Facultad 2

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Sistema informático de registro y control temporal de acciones de un partido de balonmano por los árbitros de mesa

Autor: Eileen Francis Herrera Sánchez

Tutores: Ing. Vladimir Milián Núñez

Ing. Roberto Antonio Infante Milanés

Cotutor: Msc. Lic. Aloy Machado Sánchez

La Habana, 2018

"Las oportunidades grandes nacen de haber sabido aprovechar las pequeñas"

Bill Gates

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora del trabajo de diploma titulado: “Sistema informático de registro y control temporal de acciones de un partido de balonmano por los árbitros de mesa” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Autor:_____

Eileen Francis Herrera Sánchez

Tutores:

_____ _____

Ing. Vladimir Milián Núñez Ing. Roberto Antonio Infante Milanés

Cotutor: _____

Msc. Lic. Aloy Machado Sánchez

DEDICATORIA

El presente trabajo de Diploma va dedicado a mi Madre, por ser la persona más importante en mi vida.

AGRADECIMIENTOS

A mis padres, en especial a mi madre, por estar siempre a mi lado apoyándome, por tanto amor y sacrificio, por ser la mejor madre del mundo.

A mi novio Dario por estar conmigo en los buenos y malos momentos, sobre todo en estos momentos de estrés, gracias amor por ayudarme, apoyarme y quererme tanto. Te amo.

A mi tía Horten, a mi primo Jose Luis, a mi abuela Hortensia, a mi abuela Maria, gracias por su cariño y apoyo.

A mis suegros, abuelos suegros y cuñados, gracias por tantas alegrías y amor brindado.

A Joel por fastidiarme tanto y estar conmigo todo el tiempo, te quiero mucho.

A Devorat por su amistad en estos años de universidad, por ser mi confidente, mi amiga.

A Jessica, por estar todo el tiempo a mi lado, por ayudarme tanto con la tesis, por contenerme en los momentos tristes, por ser mi amiga y mi cómplice.

A mis compañeras de apartamento, gracias por tantos momentos, buenos, malos y regulares, son momentos que siempre recordaré.

A mis amistades del viejo grupo 2301, Emilio, Ernesto, Javier.

A mis compañeros de aula, gracias por su apoyo, siempre me acordaré de ustedes.

A Zoila, Roberto, Vladimir y Aloy, muchas gracias por su apoyo y por su tiempo, he aprendido mucho de ustedes.

A todos los que me han apoyado en este largo camino, muchas gracias.

RESUMEN

Los avances tecnológicos influyen cada vez más en los métodos de adquisición del conocimiento humano. Dentro del balonmano y como medio de control y evaluación del rendimiento, se utilizan con mayor frecuencia herramientas tecnológicas para el análisis del juego y de los jugadores.

El siguiente trabajo describe el desarrollo de una aplicación para dispositivos móviles con sistema operativo Android llamada TableStatisticsHB, que permite el registro y control de las acciones de un partido de balonmano recogidas por los árbitros de mesa. Tiene como finalidad registrar todas las acciones que se generan durante un partido de balonmano, siguiendo el formato de la planilla oficial de balonmano utilizada por los árbitros de mesa. Permite además gestionar partidos, equipos, jugadores y torneos.

Para el desarrollo de la aplicación se utilizó la metodología AUP-UCI. Se emplearon diversas tecnologías, entre ellas: el lenguaje de programación Java, el sistema gestor de bases de datos SQlite y el entorno de desarrollo AndroidStudio para la implementación de las funcionalidades. Se obtuvo como resultado una aplicación que agiliza el proceso de recogida de información de los partidos de balonmano en tiempo real, además calcula indicadores necesarios que facilitan el trabajo de los árbitros de mesa.

Palabras clave: Android, equipos de balonmano, acciones del juego, indicadores y dispositivos móviles

ABSTRACT

Technological advances are influenced the human's knowledge's acquisitions methods. In handball as high performance evaluation and control method, are using more frequently technological tools for game and players performance analysis.

In this diploma thesis, the development of TableStatisticsHB, on Android app that allows to referees annotate and control the main actions in a handball game, are describe. This apps allows annotate all main actions doing by players in a handball game, fallow the structure of the official annotate template, used by table referees. Also, is allowed management games, teams, players and tournaments.

To guide a development, the AUP-UCI methodology is used. The technologies used are: Java as programming language, SQlite as database management, and Android Studio as Integrated Development Environment. The final result is on Android app that agilize the real time information annotate process in a handball game, and also, calculate the main necessary metrics used by table referees.

Key words: Android, handball teams, game actions, metrics and mobile devices.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las Tecnologías de la Información y la Comunicación (TIC) se han convertido en herramientas fundamentales para el desarrollo de la actividad humana en todas las esferas de la sociedad. Una de ellas son los dispositivos móviles inteligentes, cuya utilización ha ido en aumento durante los últimos años, debido principalmente a una mayor apertura de los usuarios, a las redes sociales e Internet. Sin embargo, estos no han sido los únicos factores que han influido en este aumento, ya que en buena parte se ha debido a la necesidad de la informatización de procesos esenciales en el ámbito profesional y social.

Estos dispositivos son herramientas que funcionan gracias a un Sistema Operativo (SO) determinado, cuyas particularidades y niveles de complejidad cada día son mayores, haciendo posible administrar diferentes tipos de aplicaciones con avanzadas capacidades.

Entre los sistemas operativos vinculados a la tecnología móvil destacan Android, IOs y Windows Phone. Android es un sistema gratuito y multiplataforma, entre sus principales características están que su núcleo está basado en el kernel de Linux lo que representa que sea de código abierto, utiliza SQLite para el almacenamiento de datos y soporta el lenguaje de programación Java. Por estas características Android es el más utilizado en la actualidad. (Malave, 2011)

Esto quedó demostrado en un estudio realizado por la International Data Corporation (IDC) en agosto de 2017, como parte del continuo análisis del mercado asociado a las tecnologías móviles, se concluyó que un 85% de usuarios hacen uso de Android, 14,7% utilizan IOs, un 0,1% usan Windows Phone y un 0,1% utilizan otro sistema operativo. (International Data Corporation. IDC, 2017)

Una de las esferas donde destaca actualmente el empleo de las TICs, es el deporte. Cada día resulta más frecuente la utilización de aplicaciones informáticas en la investigación del comportamiento humano, lo que permite reunir datos más complejos y en tiempo real de un evento deportivo. (García, 2016)

El balonmano moderno es uno de los deportes donde la aplicación de los avances científicos-tecnológicos facilita la observación y evaluación sistemática de las acciones táctico-ofensivas tanto individual como colectiva, además de la información de las acciones realizadas tanto durante los entrenamientos como en los diferentes torneos.

En un juego oficial, los árbitros de mesa llevan el control de los equipos y de los jugadores en el terreno. Durante el partido, deben controlar las siguientes acciones:

- Cuando un jugador anota un gol, este se refleja en el resultado del jugador y del equipo para el cual juega.

- Cuando un jugador es amonestado (tarjeta amarilla), este se le anota al jugador (si es la segunda tarjeta personal, el jugador debe ser expulsado por dos minutos) y al equipo (a partir de la tercera tarjeta colectiva se penaliza con expulsión por dos minutos)
- Cuando un jugador es expulsado, se debe llevar el control del tiempo de la expulsión (dos minutos), y se anota la expulsión. Si es la tercera expulsión se le aplica tarjeta roja: expulsión del terreno.
- Se debe controlar la duración de los tiempos técnicos pedidos por los entrenadores (un minuto para cada equipo en cada mitad), duración del partido, los tiempos pedidos por los árbitros y los tiempos de las expulsiones.

Estas acciones se plasman en una planilla de papel, de forma manual, que luego se utiliza para la información general del juego (resultado, máximos goleadores) y para el control estadístico del torneo. Durante el proceso de llenado de la planilla, pueden cometerse errores tipográficos, omisión y/o cambio de nombre y/o número de algún jugador, lo que conlleva a que no estén actualizadas las estadísticas reales de los mismos. Por otro lado, estas planillas pueden deteriorarse o romperse, conllevando a la correspondiente pérdida de la información contenida en las mismas.

Teniendo en cuenta lo planteado anteriormente se define como **problema a resolver:** ¿cómo facilitar a los árbitros de mesa la recogida y resumen de las acciones de un partido de balonmano?

Se identifica como **objeto de estudio** para esta investigación: la gestión de acciones en un partido de balonmano y como **campo de acción** el proceso de gestión de información de acciones en los partidos de balonmano por los árbitros de mesa mediante dispositivos móviles.

Para darle solución al problema planteado anteriormente el **objetivo general** del presente trabajo es: desarrollar un sistema informático de registro y control temporal de acciones de un partido de balonmano, para los árbitros de mesa.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas de investigación:**

1. Análisis de la bibliografía técnica del balonmano para conocer las reglas del deporte y de las competiciones.
2. Análisis de aplicaciones informáticas existentes, destinadas a la recogida de información en los partidos de balonmano, para evaluar la factibilidad del desarrollo del sistema.
3. Análisis de las diferentes herramientas, tecnologías y metodología de desarrollo de software existentes, para conocer las ventajas que brindan y realizar la selección de las más convenientes.
4. Realización de entrevistas con los clientes que desean informatizar el proceso de recogida de acciones significativas en los juegos de balonmano, para identificar los requerimientos -del sistema.
5. Implementación del sistema propuesto en correspondencia con las fases de la metodología seleccionada, para dar cumplimiento a los requisitos definidos.

6. Ejecución de pruebas de software a las funcionalidades implementadas del sistema, para comprobar el correcto funcionamiento del mismo.

Para llevar a cabo el desarrollo de la investigación se utilizan los siguientes **métodos científicos:**

Métodos teóricos

Análisis-Síntesis: se hace uso de este método para enmarcar los elementos teóricos de la investigación facilitando el análisis y comprensión de la documentación consultada. Contribuyendo a realizar comparaciones entre disímiles aplicaciones *Android*, extraer y precisar los elementos y características fundamentales del diseño de la propuesta de solución y establecer las conclusiones de la investigación.

Modelación: utilizado para modelar la arquitectura, crear los artefactos, diagramas y modelos a utilizar en el desarrollo de la aplicación.

Métodos empíricos

Entrevista (de forma no estructurada): se realizaron entrevistas no estructuras a entrenadores de equipos de balonmano para identificar las principales funcionalidades a contener en la propuesta de solución, así como la información que se debe gestionar.

El trabajo de diploma se divide en 3 capítulos, los cuales estarán estructurados de la siguiente forma:

Capítulo1. Fundamentos teóricos de la gestión de información de acciones en el balonmano

Se describen los conceptos asociados al objeto de estudio y se analizan soluciones existentes a nivel nacional e internacional, vinculadas a la recogida de acciones en un partido de balonmano. Se fundamenta el uso de las herramientas, tecnologías, metodología de desarrollo de software y lenguaje de programación que se utilizan en la solución.

Capítulo2. Sistema informático de registro y control temporal de acciones de un partido de balonmano por los árbitros de mesa

Se elabora un modelo de dominio donde se analizan las entidades y conceptos presentes en el contexto donde se desarrolla la propuesta de solución y posteriormente se describen los requisitos funcionales y no funcionales. Se definen los casos de uso del sistema, así como la arquitectura de software y patrón arquitectónico utilizados en la solución. Además, se describen los patrones de diseño, los estándares de codificación y el modelo físico de la base de datos de la aplicación.

