el mensaje "`Campeonato eliminado con éxito`" como se muestra en la figura 3.24.

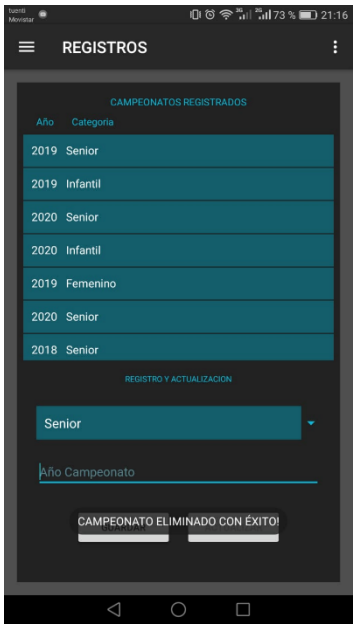


Figura 3.24. Resultado eliminar registro

3.1.3.5. Presentación de información registrada

A continuación, se muestra la presentación de la información registrada en las pantallas del prototipo:

- **Mostrar el calendario de partidos**

1. Abrir el prototipo desde el menú de aplicaciones.
2. En la figura 3.25 se muestra el calendario de los partidos.

Figura 3.25. Ejemplo calendario de partidos

- **Mostrar Noticias**

1. Abrir el prototipo desde el menú de aplicaciones.
2. Seleccionar la pestaña de Noticias como se muestra en la figura 3.26.

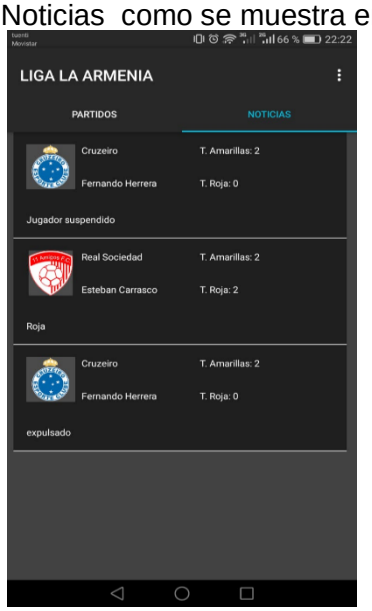


Figura 3.26. Ejemplo de noticias

- **Detalles de partidos**

1. Seleccionar la pestaña de partidos como se muestra en la figura 3.27.
2. Seleccionar el partido que se desea obtener información.
3. Seleccionar el botón con el nombre de un equipo para ver su alineación como se muestra en la figura 3.28.

Figura 3.27. Ejemplo detalle de partido

Finalmente, la siguiente pantalla muestra las alineaciones de cada uno de los equipos:

Figura 3.28. Ejemplo alineaciones de equipos

- **Estadísticas de los partidos**

1. Seleccionar un partido del cual se quiera obtener información.
2. Seleccionar la pestaña de `Estadísticas` como se muestra en la siguiente:

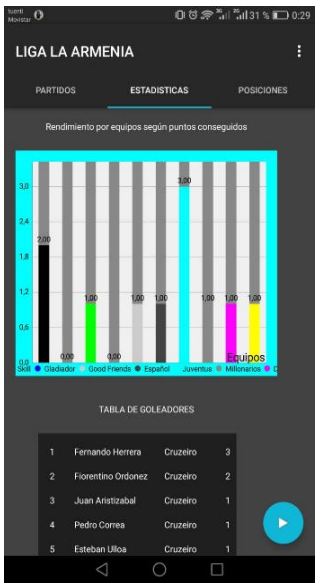


Figura 3.29. Ejemplo estadísticas de equipos

La figura 3.29 muestra la tabla de goleadores y un gráfico que permite visualizar el rendimiento de cada equipo según sus puntos conseguidos.

- **Tabla de posiciones**

1. Seleccionar un partido del cual se quiera obtener información.
2. Seleccionar la pestaña de Posiciones como se muestra en la figura 3.30.

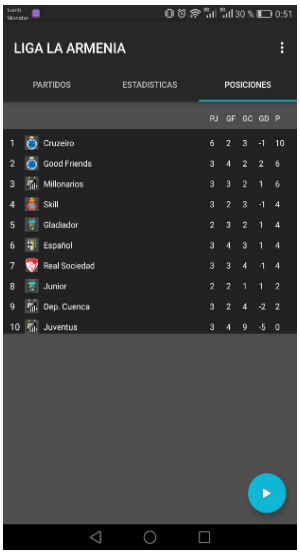


Figura 3.30. Ejemplo de tabla de posiciones

En la figura 3.30 se muestra la tabla de posiciones de todos los equipos que conforman un campeonato ordenados según la cantidad de puntos que hayan conseguido.