Capítulo 3. Implementación y validación del sistema

A partir de los resultados del diseño se explica la implementación de la solución. Como parte de la implementación se muestra el diagrama de componentes, el cual representa la organización y las dependencias entre los componentes del sistema. Se muestra el diagrama de despliegue y el diagrama

de paquetes que permite visualizar como está dividido el sistema en agrupaciones lógicas. Además, se especifican las pruebas ejecutadas a la aplicación con el objetivo de verificar su correspondencia con los requerimientos definidos.

CAPÍTULO 1. FUNDAMENTOS TEÓRICOS DE LA GESTIÓN DE INFORMACIÓN DE ACCIONES EN EL BALONMANO

En este capítulo se exponen elementos asociados al objeto de estudio de la investigación. Se caracterizan soluciones nacionales e internacionales, destinadas a la recogida de acciones significativas en un partido de balonmano. Se describen las tecnologías y herramientas que apoyan el diseño e implementación del sistema, así como la metodología de desarrollo de software a utilizar.

1.1 Conceptos asociados al objeto de estudio

El balonmano es un deporte de pelota en el que se enfrentan dos equipos, compuesto de catorce jugadores cada uno, siete sobre el terreno de juego y siete en el banquillo. Los equipos tienen por objetivo enviar el balón a la portería contraria. En un partido de balonmano además de los jugadores intervienen cuatro árbitros, dos en el terreno, y dos en la mesa, uno tiene función de cronometrador y el otro tiene función de anotador, además tienen la responsabilidad de registrar las acciones que se generan durante el partido. (Federación Internacional de Balonmano, 2016)

Una acción de interés para los árbitros de mesa en el balonmano, es el resultado de un movimiento realizado por un jugador ya sea en ataque, defensa o contraataque, puede ser un gol, una falta, un lanzamiento fallido o una expulsión. Estas acciones se registran en el acta del partido (planilla oficial del balonmano), que es redactada durante el partido por los árbitros de mesa. Al finalizar el juego, con el objetivo de llevar el control de los equipos y partidos, se hace un análisis de dichas acciones a partir del cálculo de los siguientes indicadores: (Federación Internacional de Balonmano, 2016)

1. Total de Expulsiones por equipo (TE)
2. Total de Amarillas por equipo (TA)
3. Partidos Jugados (PJ)
4. Partidos Ganados (PG)
5. Partidos Empatados (PE)
6. Partidos Perdidos (PP)
7. Goles totales del equipo (G)
8. Lanzamientos Fallidos totales por equipo (LF)
9. Por ciento de efectividad de los equipos (PE), $PE = \frac{G}{LF+G} * 100$

El registro y control temporal de las acciones le permite a los árbitros arribar a conclusiones sobre el comportamiento de los equipos en los partidos y por ende en los torneos, permite por ejemplo determinar el equipo más goleador, el que más faltas cometió y el lugar que dicho equipo ocupa en un torneo.

1.2 Análisis de otras soluciones existentes

En el mundo existen diversas aplicaciones para el balonmano, algunas enfocadas a la recogida de datos de los jugadores y equipos por parte de los entrenadores, y otras más sencillas como son: las pizarras tácticas y los marcadores. Todas estas soluciones facilitan el desarrollo del juego en tiempo real. No se encontró ninguna aplicación que en su totalidad gestione los datos que se recogen en la mesa por parte de los árbitros. Se hizo necesario el estudio de otras soluciones existentes, para encontrar elementos significativos que contribuyen al desarrollo del sistema propuesto.

1.2.1 Aplicaciones Internacionales

Para el desarrollo de la propuesta de solución, fue necesario el análisis de aplicaciones internacionales utilizadas en el balonmano, a continuación, se presentan las principales características de cada una de ellas.

Estadísticas balonmano

Es una aplicación con sistema operativo Android que permite crear estadísticas profesionales de los equipos en un partido de balonmano, acceder a ellas en tiempo real durante el partido o como evaluación de toda la temporada. Ideal para que entrenadores y gestores identifiquen fortalezas y debilidades de cada jugador y equipo, y mejoren sus habilidades. (Google Play, 2017)

Debido a la rapidez de los partidos, esta aplicación presta especial atención a una introducción de datos rápida e intuitiva, cada acción puede introducirse en dos toques, escogiendo de listas concisas. Permite añadir equipos, jugadores y varios tipos de torneos. En preferencias es posible elegir las acciones que tienen que introducirse y considerarse en las estadísticas y el usuario puede definir hasta 10 acciones. (Google Play, 2017)

Brinda estadísticas completas para cada partido disponibles durante el mismo. Para una vista completa de toda la temporada permite generar estadísticas del torneo que contienen a todos los jugadores que jugaron al menos un partido, el número de partidos, las acciones y los tiros y niveles de consolidación. (Google Play, 2017)

Esta aplicación contribuye al desarrollo de la solución propuesta, tiene elementos importantes como la gestión de los jugadores y de los equipos. Además, tiene una interfaz amigable, intuitiva y de fácil uso, que sirve como guía para la implementación de la aplicación en cuestión. A su vez “Estadísticas Balonmano” tiene entre sus limitantes que no tiene un marcador para ver la puntuación durante un partido, no es de código abierto y para su utilización requiere de una licencia pagada.

HB-All Handball Statistics

Esta aplicación permite registrar y analizar un partido de balonmano sobre la marcha. Trabaja bajo el sistema operativo Android.

Presenta las siguientes características: (Google Play, 2017)

- Registra los datos de un partido de balonmano (equipos, jugadores, la posición meta y posición en el campo
- Permite subir y publicar los registros de los partidos a un servidor
- Permite descargar un archivo de Excel creado a partir del partido cargado en el servidor
- Imprime un informe de síntesis a partir del partido cargado en el servidor
- Tiene la opción de etiquetar un partido con un identificador único. Este identificador se puede consultar en el servidor (por ejemplo, para obtener todos los resultados de su club)
- Envía un breve resumen a través de las intenciones de Android, por ejemplo, a una dirección de correo o Google+

Esta aplicación presenta elementos significativos que sirven de apoyo para el desarrollo de la propuesta de solución, como es el caso del registro de datos de un partido y la impresión de un informe resumen. Por otro lado tiene las siguientes limitantes: no permite la recogida de datos en tiempo real: anotación de acciones de los jugadores durante el partido. No es un software gratuito ni de código abierto.

Scoreboard handball

Aplicación con sistema operativo Android basada en un marcador de balonmano que muestra el nombre de los equipos, el puntaje, el tiempo y el período del partido. Tiene opciones para modificar los minutos por períodos y el número de períodos regulares del partido, y para habilitar y deshabilitar el tiempo de exclusión y utilizarlo en el marcador. (Google Play, 2017)

Esta aplicación tiene las siguientes características: (Google Play, 2017)

- Se puede utilizar en teléfonos y tabletas
- Es movible a la tarjeta SD
- Tiene precisión de centésimas en el último minuto del juego
- Permite la edición del nombre de los equipos
- Permite la cancelación de los últimos 20 cambios
- Tiene sonido al finalizar el tiempo
- Memoriza la información al salir de la aplicación
- No activa el salvapantallas durante el juego
- Es compatible con el sistema operativo Android desde la versión 1.6

Esta aplicación propone ideas a tener en cuenta en el diseño del marcador que se necesita para la solución propuesta. Pero se limita a las necesidades del cliente ya que solo registra la puntuación del partido.

1.2.2 Aplicaciones nacionales

En el estudio de aplicaciones para el balonmano en Cuba, se encontraron aplicaciones que gestionan información de los equipos, jugadores y partidos. A continuación se analizan y presentan características de cada una de ellas, con el objetivo de encontrar elementos que contribuyen en el desarrollo de la propuesta de solución.

Sistema de Anotación del Balonmano

El Sistema de Anotación del Balonmano permite llevar el control de un conjunto de datos que se recogen durante los partidos de este deporte, con el objetivo de ser analizados y que permitan tomar decisiones. Permite llevar el control de un partido en tiempo real, brindando al anotador tablas en las cuáles puede recoger los datos predeterminados en ellas de forma rápida y sencilla teniendo apoyo del mouse y del teclado, además, permite la gestión de partidos, campeonatos, equipos y jugadores. (Obregón, y otros, 2010)

Esta aplicación permite: (Obregón, y otros, 2010)

- Consultar Datos de un Juego
- Consultar Datos de un Equipo en un Campeonato
- Crear Campeonato
- Insertar Equipos
- Iniciar juego

Esta aplicación a pesar de llevar el registro de una gran cantidad de datos de los jugadores, tiene como principal limitante que es una aplicación de escritorio, por lo tanto no funciona con el sistema operativo Android, y no se puede utilizar en dispositivos móviles y tabletas.

AndroHB

Es una aplicación para móviles inteligentes con sistema operativo Android desarrollada en la Universidad de las Ciencias Informáticas (UCI) en el 2017, la cual cuenta con un cronómetro para llevar el tiempo del partido, con tres cronómetros para las amonestaciones, una pizarra táctica y permite exportar archivos con formato de documento portátil (Portable Document Format, PDF por sus siglas en inglés) con el registro de los datos recogidos. (Martínez, y otros, 2017)

Mediante la información introducida por el usuario la aplicación calcula y muestra automáticamente una serie de estadísticas facilitando el trabajo en el proceso de recogida de datos y cálculo de las estadísticas sirviendo así de apoyo para la toma de decisiones. (Martínez, y otros, 2017)

Presenta las siguientes funcionalidades: (Martínez, y otros, 2017)

1. Gestionar torneo
2. Gestionar jugadores
3. Gestionar equipos
4. Crear nuevo partido

Esta aplicación no permite registrar la información que manejan los árbitros de mesa, debido a que presenta las siguientes limitaciones:

- Solo permite registrar las acciones generadas por un equipo en un partido.
- Calcula estadísticas a partir de indicadores que son de interés para los entrenadores, no para los árbitros.
- La información que almacena no tiene la estructura de la planilla oficial del balonmano que utilizan los árbitros de mesa.
- Está centrada en los jugadores, no en los equipos.
- Solo permite controlar el tiempo de amonestación de tres jugadores.

1.2.3 Resultado del estudio de las herramientas

Las herramientas analizadas anteriormente permiten tener una visión de cómo llevar a cabo la implementación de la propuesta de solución. La mayoría de estas aplicaciones no son gratuitas ni de código abierto, lo que no hace posible la adquisición de su código fuente, y que no pueda ser reutilizado. Además, no cuentan con una funcionalidad que registre los datos que controlan los árbitros de mesa. A partir de los elementos antes señalados se concluyó que ninguna de las herramientas analizadas, constituye una solución al problema de la investigación. Por tanto, se decide implementar un sistema que les permita a los árbitros de mesa, la recogida de los datos más importantes en los partidos de balonmano.

1.3 Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un software de calidad. Indican paso a paso todas las actividades a realizar para lograr el producto informático deseado, qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Por lo tanto, elegir la mejor metodología para un equipo, en un determinado proyecto es un elemento trascendental. (Gutierrez, 2011)

Existen dos grupos de metodologías, las tradicionales y las ágiles. Teniendo en cuenta la filosofía de desarrollo de las metodologías, aquellas con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, son las tradicionales o pesadas. (Gutierrez, 2011)

Estas metodologías se centran especialmente en el control del proceso de desarrollo de software, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar. (Figueroa, y otros, 2014)

Ante las limitaciones de las tradicionales surgen las metodologías ágiles, diseñadas para equipos de desarrollo pequeños, con plazos de tiempo reducidos y requisitos volátiles. En esta metodología el cliente forma parte del equipo, requiere de pocos artefactos y hace menos énfasis en la arquitectura del proyecto. (Letelier, y otros, 2013)

Para la selección de la metodología existen varios métodos: el Gráfico propuesto por Boehm y Turner, Método de Expertos y la Matriz de Evaluación de Metodología. Se elige el método propuesto por Boehm y Turner debido a las ventajas que este posee. Es un modelo de fácil comprensión capaz de evaluar cuantificar e identificar 5 variables críticas a la hora de definir la metodología a utilizar. (Boeras, y otros, 2012)

Criterios que propone el gráfico de Bohem y Turner: (Boeras, y otros, 2012)

1. Personal: representa la proporción del personal con experiencia alta, media y baja
2. Criticidad: se utiliza para evaluar la naturaleza del daño ocasionado por defectos que no hayan sido detectados al producto, su evaluación puede ser cualitativa
3. Cultura: las organizaciones y las personas que relaciona el proyecto pueden depender de la confianza o de la relación contractual. Esto refleja el nivel de ceremonia necesario y aceptado: documentación, control, formalismo en las comunicaciones
4. Tamaño: representa la cantidad de personas involucradas en el proyecto
5. Dinamismo: representa la rapidez con la que pueden estar cambiando los requerimientos del proyecto

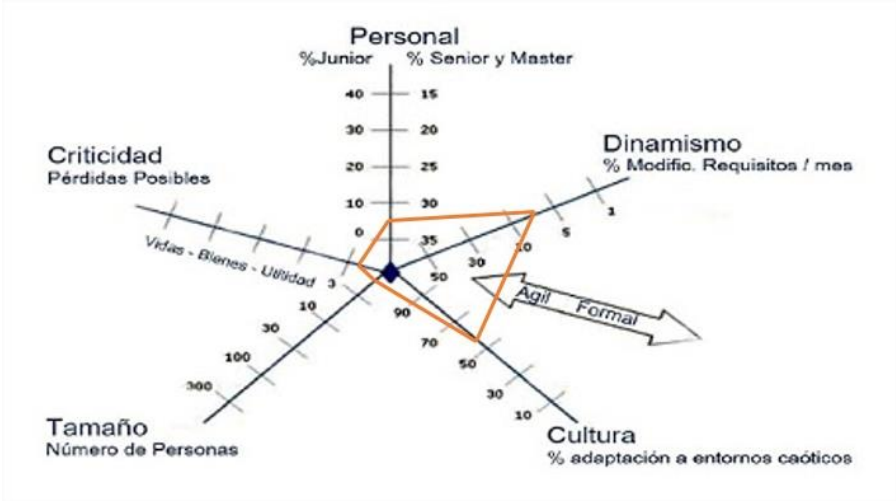


Figura 1. Aplicación del método de Boehm y Turner

Analizando cada uno de estos cinco criterios fundamentales en correspondencia con las características del equipo, se decide utilizar una metodología ágil.

Dentro de las metodologías ágiles se selecciona Proceso Unificado Ágil (Agile Unified Process, AUP en inglés), fue creada por Scott Ambler en el 2002. Es una versión simplificada de Rational Unified Process (RUP) desarrollado por IBM. Describe una manera simple de entender el desarrollo de aplicaciones de negocio usando técnicas ágiles y conceptos heredados de RUP. (Rodríguez, 2015)

AUP es un marco de desarrollo interactivo e incremental que aplica técnicas ágiles, incluyendo Desarrollo Dirigido por Pruebas, Modelado Ágil, Gestión de Cambios Ágil, y Refactorización de Base de Datos para mejorar la productividad. (Rodríguez, 2015)

Esta metodología posee 4 fases igual que RUP: (Rodríguez, 2015)

1. Inicio: identificar el alcance inicial del proyecto, una arquitectura recomendable para el sistema, obtener fondos y la aceptación por parte de las personas involucradas
2. Elaboración: el objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura
3. Construcción: durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
4. Transición: validación e implantación del sistema

Para el desarrollo de la aplicación se seleccionó como metodología de desarrollo AUP en una variante desarrollada en la Universidad de las Ciencias Informáticas (UCI), nombrada AUP-UCI. Esta metodología fue ideada para tomar aspectos importantes de varias metodologías ágiles con las que se trabaja en la universidad y unificarlos en esta debido a que en ocasiones los proyectos no convergen en una sola metodología de desarrollo según lo descrito por Sánchez en el 2015.

Se selecciona esta metodología ya que es utilizada actualmente en la UCI. Además propone una gestión de cambios ágil, permitiendo adaptar la aplicación a nuevas condiciones en dependencia de las necesidades del cliente. Otro punto a su favor es que la propuesta de solución va a ser desarrollada por un solo programador de mediana experiencia.

En AUP-UCI a partir de que el modelado de negocio propone tres variantes a utilizar en los proyectos (Casos de Uso del Negocio, Descripción de Procesos de Negocio, Modelo Conceptual) y existen tres formas de encapsular los requisitos (Casos de Uso del Sistema, Historias de Usuario, Descripción de Requisitos por Proceso), surgen cuatro escenarios para modelar el sistema en los proyectos. (Sánchez, 2015)

Después de evaluar el negocio se selecciona el escenario número 2, ya que la aplicación tiene como principal objetivo la gestión y presentación de información. Además no es necesario incluir las responsabilidades de las personas que van a ejecutar las actividades, por tanto solo será necesario modelar exclusivamente los conceptos fundamentales del negocio. (Sánchez, 2015)

1.4 Lenguajes, herramientas y tecnologías

En este epígrafe se describen los lenguajes, herramientas y tecnologías empleadas para la modelación, diseño e implementación de la aplicación.

1.4.1 Lenguajes

Lenguaje de programación

Java es un lenguaje de programación simple, orientado a objetos que surge a finales de 1995. Incluye una combinación de características que lo hace único, y está siendo adoptado por multitud de fabricantes como herramienta básica, para el desarrollo de aplicaciones comerciales de gran repercusión. Una de las características más importantes es que los programas “ejecutables”, creados por el compilador de Java, son independientes de la arquitectura, además se ejecutan indistintamente en una gran variedad de equipos con diferentes microprocesadores y sistemas operativos. (Pavón, 2013)

Se seleccionó Java como lenguaje de programación debido a que es el lenguaje que propone Android para el desarrollo de aplicaciones móviles, se utiliza la versión 1.8.0_77, por la compatibilidad que tiene con el entorno de desarrollo AndroidStudio 2.1.3.

Lenguaje de modelado

Lenguaje Unificado de Modelado (Unified Modeling Language, UML por sus siglas en inglés), es un lenguaje gráfico de modelado que permite visualizar, especificar, construir y documentar un sistema de software. Incluye aspectos conceptuales tales como: procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes

reutilizables. (Sierra, 2013)

En el desarrollo de la aplicación se utiliza UML en su versión 1.0, para representar las diferentes partes de un sistema, los diagramas de casos de uso, de despliegue, de componentes, de colaboración y el resto de los diagramas.

1.4.2 Herramientas y tecnologías

Para el desarrollo de la aplicación es necesario el uso de algunas herramientas y tecnologías, las cuales se describen en el desarrollo de este acápite.

Entorno de desarrollo integrado

Se seleccionó Android Studio como entorno de desarrollo, debido a que es el entorno de desarrollo integrado (Integrated Development Environment, IDE por sus siglas en inglés) oficial de Android, para el desarrollo de aplicaciones móviles, se utiliza en su versión 2.1.3.

Android Studio tiene herramientas especializadas que hacen el proceso más fácil, rápido y sencillo. Posee un editor gráfico que permite generar interfaces sin la necesidad de crear código. Se tiene acceso a herramientas personalizadas y optimizadas para trabajar en Android, de forma que las interfaces son específicamente pensadas para el trabajo en entornos Android. Posee completamiento automático de código y corrección de errores. (Valdés, 2016)

Sistema gestor de bases de datos

SQLite es una librería de código abierto y distribuida bajo dominio público, que implementa un gestor de bases de datos SQL embebido, sin configuración y transaccional. Es caracterizado por mantener el almacenamiento de información persistente de forma segura y sencilla. (Montiel, 2008)

Principales características (Montiel, 2008)

- Con todas sus características habilitadas la librería tiene un tamaño inferior a 250kb. Deshabilitando características opcionales, el tamaño puede estar por debajo de los 180kb. Esto permite utilizarla en dispositivos con poca memoria.
- Requiere muy poco soporte de librerías externas o del sistema operativo. Esto la hace apropiada para usarla en dispositivos pequeños que no son tan completos como una computadora.
- Es un sistema completo de bases de datos que soporta múltiples tablas, índices, triggers y vistas.
- La base de datos se almacena en un único fichero a diferencia de otros sistemas gestores de bases de datos que hacen uso de varios archivos.

Se utiliza SQlite porque es el gestor de bases de datos integrado al entorno de desarrollo AndroidStudio.

Se seleccionó la librería de Java (Object Relational Mapping Lite ORMLite por sus siglas en inglés), que proporciona algunas funcionalidades simples y livianas, facilitando el trabajo con las bases de datos.

Esta librería permite realizar consultas a través de código Java, reduciendo el tiempo de respuestas de dichas consultas.

Principales características de ORMLite: (Android Corporation, 2017)

- Agilidad en el desarrollo software
- Son usados de manera mayoritaria en sistemas de Bases de Datos SQL.
- Mantenimiento y reutilización de código

Herramienta de modelado UML

Visual Paradigm es una herramienta CASE (Ingeniería de Software Asistida por Computación). La misma permite el modelado de diagramas para el desarrollo de programas informáticos. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Fue diseñado para la construcción de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. (Valdés, 2016) Para el modelado de diagramas de la aplicación se utilizó Visual Paradigm en su versión 8.0.

Conclusiones del capítulo

En este capítulo se definieron conceptos fundamentales asociados al objeto de estudio, se especificaron los indicadores a utilizar para llevar el control estadístico de las acciones registradas por los árbitros de mesa generadas durante un partido. Se analizaron las aplicaciones existentes a nivel nacional e internacional y se arribó a la conclusión de que ninguna de estas aplicaciones cumple en su totalidad con las necesidades del cliente, sin embargo, tienen elementos que sirven como base para el desarrollo del nuevo sistema. El estudio de las metodologías permitió seleccionar la más apropiada para el proyecto, según sus características, en este caso AUP-UCI. Se definieron también las herramientas y tecnologías a emplear teniendo en cuenta las necesidades del desarrollo de la propuesta de solución. Se seleccionó Java 1.8.0_77 como lenguaje de programación, Visual Paradigm 8.0 para el modelado UML, Android Studio en su versión 2.1.3 como IDE y SQLite como sistema gestor de bases de datos, con la librería ORMLite para el mapeo de los datos.