- **Videos de usuarios:**

1. Seleccionar un partido del cual se quiera obtener información.
2. Presionar el ícono de videos como se muestra en la figura 3.31.

Figura 3.31. Ejemplo de Videos de usuarios

A continuación, se muestra en la figura 3.32 los videos compartidos por los usuarios con el prototipo.

Figura 3.32. Resultado de videos de usuarios

3.2. Pruebas de validación

La administración de la liga barrial utilizó el prototipo con la finalidad de realizar las pruebas de validación respectivas.

Para llevar a cabo lo anteriormente mencionado se realizaron 3 entrevistas a los administradores, 4 a vocalía y 5 a los dirigentes de los equipos de la liga barrial cuyos formatos se encuentran en el anexo E.

A continuación, se presentan los resultados obtenidos de la encuesta de validación:

a. La primera pregunta permite conocer si en el prototipo se muestran los partidos correspondientes a la Liga Deportiva “La Armenia”. En la figura 3.33 se muestra que el 100% de los encuestados corroboraron esta información.

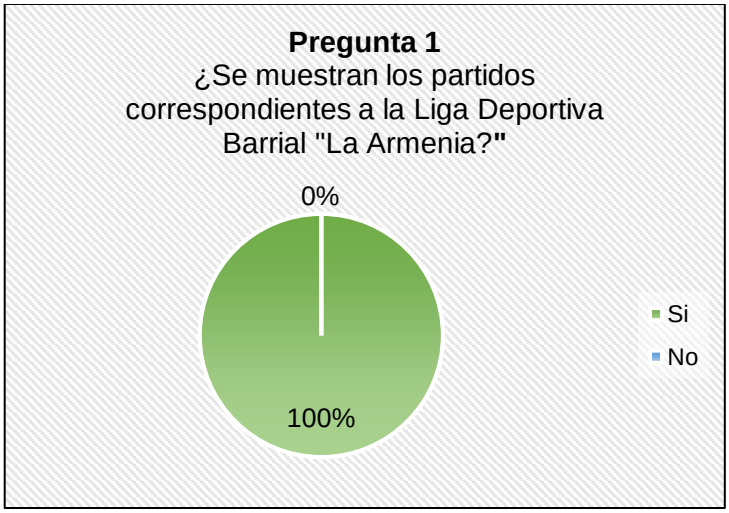


Figura 3.33. Respuesta a la primera pregunta

b. La segunda pregunta está vinculada a la primera debido a que intenta conocer si al usuario le parece amigable la interfaz. En la figura 3.34 se indica que el 91,67% de los encuestados se encuentra conforme con la interfaz, mientras que el 8,33% de los entrevistados creen que podría ser más amigable.

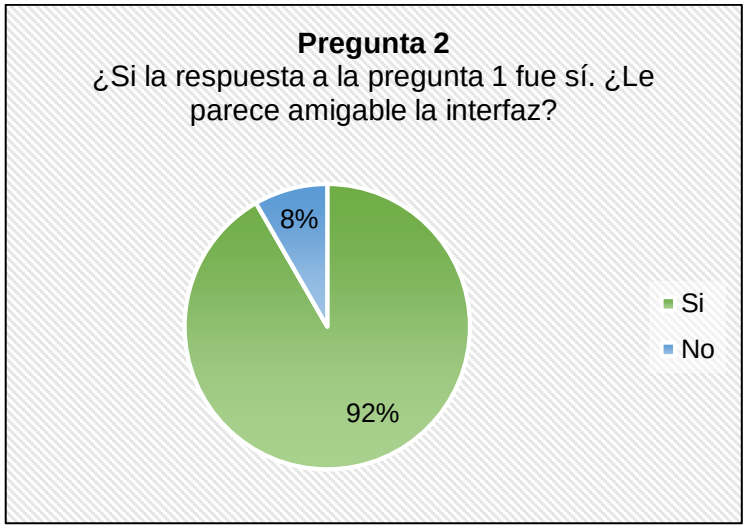


Figura 3.34. Respuesta a la segunda pregunta

c. La tercera pregunta permite conocer si se diferencia en el prototipo los partidos que se encuentran jugando en tiempo real de los que no. En la figura 3.35 se observa que el 100% de los encuestados diferencian el estado de los partidos.