CAPÍTULO 2. SISTEMA INFORMÁTICO DE REGISTRO Y CONTROL TEMPORAL DE ACCIONES DE UN PARTIDO DE BALONMANO POR LOS ÁRBITROS DE MESA

El presente capítulo tiene como objetivo describir la solución propuesta. Se presenta el modelo de dominio donde se analizan las entidades y conceptos presentes en el contexto donde se desarrolla la propuesta de solución y se definen las relaciones existentes entre cada uno de estos. Se especifican los requisitos funcionales y no funcionales con los que debe cumplir el sistema, se describe el diagrama de casos de uso del sistema y los actores involucrados. Se define la arquitectura de software y el patrón arquitectónico a utilizar en el desarrollo de la aplicación, además se describen los estándares de codificación y el modelo físico de la base de datos del sistema.

2.1 Propuesta de solución

El balonmano es un deporte muy dinámico, donde el intervalo de tiempo entre la ocurrencia de las acciones es muy corto. A pesar de esto, los árbitros de mesa deben ser capaces de registrar dichas acciones y controlar los tiempos técnicos (duración general del partido, tiempo de las amonestaciones y expulsiones de cada jugador). Estas acciones se plasman en una planilla de papel, que luego es utilizada para la información general del juego y el control estadístico del torneo.

Durante el registro manual del partido, pueden cometerse errores tipográficos, omisión y/o cambio de nombre y/o número de algún jugador, lo que conlleva a que no estén actualizadas las estadísticas reales de los mismos. Por otro lado, estas planillas pueden deteriorarse o romperse, conllevando a la correspondiente pérdida de la información contenida en las mismas.

Con el objetivo de facilitar el trabajo de los árbitros de mesa y agilizar el registro y control de las acciones en los partidos, se propone implementar la aplicación TableStatisticsHB, que permite la recogida en tiempo real, de datos relacionados con las acciones registradas por los árbitros de mesa durante un partido de balonmano, de una manera rápida y sencilla.

TableStatisticsHB es una aplicación que les permite a los árbitros de mesa la recogida en tiempo real, de datos relacionados con las acciones de un partido de balonmano, de una manera rápida y sencilla.

Brinda como funcionalidades: la gestión de los torneos, equipos, jugadores y partidos. La gestión de los partidos posibilita la anotación de las principales acciones como son: goles y faltas, además de llevar el control del tiempo de duración de los partidos y de las expulsiones de los jugadores. A partir de estas acciones se genera un resumen del partido que se muestra al finalizar el mismo, dicho resumen se puede exportar como un archivo con formato PDF. La gestión de torneos, equipos y jugadores se puede hacer tanto durante el partido como después de finalizado, a través del menú de la aplicación, al igual que la consulta del resumen de los juegos.

Al instalar la aplicación en el dispositivo móvil, se crea automáticamente una carpeta con el nombre TableStatisticsHB, que contiene la base de datos de la aplicación (tshb.db), esta carpeta se puede compartir mediante bluetooth o wifi con otros dispositivos, permitiendo de esta forma la replicación de la información contenida en la base de datos.

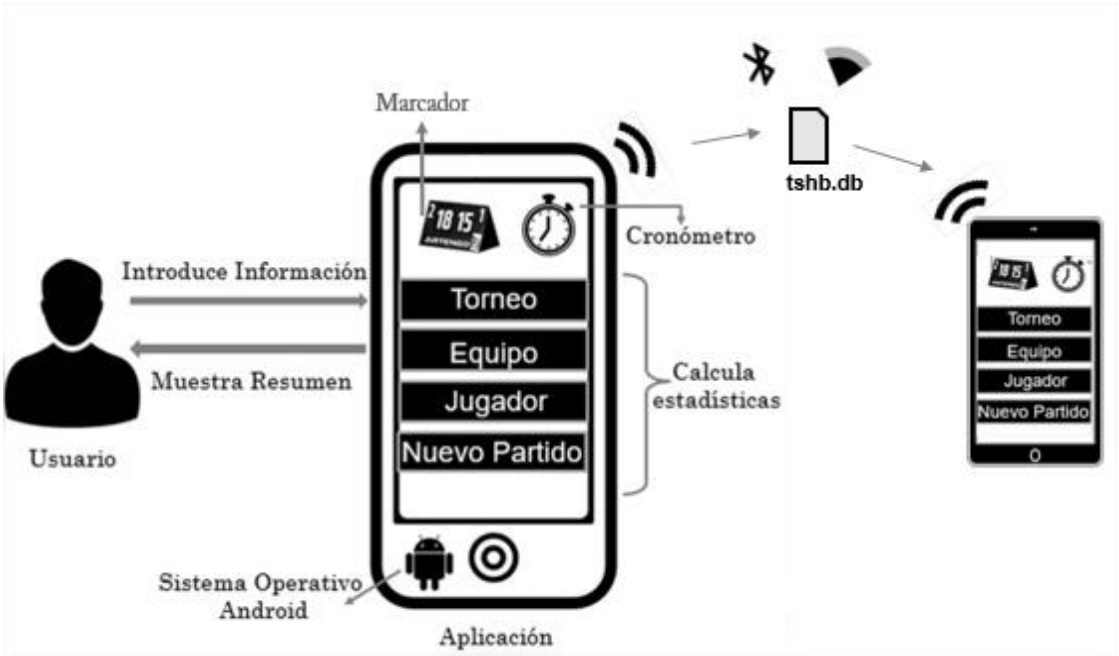


Figura 2. Propuesta de solución

2.2 Modelo conceptual

El modelo de dominio o modelo conceptual muestra (a los moderadores) clases conceptuales significativas en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. (Rio, 2011)

Utilizando la notación UML, un modelo de dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. (Rio, 2011)

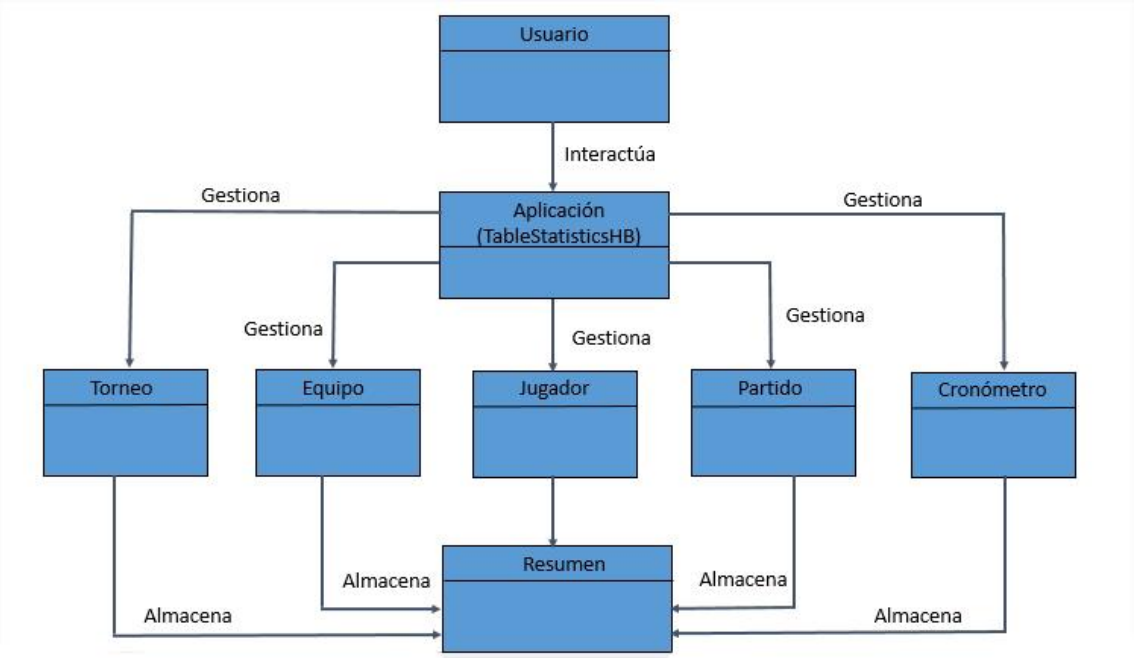


Figura 3. Modelo Conceptual

Descripción del modelo conceptual

Usuario: es la persona que interactúa con la aplicación

Dispositivo Móvil: es el medio donde se encuentra instalada la solución propuesta

Equipo: es el colectivo integrado por varios jugadores

Jugador: es la persona que juega en un partido de balonmano

Partido: forma de competición donde se enfrentan dos equipos

Torneo: hace referencia al evento principal de balonmano, donde intervienen jugadores, equipos y partidos

Resumen: al finalizar el partido la aplicación muestra un resumen de las acciones generadas en el partido, dígase cantidad de goles, cantidad de faltas y el tiempo en el que sucedieron dichas acciones.

Cronómetro: se encarga de llevar el control de los tiempos en la aplicación, ya sea el tiempo de duración de un partido o el tiempo de amonestación de un jugador.

2.3 Requisitos funcionales del sistema

Los requerimientos funcionales describen lo que el sistema o software debe hacer. Estos siempre se refieren a funciones específicas que debe hacer la solución, todo en base a las indagaciones hechas en el proceso de levantamiento de requerimientos y su respectivo análisis en las definiciones de Requerimientos de Negocio y Requerimientos del Sistema. (Martínez, y otros, 2010)

Tabla 1. Listado de requisitos funcionales

Número del requisito	Nombre del requisito	Descripción
RF1	Gestionar torneo	Permite al usuario gestionar la información referente a los torneos
RF1.1	Listar torneo	Lista todos los torneos existentes en la base de datos y los muestra al usuario
RF1.2	Insertar torneo	Permite al usuario añadir a la base de datos un nuevo torneo
RF1.3	Editar torneo	Permite al usuario modificar un torneo seleccionado de la lista
RF1.4	Mostrar detalles torneo	Muestra los resultados de los equipos y de los partidos jugados en el torneo
RF1.5	Eliminar torneo	Permite al usuario eliminar un torneo
RF2	Gestionar jugador	Permite al usuario gestionar la información referente a los jugadores
RF2.1	Listar jugador	Lista todos los jugadores existentes en la base de datos
RF2.2	Insertar jugador	Permite al usuario añadir a la base de datos un nuevo jugador
RF2.3	Editar jugador	Permite al usuario modificar un jugador seleccionado de la lista
RF2.4	Eliminar jugador	Permite al usuario eliminar un jugador

RF3	Gestionar equipo	Permite al usuario gestionar la información referente a los equipos
RF3.1	Listar equipo	Lista todos los equipos existentes en la base de datos
RF3.2	Insertar equipo	Permite al usuario añadir a la base de datos un nuevo equipo
RF3.3	Editar equipo	Permite al usuario modificar un equipo seleccionado de la lista
RF3.4	Eliminar equipo	Permite al usuario eliminar un equipo
RF3.5	Mostrar detalles equipo	Muestra al usuario los jugadores que pertenecen al equipo seleccionado
RF4	Crear nuevo partido	Permite al usuario crear un nuevo partido a partir de un torneo y equipos previamente creados
RF4.1	Mostrar Cronómetro de juego	Genera un cronómetro que lleva el control del tiempo del partido
RF4.2	Mostrar cronómetros de amonestaciones	Genera un cronómetro cada vez que un jugador es amonestado y expulsado del partido por un tiempo
RF4.3	Insertar acción	Permite al usuario insertar una acción significativa del partido (gol o falta)
RF4.4	Registrar Estadísticas	Durante un partido a partir de la información introducida por el usuario, la aplicación registra las estadísticas de

		los equipos que intervienen en el partido
RF4.5	Generar resumen	Al finalizar el partido el sistema genera un resumen con los datos registrados por el usuario durante el partido
RF4.6	Exportar a PDF	Se puede exportar al finalizar el partido un documento con formato PDF, con la planilla que se genera durante el partido

2.4 Requisitos no funcionales del sistema

Los requisitos no funcionales restringen el sistema en desarrollo y el proceso de desarrollo que se debe utilizar. Pueden ser requisitos del producto, organizacionales o externos. A menudo están relacionados con las propiedades emergentes del sistema y, por lo tanto, se aplican al sistema completo. (Sommerville, 2005)

Requisitos no funcionales de TableStatisticsHB

Usabilidad

- La aplicación debe poseer una interfaz amigable e intuitiva, para facilitar la interacción de usuarios sin experiencia con el sistema.
- La aplicación debe poseer un diseño adaptativo a diferentes tamaños de pantalla, a fin de garantizar la adecuada visualización en diferentes dispositivos móviles.

Apariencia e interfaz

- Todos los textos y mensajes que se mostrarán en pantalla tendrán una letra legible y en español.

Hardware

- Será necesario disponer de un dispositivo móvil inteligente, puede ser un teléfono celular o una tableta.
- El dispositivo debe tener como mínimo: un procesador de un núcleo a 800 MHz, 512Mb de memoria RAM y 80Mb de espacio disponible.

Software

- El dispositivo móvil debe tener un sistema operativo Android versión 4.0 o superior.

2.5 Definición de casos de uso

Un caso de uso (CU) capta un contrato que describe el comportamiento del sistema en distintas condiciones en las que el sistema responde a una petición de alguno de sus participantes. (Pressman, 2010)

A partir del análisis de los requisitos funcionales se definieron los siguientes casos de uso del sistema (CUS):

- CUS1: Gestionar torneos
- CUS2: Gestionar equipos
- CUS3: Gestionar jugadores
- CUS4: Crear nuevo partido

2.5.1 Actor del sistema

Tabla 2. Actor del sistema

Actor	**Descripción**
Usuario	Es la persona que interactúa con la aplicación, dígase un miembro del cuerpo técnico del equipo de balonmano, en este caso un árbitro de mesa. Será el encargado de introducir la información necesaria sobre los torneos, equipos, jugadores y partidos.

2.5.2 Diagrama de casos de uso del sistema

En la figura 4, se muestra el diagrama de casos de uso del sistema.

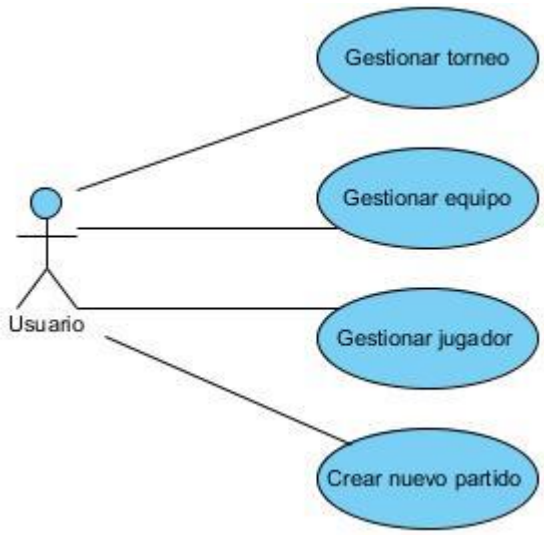


Figura 4. Diagrama de casos de uso del sistema

2.5.3 Descripción de los casos de uso del sistema

La descripción del caso de uso Gestionar jugadores se puede observar en la (tabla3). Las restantes descripciones se pueden consultar en el Anexo 1. Descripción de casos de uso.

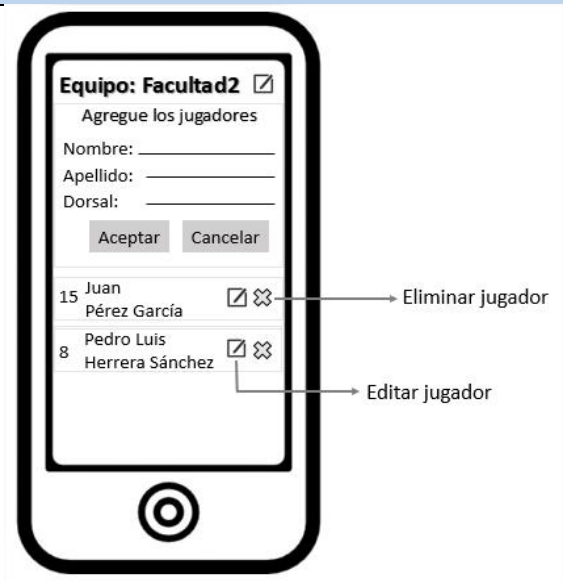
Tabla 3. CUS Gestionar jugador

Nombre	Gestionar Jugadores
Objetivo	Gestionar los jugadores
Actor	Usuario
Resumen	El caso de uso inicia cuando el usuario desea insertar, listar, editar o eliminar jugadores
Complejidad	Media
Prioridad	Media
Precondiciones	El usuario selecciona o crea un jugador dentro de la vista Equipo
Postcondiciones	Se inserta, lista, edita o elimina los jugadores
Flujo de eventos	
Flujo básico Gestionar jugadores	
Actor	**Sistema**
Indica que desea insertar, listar, editar o eliminar un jugador	Si el usuario decide listar los jugadores ver sección 1. Listar jugadores Si el usuario decide insertar un nuevo jugador ver

		sección 2. Insertar jugador Si el usuario decide editar algún jugador ver sección 3. Editar jugador Si el usuario decide eliminar jugador ver sección 4. Eliminar jugador
Sección 1 Listar jugador		
Flujo básico Listar jugador		
Actor		**Sistema**
1	Selecciona un equipo	
2		Muestra una interfaz con una lista de todos los jugadores existentes en ese equipo. Las opciones que se brindan son: -insertar un jugador (ver la sección 2 Insertar Jugador) -editar cada jugador (ver la sección 3 Editar Jugador) -eliminar jugador (ver la sección 4 Eliminar Jugador)
Sección 2 Insertar jugador		
Flujo básico Insertar jugador		
Actor		**Sistema**
1	Selecciona un equipo	
2		Muestra la interfaz Insertar jugador que permite introducir los valores referentes al jugador a insertar: -nombres -apellidos -dorsal
3	Selecciona el botón Aceptar	
4		Valida los valores introducidos del jugador y lo muestra en esa misma pantalla, debajo del formulario para crear el jugador.

Flujo Alterno Insertar jugador		
Actor		**Sistema**
1	En caso de hacer clic en el botón Aceptar y haber dejado algún campo en blanco	
2		El sistema muestra el mensaje de error “Todos los campos son obligatorios”.
Sección 3 Editar jugador		
Flujo básico Editar jugador		
Actor		**Sistema**
1	Selecciona la opción que permite editar el jugador	
2		Muestra la interfaz Editar jugador que permite editar los valores referentes al jugador: -nombre -apellido -dorsal
3	Selecciona el botón Aceptar	
4		Valida los nuevos valores introducidos del jugador y muestra el jugador con sus nuevos valores.
Flujo alterno Editar jugador		
Actor		**Sistema**
1	En caso de dar clic en el botón Aceptar y haber dejado algún campo en blanco	
2		El sistema muestra el mensaje de error “Todos los campos son obligatorios”.
Sección 4 Eliminar jugador		
Flujo básico Eliminar jugador		
Actor		**Sistema**

1	Selecciona la opción que permite eliminar jugador	
2		Muestra un mensaje de información: “Este jugador se eliminará”.
3	Selecciona el botón Confirmar	
4		Elimina el jugador seleccionado y actualiza la lista.
Flujo alterno Eliminar jugador		
Actor		
1	Una vez seleccionado el jugador presiona el botón cancelar	
2		Cancela la petición y muestra nuevamente el listado de los jugadores.
Relaciones	**CU incluidos**	**-**
Prototipo elemental de interfaz gráfica de usuario		



2.6 Arquitectura de software

La arquitectura del software de un programa o sistema de cómputo es la estructura o estructuras del sistema, lo que comprende a los componentes del software, sus propiedades externas visibles y las relaciones entre ellos. (Pressman, 2010)

Una arquitectura del diseño en tres capas, desacopla la interfaz de la navegación y del comportamiento de la aplicación. Mantiene separadas la interfaz, la aplicación y la navegación, lo que simplifica la implementación y mejora la reutilización del código. (Pressman, 2010)

Teniendo en cuenta lo planteado anteriormente, para el desarrollo del sistema se utiliza la arquitectura en tres capas (ver figura 5), la cual permite la implementación de TableStatisticsHB por niveles, logrando la reducción del grado de complejidad y facilitando la adaptación a los cambios permitiendo modificar solo el componente necesario sin tener que revisar el código entero. El sistema se divide en las siguientes capas:

- Capa de presentación (o interfaz del usuario): comunica y captura la información proporcionada por el usuario, permite la interacción entre el usuario y el sistema, se evidencia en las interfaces de usuario.
- Capa lógica de presentación: procesa las peticiones del usuario y se envían las respuestas tras el proceso. Esta capa sirve de intermediaria entre la Interfaz y el Acceso a datos.
- Capa de acceso a datos: se encuentran los datos y es la encargada de acceder a los mismos. Está formada por un gestor de bases de datos que realiza todo el almacenamiento o recuperación de información desde la capa de negocio para proveérsela al usuario.

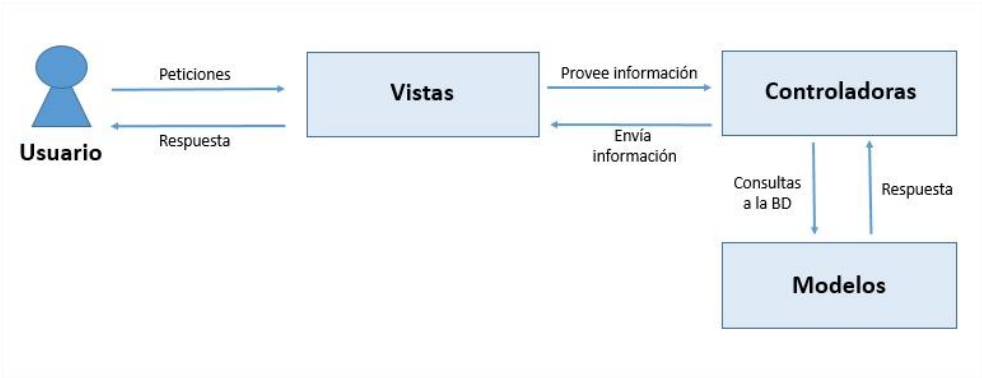


Figura 5. Arquitectura del sistema TableStatisticsHB

2.7 Patrón arquitectónico

Los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes. (Cárdenas, y otros, 2010)

Para el desarrollo de TableStatisticsHB se utiliza el patrón Modelo-Vista-Controlador (MVC), ver figura 6. El cual es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. (Fernández, y otros, 2012)

En la aplicación estos componentes están conformados de la siguiente forma

- Modelo: almacena y gestiona todos los datos que se manejan en la aplicación, incluye la base de datos de los torneos, partidos, equipos y jugadores, así como las acciones recogidas en los partidos y las estadísticas de los torneos.
- Vista: agrupa los layouts, son las vistas que el usuario visualiza en pantalla. Son clases con extensión xml (Lenguaje de Marco Extensible, XML por sus siglas del inglés eXtensible Markup Language), también incluye los resources (recursos del sistema) y los widgets (componentes de los layouts).
- Controlador: incluye los activities, los fragments y los adapters. Los activities son clases encargadas de recibir las peticiones del usuario a través de las vistas, para su posterior análisis y trabajo con la base de datos, los fragments son fragmentos de la interfaz de usuario que pueden añadirse o eliminarse si afectar al resto de los elementos de una interfaz y los adapters son clases que permiten adaptar información a los componentes en las vistas o layouts.