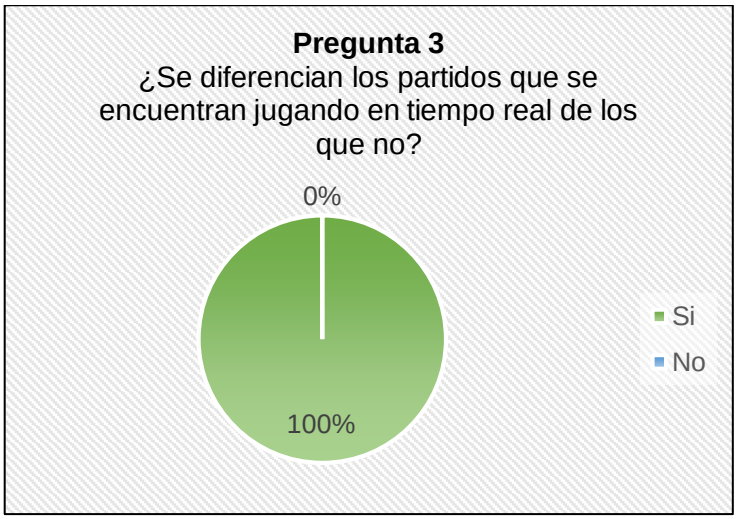


Figura 3.35. Respuesta a la tercera pregunta

d. La cuarta pregunta permite verificar si en la interfaz se muestra el tiempo transcurrido de un partido y el marcador. La figura 3.36 indica que el 100% de los encuestados visualizan el correcto funcionamiento de la información del partido.

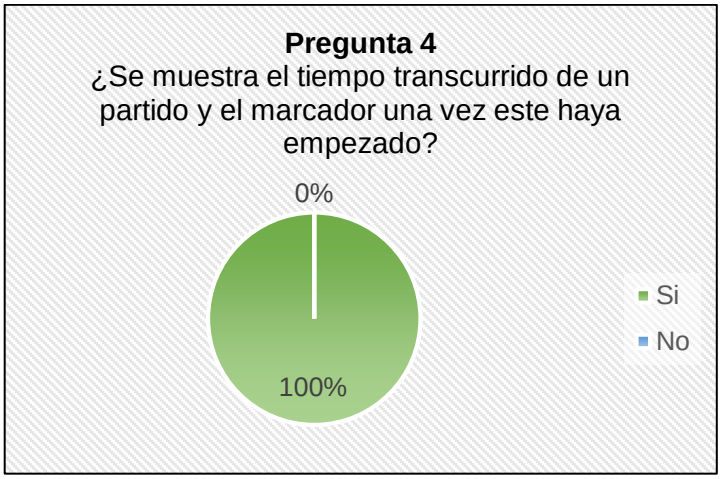


Figura 3.36. Respuesta a la cuarta pregunta

e. La quinta pregunta permite confirmar si al momento de crear un nuevo usuario administrador se ejecuta la transacción exitosamente. En la figura 3.37 se indica que el 100% de los entrevistados consiguieron registrar un nuevo perfil de administrador.

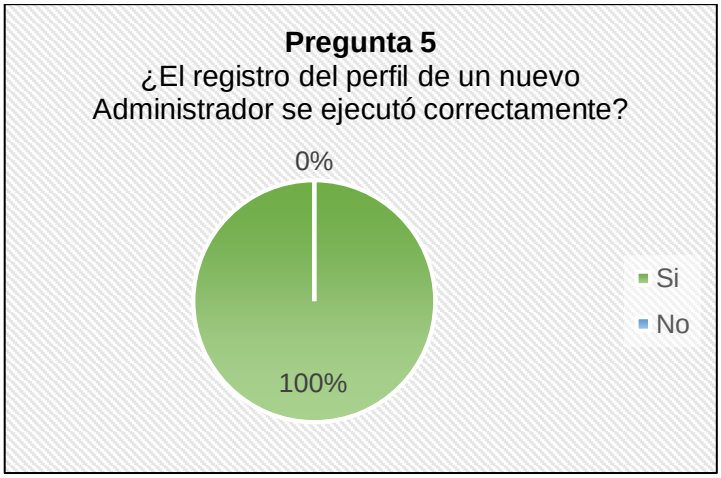


Figura 3.37. Respuesta a la quinta pregunta

f. La sexta pregunta permite confirmar si en la pestaña tabla de posiciones presenta todos los equipos del campeonato ordenados según el porcentaje acumulado. En la figura 3.38 indica que el 100% de los entrevistados visualizaron esta información de acuerdo a lo especificado.

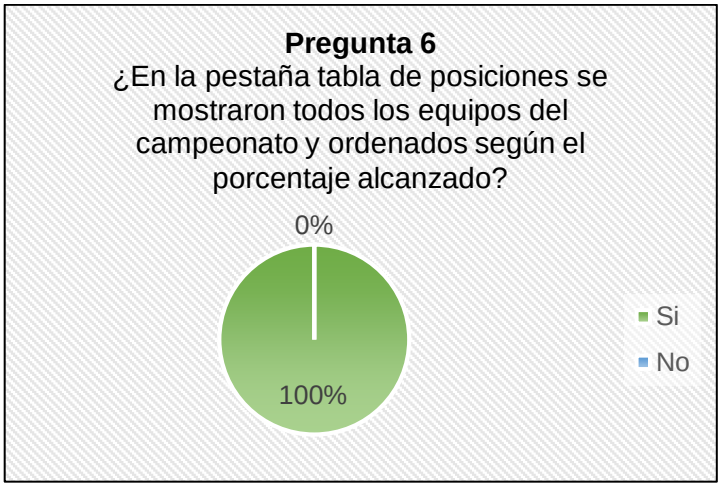


Figura 3.38. Respuesta a la sexta pregunta

g. La séptima pregunta pretende conocer si en la tabla de posiciones se resalta a los equipos que se encuentran jugando en vivo con su respectivo marcador. En la figura 3.39 se muestra que el 100% de los entrevistados confirmaron la visualización de esta información.

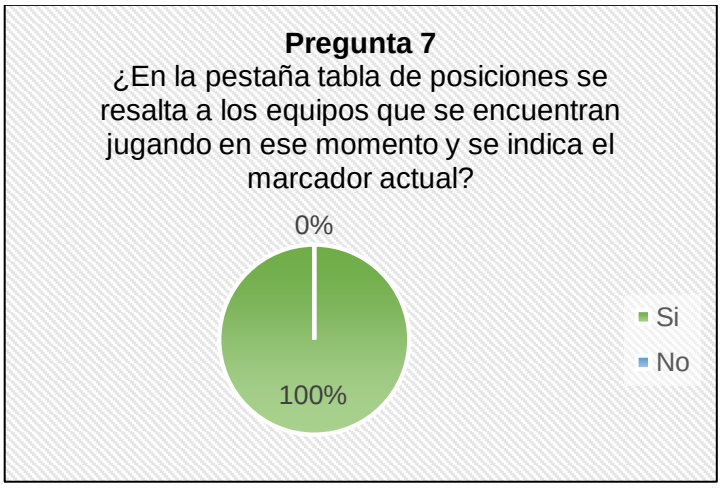


Figura 3.39. Respuesta a la séptima pregunta

h. La octava pregunta pretende conocer si en la sección de noticias muestra a los jugadores suspendidos por acumulación de tarjetas. La figura 3.39 muestra que el 100% de los entrevistados visualizaron esta sección de noticias.

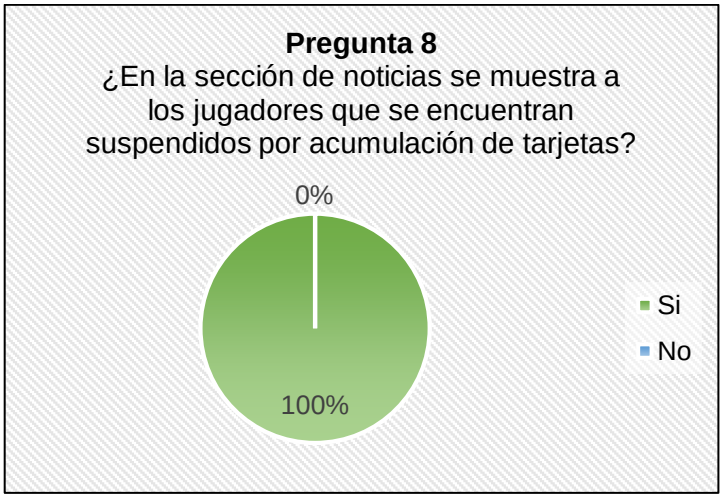


Figura 3.40. Respuesta a la octava pregunta

i. La novena pregunta permite confirmar que la gráfica que se muestra en la pestaña estadística facilita la visualización del rendimiento de los equipos. En la figura 3.41 se observa que el 91,63% de los entrevistados confirman una fácil comprensión de dicha información, mientras, que el 8,33% de los encuestados opinan que se podría mejorar esta presentación.

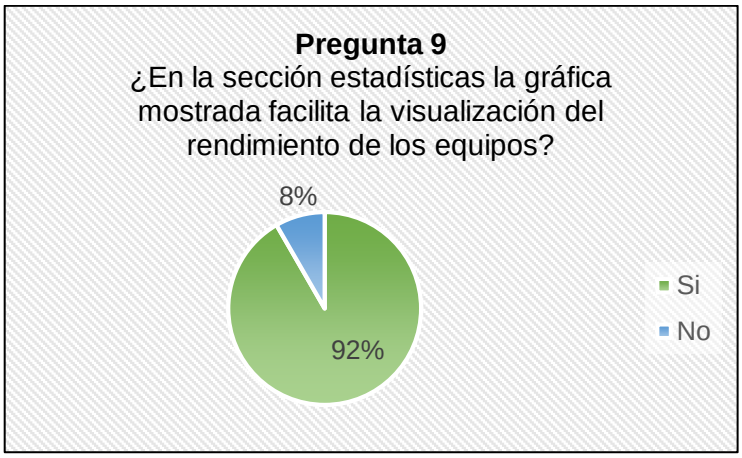


Figura 3.41. Respuesta a la novena pregunta

j. La décima pregunta pretende confirmar que en la sección registros se guarda con facilidad los datos de los equipos. La gráfica 3.42 indica que el 100% de los entrevistados obtuvieron éxito en esta actividad.