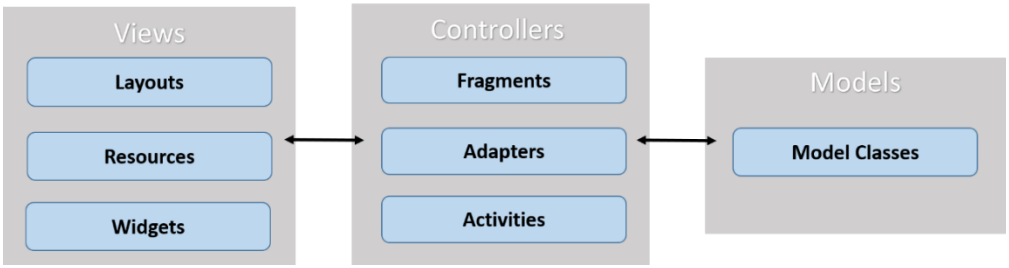


Figura 6. Patrón arquitectónico del sistema TableStatisticsHB

2.8 Patrones de diseño

Un patrón de diseño nombra, abstrae e identifica los aspectos clave de un diseño estructurado, común, que lo hace útil para la creación de diseños orientados a objetos reutilizables. Identifican las clases participantes y las instancias, sus papeles y colaboraciones, y la distribución de responsabilidades. Cada patrón de diseño se enfoca sobre un particular diseño orientado a objetos. (Martínez, 2011)

2.8.1 Patrones GRASP

Los Patrones Generales de Software para Asignación de Responsabilidades (General Responsibility Assignment Software Principles or Patterns GRASP por sus siglas en inglés), son una colección de principios de diseño orientados a objetos, que guían la asignación de responsabilidades sobre los objetos. Es un intento de documentar lo que diseñadores expertos probablemente conozcan intuitivamente. (Trellini, 2015)

A continuación se muestran ejemplos de los patrones utilizados en la aplicación

Experto: responde a la pregunta: ¿cuál es el principio más básico para añadir responsabilidades en una clase? y propone como solución asignar responsabilidades al experto de la información, es decir, a la clase que tiene la información necesaria para llevar la tarea a cabo. (Martínez, 2015)

Este patrón se evidencia en las clases Torneo.java, Equipo.java, Jugador.java y Partido.java, cada una tiene los métodos necesarios para la gestión de la información de los torneos, jugadores, equipos y partidos (ver figura 7).

```
@DatabaseTable(tableName = "equipo")
public class Equipo {
    public final static String ID = "id";
    public final static String NOMBRE = "nombre_equipo";
    public final static String GRUPO = "grupo";

    @DatabaseField(generatedId = true, columnName = ID)
    private int id;
    @DatabaseField(columnName = NOMBRE)
    private String nombre_equipo;
    @DatabaseField(columnName = GRUPO)
    private String grupo;

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getNombre_equipo() { return nombre_equipo; }
    public void setNombre_equipo(String nombre_equipo) { this.nombre_equipo = nombre_equipo; }

    public String getGrupo() { return grupo; }
    public void setGrupo(String grupo) { this.grupo = grupo; }
}
```

Figura 7. Ejemplo del patrón experto en la aplicación

Creador: responde a la pregunta: ¿quién debe ser responsable en la creación de una nueva instancia de una clase? y como solución plantea que debe recaer en la clase que agrega, contiene, registra, utiliza o tiene los datos de inicialización de la nueva instancia. (Martínez, 2015)

Este patrón se ve reflejado en la clase Insertar Jugador, donde se crean instancias de otras clases, por ejemplo jugador es un objeto que se crea de la clase Jugador.java. Ver figura 8.

```
public class InsertarJugador extends AppCompatActivity {

    EditText nombre, apellido, numero;
    Button button;
    DBManager dbManager = new DBManager(this);
    Jugador jugador = new Jugador();
```

Figura 8. Ejemplo del patrón creador en la aplicación

Bajo Acoplamiento: este patrón responde a la pregunta: ¿cómo soportar baja dependencia e incrementar la reutilización? y propone asignar responsabilidades de tal manera que el acoplamiento sea el menor posible. (Martínez, 2015)

Este patrón se evidencia en la relación entre las clases Equipo_Main.java y Jugador.java (ver figura 9), Equipo_Main.java contiene una lista de jugadores que la obtiene de la clase modelo Jugador.java, la cual incluye todos los métodos necesarios para gestionar toda la información referente a los jugadores, de esta manera cualquier cambio que se efectúe con un jugador en Jugador.java no implica un cambio en la clase Equipo_Main.java.

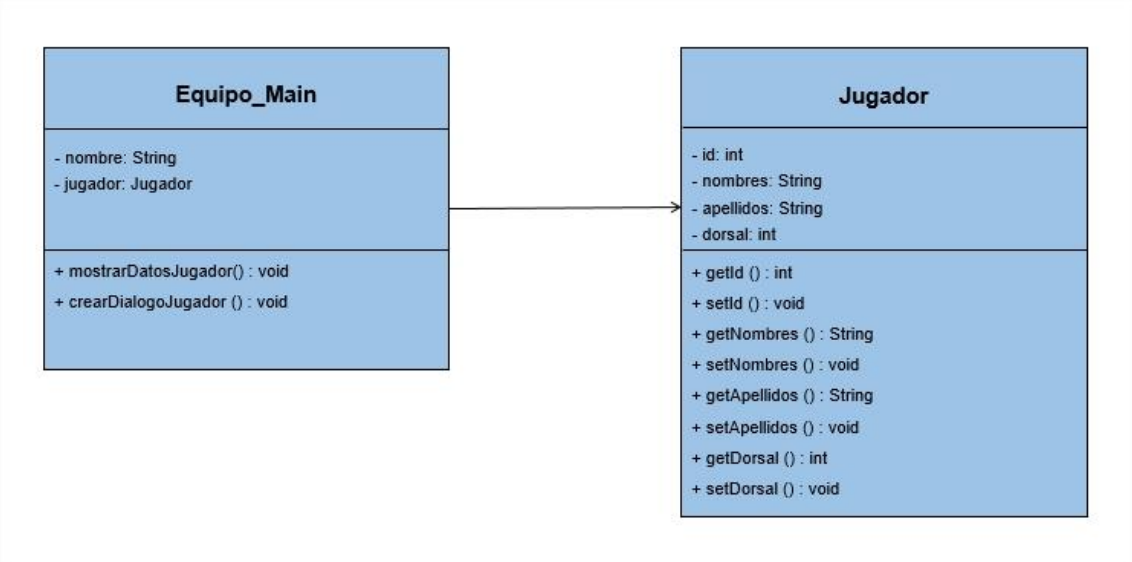


Figura 9. Ejemplo del patrón bajo acoplamiento en la aplicación

Alta Cohesión: este patrón responde a la pregunta: ¿cómo lograr que la complejidad sea lo más manejable posible? Propone asignar responsabilidades procurando que la cohesión sea lo más alta posible. (Martínez, 2015)

En la figura 10, se observa un ejemplo del patrón alta cohesión en la aplicación. GenerarPDF.java es la clase responsable únicamente de generar un archivo con formato PDF, con la planilla oficial de balonmano generada en un partido, dicha planilla se exporta desde la clase Partido_Main.java mediante el método generarPDF perteneciente a la clase GenerarPDF.java.

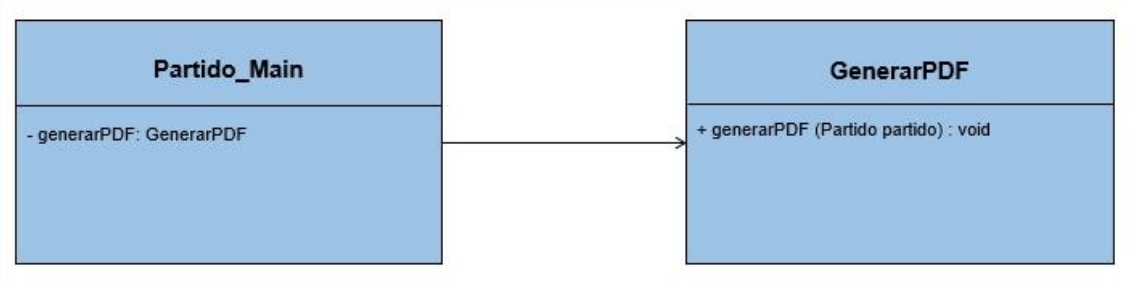


Figura 10. Ejemplo del patrón alta cohesión en la aplicación

Controlador: este patrón responde a la pregunta: ¿quién debe manejar eventos del sistema? Propone asignar responsabilidades para el manejo de mensajes de eventos del sistema a una clase que

represente al conjunto del sistema o negocio, represente algo del mundo real que está activo, o representa un administrador artificioso para todos los eventos del sistema. (Martínez, 2015)

Este patrón se evidencia en todas las clases controladoras de la aplicación, las cuales son las intermediarias entre las clases interfaz y las clases del modelo de datos. Por ejemplo se evidencia en las clases EquipoMain.java, MainActivity.java, TorneoMain.java y PartidoMain.java. En la figura 11 se muestra la clase controladora PartidoMain.java.

```
package com.example.eileen.managerhb.Controladora.Partido;

import ...

public class PartidoMain extends AppCompatActivity {
    private TabLayout tablayout;
    private ViewPager viewPager;
    private ViewPagerAdapter adapter;
    private Partido partido;
    private Torneo torneo;
    String duracion;
    private DBManager dbManager;
    GenerarPDF pdf;

    Fragment_EquipoA tab1;
    Fragment_EquipoB tab2;
    Fragment_Estadisticas_A tab3 = null;
    Fragment_Estadisticas_B tab4;
    Fragment_Resumen tab5;
```

Figura 11. Ejemplo de la utilización del patrón controlador en la aplicación

2.8.2 Patrones GoF

Los patrones Gang of four (Gof) o grupo de los cuatro, se dividen en tres grupos: de creación, estructurales y de comportamiento. Seguidamente se muestran ejemplos de los utilizados en TableStatisticsHB.