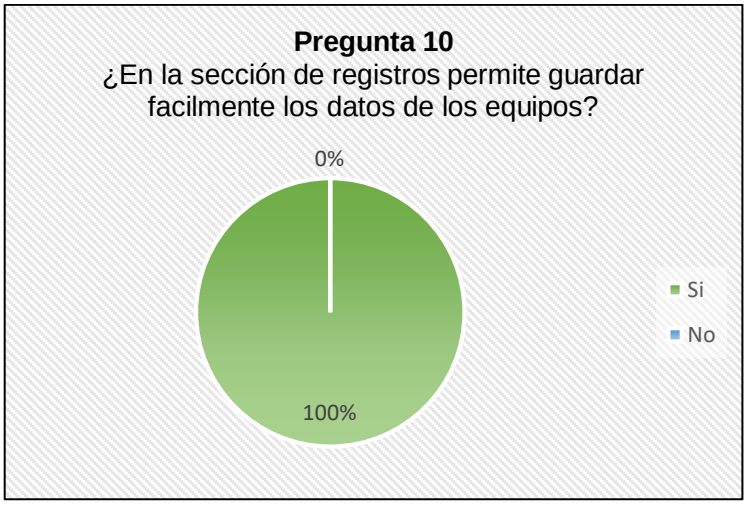


Figura 3.42. Respuesta a la décima pregunta

k. La décima primera pregunta permite confirmar si la actualización de los datos de un partido se realiza correctamente y en tiempo real. La gráfica 3.43 indica que el 100% de los entrevistados obtuvieron éxito en esta actividad.

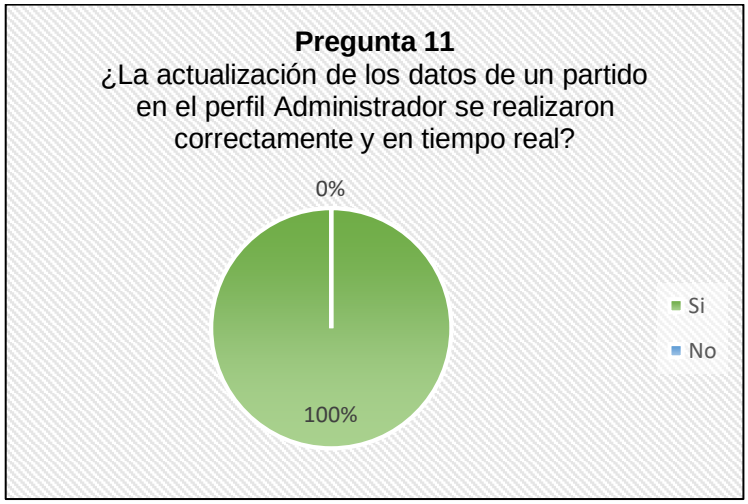


Figura 3.43. Respuesta a la décima primera pregunta

A continuación, en la tabla 3.1 se muestran los resultados de las encuestas realizadas en las entrevistas a los usuarios. Estos resultados indican que los requerimientos establecidos en un inicio se cumplieron con un alto porcentaje de aceptación por parte de los usuarios finales y que la aplicación es bastante amigable en su uso y fácil de utilizar mostrando la información necesaria en cuanto a su equipo favorito se trata como la tabla de posiciones el calendario de los partidos y los partidos que se encuentran jugando en tiempo real.

Tabla 3.1. Resumen de encuestas de validación

N°	Pregunta	Resultados
1.	¿Se muestran los partidos correspondientes a la Liga Deportiva Barrial “La Armenia”?	
	• Si	100%
	• No	0%
2.	Si la respuesta a la pregunta 1 fue sí. ¿Le parece amigable la interfaz gráfica?	
	• Si	91,67%
	• No	8,33%
3.	¿Se diferencian los partidos que se encuentran jugando en tiempo real de los que no?	
	• Si	100%
	• No	0%
4.	¿Se muestra el tiempo transcurrido de un partido y el marcador una vez este haya empezado?	
	• Si	100%
	• No	0%
5.	¿El registro del perfil de un nuevo Administrador se ejecutó correctamente?	
	• Si	100%
	• No	0%
6.	¿En la pestaña tabla de posiciones se mostraron todos los equipos del campeonato y ordenados según el puntaje que hayan acumulado?	
	• Si	100%
	• No	0%
7.	¿En la tabla de posiciones se resalta a los equipos que se encuentran jugando en ese momento y se indica el marcador actual?	
	• Si	100%
	• No	0%
8.	¿En la sección de noticias se muestra a los jugadores que se encuentran suspendidos por acumulación de tarjetas?	
	• Si	100%
	• No	0%
9.	¿En la sección Estadísticas la gráfica mostrada facilita la visualización del rendimiento de los equipos?	
	• Si	91.67%
	• No	8.33%

10.	¿En la sección de Registros permite guardar fácilmente los datos de los equipos?	
	• Si	100%
	• No	0%
11.	¿La actualización de los datos de un partido en el perfil se realizaron correctamente y en tiempo real?	
	• Si	0%
	• No	0%

3.3. Corrección de errores

A continuación, se muestran los errores encontrados en la fase de pruebas del prototipo.

3.3.1. Corrección del cliente Android

Mientras los usuarios probaban la aplicación se observó que al momento de seleccionar alguno de los partidos, la aplicación se colgaba al momento de iniciar el fragmento `EventoPartidosFragment`. Se verificó que la excepción lanzada era *IndexOutOfBoundsException*, la misma sucedía debido a que no todos los equipos se encontraban con la cantidad mínima de jugadores registrados. Para solucionar el inconveniente se capturó la excepción mediante un *catch* como se muestra en el código 3.1.

```
catch (IndexOutOfBoundsException e){
    Toast.makeText(getContext(), "JUGADORES NO REGISTRADOS" + e.toString(), Toast.LENGTH_LONG).show();
}
```

Código 3.1. Captura de excepción *`EventoPartidosFragment`*

3.3.2. Corrección del Servidor

En la sección de *Estadísticas* no se visualizaba la tabla de goleadores dentro del fragmento EstadisticasFragment, se verificó que el inconveniente se generaba en la consulta que se realizaba desde el servidor hacia la base de datos, el error que se mostraba desde un navegador era "*The exception message is 'Unable to cast object of type 'System.Int32' to type 'System.String*". Este se generaba debido a que se eliminó una columna de la tabla Jugadores en la base de datos. Se corrigió el inconveniente diferenciando cada una de las tablas dentro del servicio WCF, como se muestra en el código 3.2.

```
177     public List<Jugador> getJugadorEstadistica()
178     {
179         string query = "select * from Posiciones p, Jugador j, Equipo e where p.idEquipo = j.idEquipo and j.idEquipo = e.idEquipo";
180         SqlConnection cnn = new SqlConnection(cadenaConexion);
181         SqlCommand cmm = new SqlCommand(query, cnn);
182         cnn.Open();
183         SqlDataReader leer = cmm.ExecuteReader();
184 
185         while (leer.Read())
186         {
187             listaJugador.Add(new Jugador
188             {
189                 IdJugador = leer.GetInt32(10),
190                 NombreJugador = leer.GetString(12),
191                 ApellidoJugador = leer.GetString(13),
192                 GolesJugador = leer.GetInt32(20),
193                 NombreEquipo = leer.GetString(25)
194 
195             });
196         }
197         return listaJugador;
198     }
```

Código 3.2. Corrección método `getJugadorEstadísticas`

La interfaz corregida se muestra en la figura 3.44.

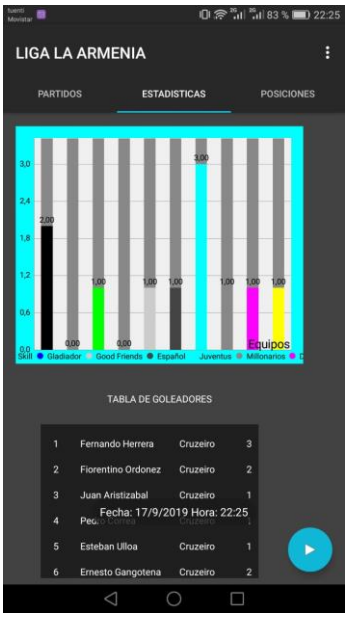


Figura 3.444. Interfaz Estadísticas corregida

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Finalizado el presente Proyecto de Titulación se obtiene como producto final un sistema prototipo de gestión de partidos de fútbol mediante el uso de dispositivos móviles con sistema operativo Android para la Liga Deportiva Barrial “La Armenia”, el mismo que aprovecha el gran despliegue de dispositivos inteligentes que existe en la sociedad actualmente. El sistema prototipo logra mejorar de forma considerable la forma actual de gestión de partidos, solucionando de esta forma el principal problema que es acercarse a las instalaciones de la liga barrial para obtener información de su equipo como los resultados de los partidos, tabla de posiciones y calendario de las próximas fechas.