Adapter (Adaptador): permite transformar la interfaz de una clase en la interfaz que un cliente desea. Posibilita que dos clases con interfaces incompatibles puedan comunicarse. (Bustacara, 2011)

Este patrón se evidencia en la clase PatidoXFechaAdaptador.java (ver figura 12), esta clase es la intermediaria a entre la clase controladora FragmentPartido.java y la vista fragment_partido.xml, es la responsable de adaptar la información que provee la clase controladora a la clase interfaz.

```
public class PartidoXFechaAdaptador extends RecyclerView.Adapter<PartidoXFechaAdaptador.PartidoViewHolder>  {
    Context context;
    List<Partido> partidos;
    DBManager dbManager;

    public PartidoXFechaAdaptador(Context context, List<Partido> partidos) {
        this.context = context;
        this.partidos = partidos;
    }

    @Override
```

Figura 12. Ejemplo del patrón Adaptador en la aplicación

Builder (constructor): separa la construcción de objetos complejos de su representación, para que el mismo proceso de construcción pueda crear diferentes representaciones. (Bustacara, 2011)

Este patrón se puede observar en la implementación de todos los diálogos de TableStatisticsHB, permite al programador abstraerse de la construcción del objeto Builder y utilizarlo para crear varias representaciones, en este caso crear y modificar los diálogos. (Ver ejemplo en la figura 13)

```
public void crearDialog(Context context) {
    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    LayoutInflater layoutInflater = LayoutInflater.from(context);
    final View myview = layoutInflater.inflate(R.layout.torneo_creartorneo_dialog, null);
    builder.setView(myview);
    final Dialog dialog = builder.create();
    dialog.setCancelable(false);
    dialog.getWindow().getAttributes().windowAnimations = R.style.principal_dialog_animacion;
    //linear = (LinearLayout)findViewById(R.id.linear);
    //Animation animation = AnimationUtils.loadAnimation(getApplicationContext(),R.anim.recyclerview_fade_desaparece);
    //linear.startAnimation(animation);
    dialog.show();

    final EditText n = (EditText) myview.findViewById(R.id.edit_nombre_torneo);
```

Figura 13. Ejemplo del patrón Builder en la aplicación

Singleton: el objetivo de este patrón es asegurarse de que de una clase solo existe una instancia y que esta es accesible. (Universidad del Valle, 2008)

Este patrón se observa en la aplicación (ver figura 14), cuando se crea una única instancia DBManager, que luego es utilizada en las clases que requieren un trabajo con la base de datos

```
/**
 * Created by Eileen on 30/12/2017.
 */
public class DBManager {
    static private DBManager instance;
    private DBHelper dbHelper;

    static public void init(Context context) {
        if (null == instance) {
            instance = new DBManager(context);
        }
    }

    static public DBManager getInstance() { return instance; }

    public DBManager(Context context) { dbHelper = new DBHelper(context); }

    private DBHelper getHelper() { return dbHelper; }
```

Figura 14. Ejemplo del patrón Singleton en la aplicación

.

2.9 Estándares de codificación

Los estándares de codificación se definen con vistas a mejorar el entendimiento del código perteneciente a TableStatisticsHb por otros desarrolladores y alcanzar una uniformidad en el mismo. En el desarrollo de la aplicación se utilizaron los estándares de codificación de Java y de Android, además de los siguientes elementos:

- El nombre de todas las variables y métodos comenzarán con letra minúscula y si este está compuesto por varias palabras se utilizará el estilo de escritura lowerCamelCase (traducido como “notación de camello”).
- Los nombres de las variables deben ser cortos y significativos.
- En el contenido siempre se identará con tabulaciones, nunca utilizando espacios en blanco.
- El nombre de las clases comenzará con mayúsculas y cada salto de palabras debe iniciar con mayúsculas.
- Cuando una expresión exceda la línea, se romperá después de una coma o antes de un operador.

2.10 Modelo de datos

Un modelo de datos es un conjunto de conceptos que pueden ser usados para describir-diseñar la estructura de una Base de Datos (BD). El término “estructura” de una BD se refiere a los tipos de datos, las relaciones, las restricción es que deben cumplirse, y las operaciones de manipulaci0ón de los datos (agregado, borrado, modificación y recuperación de los datos de la base). (Rocha, 2011)

El modelo físico de TableStatisticsHB se puede ver en la (figura 15).

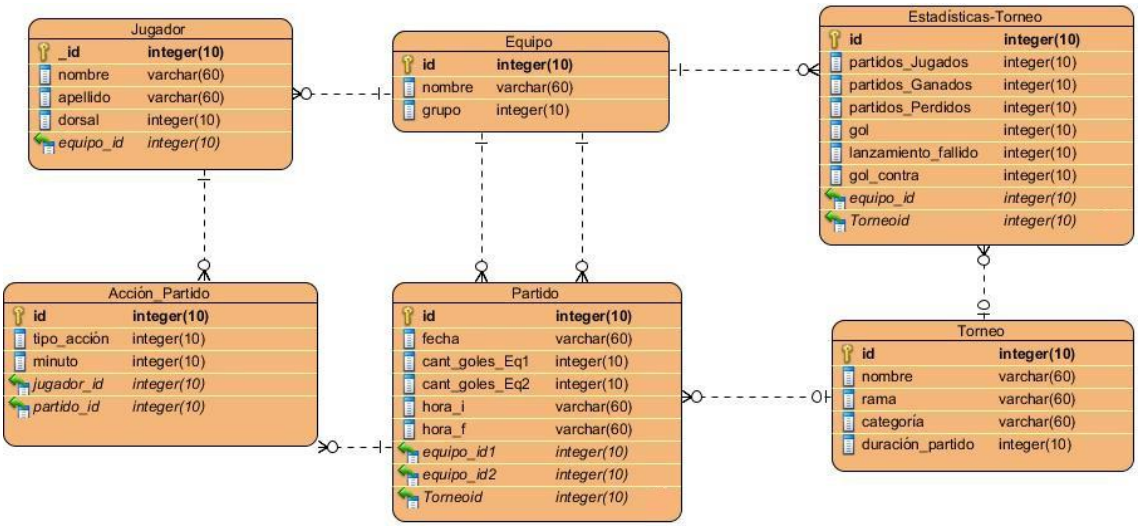


Figura 15. Modelo físico de la base de datos de la aplicación

Descripción de las tablas del modelo físico de TableStatisticsHB

A continuación se muestra una descripción de las tablas del modelo físico de la base de datos de TableStatisticsHB, se describen cada uno de los atributos contenidos en ellas, de ellos se muestra el nombre del atributo, el tipo de dato y una descripción sobre la información que almacenan. La descripción de la tabla Estadística-Torneo se puede observar en la (tabla 4), el resto de las descripciones se encuentran en el Anexo3. Descripción de las tablas de la base de datos.

Tabla 4. Descripción de la tabla Estadística-Torneo

Estadística-Torneo		
Almacena los resultados de un equipo en un torneo		
Atributo	**Tipo**	**Descripción**
Id	Integer	Etiqueta única que identifica una estadística de un equipo en un torneo
partidos_Jugados	Integer	Cantidad de partidos jugados
partidos_Ganados	Integer	Cantidad de partidos ganados
partidos_Perdidos	Integer	Cantidad de partidos perdidos
Gol	Integer	Cantidad de goles
gol_Contra	Integer	Cantidad de goles en contra
lanzamiento_fallido	Integer	Cantidad de lanzamientos fallidos
torneo_id	Integer	Atributo que hace referencia al identificador del torneo en el que se registra la estadística

equipo_id	Integer	Atributo que hace referencia al identificador del equipo del que se registra la estadística

Conclusiones del capítulo

En este capítulo se describió la propuesta de solución. Se definieron los artefactos correspondientes a las etapas de análisis y diseño de la aplicación TableStatisticHB, según la metodología AUP-UCI. Se identificaron y describieron los requisitos funcionales y no funcionales del sistema. Se definió como arquitectura de software a emplear la arquitectura en tres capas y como patrón arquitectónico el MVC, logrando así una aplicación mucho más organizada y menos compleja a la hora de realizar alguna modificación. Se identificaron además, los patrones de diseño GRASP y GoF a utilizar en el desarrollo de la solución y se diseñó la base de datos.

CAPÍTULO 3. IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA

A partir de los resultados obtenidos de las etapas de análisis y diseño, se describen, en este capítulo, los elementos correspondientes con la implementación y validación del sistema TableStatisticsHb. Se muestra el diagrama de componentes, diagrama de despliegue, diagrama de paquetes y se definen los estándares de codificación utilizados para el desarrollo de la aplicación. Se documentan las pruebas de software realizadas y se muestran los resultados obtenidos.

3.1 Modelo de implementación

El modelo de implementación representa cómo los elementos del modelo de diseño se implementan en términos de componentes. De igual forma describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y la relación existente entre ellos. (Jacobson, y otros, 2000)

3.1.1 Diagrama de componentes

En el diagrama de componentes, un componente representa un módulo físico de código o paquete, puede ser relacionado en un diagrama de componentes, el cual muestra varios componentes de un sistema, describiendo sus elementos físicos y sus dependencias. (Jacobson, y otros, 2000)

El diagrama de componentes de TableStatisticsHb, se puede observar en la figura 16.

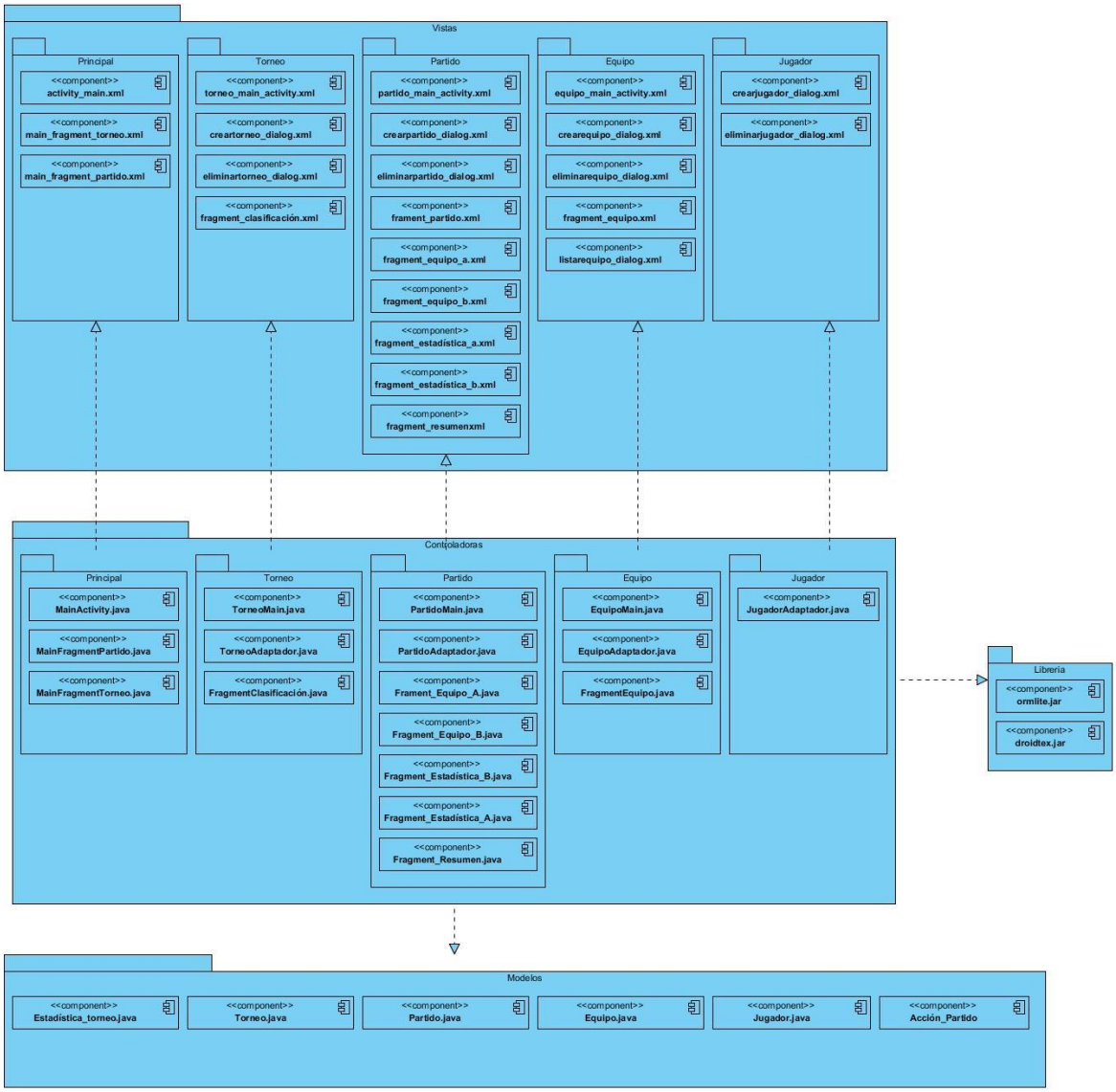


Figura 16. Diagrama de componentes de la aplicación

Descripción de los componentes del Diagrama de Componentes

Vistas: en las vistas se encuentran todas las clases interfaz, son las clases que visualiza el usuario y con las que interactúa directamente