- Se realizó el estudio de los conceptos básicos para el desarrollo del presente trabajo de titulación como la metodología ágil Kanban, WCF(Windows Communication Foundation), REST (Respresentational State Transfer), Características del Kit de Desarrollo de Software (SDK), temas que facilitaron el entendimiento y desarrollo del presente trabajo de titulación.

- El presente sistema prototipo se desarrolló en base a la metodología ágil Kanban la misma que permitió organizar las tareas a realizar, logrando optimizar en gran medida el tiempo para la realización del sistema prototipo. La metodología Kanban permitió organizar los métodos más complejos del sistema como fueron: la codificación del cronómetro cuando el usuario ingrese a la aplicación y seleccione el partido deseado debe indicar el tiempo transcurrido de un partido y el cálculo de la tabla de posiciones teniendo en cuenta los puntos acumulados y el gol diferencia que tiene cada uno de los equipos.

- Las encuestas realizadas permitieron obtener varios requerimientos de usuario los mismos que permitieron obtener las necesidades que tiene el usuario en cuanto a la obtención de información y gestión de partidos de la liga barrial. Estos requerimientos facilitan el diseño de los diagramas UML (Lenguaje Unificado de Modelado) ya que para su elaboración se parte de la información obtenida por parte del cliente. Los requerimientos de usuario también son utilizados para la elaboración de la interfaz gráfica ya que cada cliente que necesite un mismo producto puede tener requerimientos diferentes en cuanto al diseño de la interfaz gráfica de su aplicación.

- Se hizo uso de la tecnología Volley (Librería de Android que permite realizar peticiones HTTP) por su fácil uso debido a las librerías que utiliza. Sin embargo al utilizar la clase JsonObjectRequest que envía un objeto JSON hacia el servidor no permitia enviarlos mediante el uso de la clase HashMap por lo que se hizo uso de la clase StringRequest que envía datos en forma de string para lo cual se tuvo que pasar de un objeto JSON a un string para el envío de la información para que esta sea entendida por parte del servidor.

- Para mejorar el rendimiento de la aplicación se creó una clase VolleySingleton que permite crear una única instancia del tipo RequestQueue para el envío de peticiones desde el cliente Android hacia el servicio WCF evitando consumir recursos innecesarios mediante la creación de una instancia por cada petición realizada hacia el servidor.

- Para una mejor visualización y desarrollo del prototipo se hizo uso del componente RecyclerView que permite personalizar los elementos que necesitan ser mostrados de forma recurrente realizando un layout como base permitiendo tener varios componentes dentro de uno solo y se van a ir repitiendo según la información y cantidad de ítems que se tenga dentro de una lista.

- Para reducir el uso de recursos en la base de datos, las imágenes correspondientes a las fotos de los jugadores y a los escudos de los equipos son convertidas del formato .jpg a base64, este último permite almacenar la imagen ocupando menos espacio en la base de datos, además lo acepta como un atributo del tipo string. Para mostrar la imagen al usuario se utiliza un ImageView para ello se convierte la imagen del formato base64 al Bitmap.

- Para publicar el servicio WCF se tuvo dos opciones: los servidores de AWS(Amazon Web Services) y Microsoft Azure, en este último se subió el servicio WCF pero al momento de querer obtener información en un endpoint solicitaba realizar configuraciones extra, mientras que en AWS solamente se subió el servicio WCF y todos los métodos en sus respectivos endpoint funcionaron correctamente, simulando un servidor subido en IIS(Internet Information Services), escogiendo los servidores de Amazon por este detalle y por su gratuidad de uso en el primer año.

- Las pruebas que se realizaron a la aplicación indican que los requerimientos obtenidos en el capítulo II se cumplen correctamente, conclusión que es demostrada mediante las encuestas de validación que se encuentran en el capítulo III del presente trabajo de titulación.

4.2. Recomendaciones

- Para el envío de los datos de inicio de sesión de los perfiles Administrador y Vocal se hizo uso del protocolo de cifrado AES brindando un nivel medio de seguridad para la aplicación, para el desarrollo de aplicaciones que requieran un nivel más alto en seguridad como son las de instituciones financieras se recomienda utilizar soluciones de seguridad que permitan monitorear constantemente el flujo del tráfico como por ejemplo Nagios que es una herramienta gratuita generada en el año de 1996 en EEUU.

- Las imágenes que representan los escudos de los equipos y las fotos de los jugadores se las recibieron mediante *strings.* Y para mejorar el rendimiento de la aplicación se recomienda utilizar las direcciones URL donde se encuentran localizadas las imágenes.

- A la aplicación se puede agregar nuevas funcionalidades como aumentar la cantidad de ligas de las cuales se puede gestionar sus partidos, implementar un chat que permita interactuar entre los usuarios que forman parte de un mismo equipo, agregar un módulo de vocalía que permita registrar a los jugadores que van a jugar un partido mediante el uso de huella dactilar, implementar un módulo de finanzas que permita administrar los ingresos y egresos de la liga barrial.

- Para desarrollar un software en cualquier lenguaje de programación se recomienda utilizar una herramienta de desarrollo de software como Kanban que fue implementada en este trabajo de titulación, ya que permite al desarrollador tener un control ordenado de las tareas que se deben realizar para conseguir terminar el producto en un tiempo determinado.

- Para el envío de datos desde el cliente hacia el servidor se recomienda utilizar el método post de HTTP encapsulado en el formato JSON que permite mayor cantidad de caracteres en lugar de utilizar el método get que envía los datos escritos en la URI disminuyendo el número de caracteres que se puede enviar.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] Deporte más practicado. [En línea]. https://federadosconeldeporte.com/deportes-mas-practicados-mundo/

[2] Aplicaciones de Fútbol. [En línea]. https://andro4all.com/2017/05/mejores-apps-resultados-futbol-android

[3] S. Klein, Professional WCF Programming .NET Development with the Windows Communication Foundation. Wiley Publishing, Inc, pp. 10 – 15, 2007.

[4] «What Is Windows Communication Foundation», 29-mar-2017. [En línea]. Disponible en: https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf. [Último acceso: 15/04/2019].

[5] J. Flanders, Restfull .NET. Sebastopol, CA.: O'Reilly Media Inc. pp. 5 – 13. 2009.

[6] Características REST. [En línea]. Disponible en: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos. [Último acceso: 17/04/2019].

[7] J. F. DiMarzio, Beginning Android® Programming with Android Studio, Primera. Indianápolis: John Wiley & Sons, Inc., 2017.

[8] Arquitectura de Android. [En línea]. Disponible en: https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android.

[9] Arquitectura de Android. [En línea]. Disponible en: https://developer.android.com/guide/platform/.

[10] Kit de desarrollo de Android. [En línea]. Disponible en: https://developer.android.com/studio/releases/sdk-tools.

[11] D. Kroenke, Procesamiento de Bases de Datos, Fundamentos, diseño e implementación, Octava Edición. Pearson Prentice Hall, pp. 3 – 25. 2003.

[12] C. Coronel y S. Morris, Database Systems: Design, Implementation, and Management., 12th ed. Boston: Cengage Learning, 2017, pp 68 – 75.

[13] H. Kniberg and M. Skarin, Kanban and SCRUM-making the most of both: Lulu. com, 2010.

[14] Tablero Kanban. [En línea]. Disponible en: https://kanbanize.com/es/recursos-de-kanban/software-kanban/ejemplos-de-tableros-kanban/

[15] Microsoft Visual Studio. [En línea]. Disponible en: https://www.genbeta.com/herramientas/microsoft-apuesta-por-el-multiplataforma-visual-studio-2017-llega-para-windows-macos-y-linux

[16] Arquitectura Cliente – Servidor. [En línea]. Disponible en: https://sites.google.com/site/dap4gen/home/estructura-clienteservidor.

[17] M. Arias, La ingeniería de requerimientos y su importancia en el desarrollo de software, vol VI, InterSedes: Revista de las Sedes Regionales, pp. 2 -3. 2005.

[18] Requerimientos no funcionales, [En línea]. Disponible en: http://www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html.

[19] L. Craig, UML y Patrones, Una introducción al análisis y diseño orientado a objetos y al proceso unificado, Segunda Edición, Pearson Prentice Hall, pp 3 – 10

[20] R. Barker, El modelo entidad-relación CASE METHOD, Primera edición, Addison-Wesley / Díaz De Santos, pp 2 – 34.

[21] Diagrama de clases. [En línea]. Disponible en: https://diagramasuml.com/diagrama-de-clases/ [10] H. Kniberg and M. Skarin, Kanban and SCRUM-making the most of both: Lulu. com, 2010.

ANEXOS

ANEXO A. Encuesta de requerimientos de usuario.

ANEXO B. Script de la creación de la base de datos.

ANEXO C. Código fuente del servicio WCF.

ANEXO D. Código fuente del cliente Android.

ANEXO E. Encuesta de validación.

Todos los anexos se encuentran en el CD adjunto a este documento.

ORDEN DE EMPASTADO