Principal: contiene las vistas principales: main_activity.xml, main_fragment_torneo.xml y main_fragment_partido.xml

Torneo: almacena las siguientes clases que permiten crear, editar, visualizar y eliminar los torneos: torneo_main_activity.xml, creartorneo_dialog.xml y eliminartorneo_dialog.xml. Contiene además la clase fragment_clasificacion.xml, que permite mostrar una tabla de posiciones con los equipos que participan en el torneo y sus resultados en el mismo

Partido: contiene las siguientes clases que permiten que el usuario pueda gestionar toda la información referente a un partido: partido_main_activity.xml, crearpartido_dialog.xml, eliminarpartido_dialog.xml, fragment_partido.xml. También agrupa las clases: fragment_equipo_a.xml y fragment_equipo_b.xml, que muestran los jugadores de cada equipo que participa en el partido, fragment_resumen.xml que muestra un historial de las acciones (tipo de acción, minuto y jugador) realizadas por los jugadores de los equipos involucrados, Fragment_estadistica_a.xml y Fragment _estadistica_b.xml que permiten visualizar los equipos correspondientes.

Equipo: agrupa las clases que permiten gestionar los equipos: equipo_main.xml, crearequipo_dialog.xml, eliminarequipo_dialog.xml y fragment_equipo.xml. La clase equipo_main.xml muestra además todos los jugadores que tiene dicho equipo.

Jugador: contiene las clases que permiten crear, editar y eliminar un jugador: crearjugador_dialog.xml y eliminarjuagdor_dialog.xml.

Controladoras: incluye todas las clases controladoras, estas son las encargadas de procesar la información introducida por el usuario y del trabajo con la base de datos.

Principal: incluye la clase MainActivity.java que contiene: Main_FragmentPartido.java que es la que llama al diálogo crearpartido_dialog.xml y MainFragmentTorneo.java que llama al diálogo craertorneo_dialog.xml.

Torneo: contiene las clases TorneoMain.java, encargada de procesar la información referente a los torneos, TorneoAdaptaer.java es el vínculo entre las clases interfaz y las clases controladoras del torneo y FragmentClasificación.java controla la información que se muestra en fragment_clasificación.xml.

Partido: agrupa las clases que procesan la información introducida por el usuario, referente a los partidos. PartidoMain.java lleva el control del cronómetro general del partido, los cronómetros de las amonestaciones de los jugadores y el marcador del partido. PartidoAdaptador.java adapta la información de las controladoras de partido a las clases interfaz del mismo. Fragment_Equipo_A.java, Fragment_Resumen.java, Fragment_Estadísticas_B.java, Fragment_Estadísticas_A.java y Fragment_Equipo_B.java, gestionan las acciones de los jugadores de ambos equipos.

Equipo: incluye la clase EquipoMain.java que contiene la clase FragmentEquipo.java, que controla los equipos creados en el torneo y EquipoAdaptador.java adapta dicha información a las clases interfaz de equipo. EquipoMain.java gestiona además, todos los jugadores que pertenecen al equipo correspondiente.

Jugador: contiene la clase JugadorAdaptador.java, que se encarga de adaptar los datos de los jugadores a las vistas de jugador.

Librería: incluye la librería ormlite.jar, que permite hacer consultas a la base de datos mediante código java.

Modelos: agrupa todas las clases modelo de la base de datos: Estadísticas_Partido.java, Torneo.java, Partido.java, Equipo.java y Jugador.java.

3.2 Diagrama de despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos. (Sparx System, 2018)

El diagrama de despliegue de la aplicación se muestra en la figura 17.

Figura 17. Diagrama de despliegue de la aplicación

Dispositivo Móvil: representa el hardware donde se ejecutará el sistema. La base de datos que utiliza el mismo es interna y estará representada por el fichero tshb.db, el cual almacenará toda la información con que trabaja la aplicación. El sistema operativo del dispositivo debe ser Android. Este debe contar con los requisitos no funcionales especificados en el acápite 2.4.

3.3 Diagrama de paquetes

Presentan cómo se organizan los elementos de modelado en paquetes y las dependencias entre ellos, incluyendo importaciones y fusiones de paquetes. (Ruiz, y otros, 2015)

El diagrama de paquetes de la aplicación se puede ver la siguiente figura, (ver figura 18).

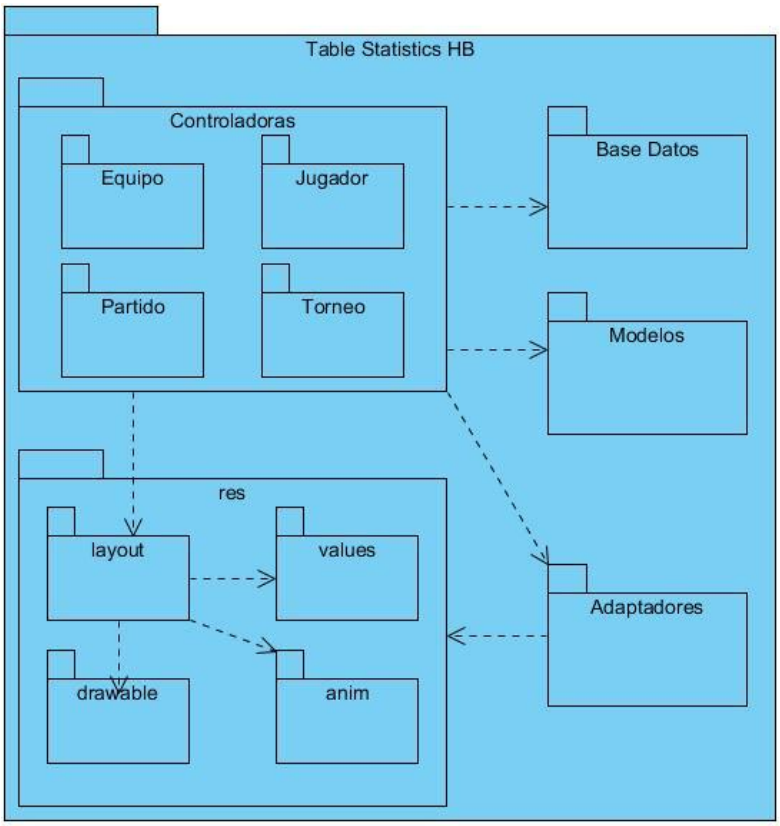


Figura 18. Diagrama de paquetes de la aplicación

Descripción de los paquetes del Diagrama de Paquetes

Controladoras: contiene todas las clases controladoras de la aplicación

Equipo: almacena todas las clases controladoras relacionadas con los equipos

Jugador: almacena todas las clases controladoras relacionadas con los jugadores

Torneo: almacena todas las clases controladoras relacionadas con los torneos

Partido: almacena todas las clases controladoras relacionadas con los partidos

Res: contiene todo lo relacionado con la interfaz de la aplicación: las vistas, imágenes, estilos, animaciones

Layout: almacena los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica

Drawable: contiene todas las imágenes de la aplicación

Anim: contiene las animaciones

Values: contiene otros ficheros XML de recursos de la aplicación, como por ejemplo cadenas de texto (strings.xml), estilos (styles.xml) y colores (colors.xml)

Base de datos: contiene las clases encargadas de la gestión de la base de datos. DBHelper es la clase responsable de la creación de la base de datos, y DBManager la que contiene todas las consultas a la base de datos

Modelos: almacena todas las clases modelos de la aplicación

Adaptadores: contiene las clases adaptadores, encargadas de tomar la información gestionada por las controladoras y adaptarla a las vistas.

3.4 Pruebas de software

Luego de la implementación de la aplicación es necesario realizarle un conjunto de pruebas, con el objetivo de detectar las fallas que presenta el sistema y corregirlas, para garantizar la calidad del producto y su cumplimiento con los requisitos definidos.

Las pruebas de software son una actividad primordial para desarrollar un software de calidad. Aseguran el correcto cumplimiento de la funcionalidad del producto, ayudan a ganar confianza, confirman la fiabilidad del uso y previenen defectos en producción. (Campos, 2015)

3.4.1 Estrategia de prueba

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuando se planean y llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recurso requerirán. (Pressman, 2010)

Las pruebas son realizadas en diferentes niveles de esfuerzo. Estos niveles se distinguen en general por el rol de quien las ejecuta y las técnicas utilizadas. (Cano, 2013)

Niveles de prueba: (Cano, 2013)

1. Pruebas unitarias
2. Pruebas de integración
3. Pruebas de sistema
4. Pruebas de aceptación

Para garantizar la calidad del sistema TableStatisticsHb, se realizaron pruebas en dos de los cuatro niveles anteriormente definidos: en el de pruebas unitarias y en el de pruebas del sistema.

Para el nivel de pruebas de unitarias se desarrollaron pruebas de caja blanca, y para el nivel de sistema pruebas de caja negra.

Las pruebas de caja blanca en ocasiones llamadas pruebas de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. (Pressman, 2010)

Por otra parte las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software, es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. (Pressman, 2010)

3.4.2 Pruebas unitarias

Las pruebas unitarias dirigen su atención a los componentes. Un componente es la unidad más pequeña especificada de un sistema, las pruebas se llevan a cabo tras la construcción o realización de cada componente para verificar que la implementación se esté llevando conforme a los estándares acordados. Hace referencia a los componentes como módulos, clases o unidades. (Campos, 2015)

La aplicación se dividió en los siguientes cuatro módulos para realizar las pruebas unitarias:

1. Partido: agrupa todas las clases referentes a los partidos
2. Torneo: agrupa todas las clases referentes a los torneos
3. Equipo: agrupa todas las clases referentes a los equipos
4. Jugador: agrupa todas las clases referentes a los jugadores.

A estos módulos se les aplicaron pruebas de caja blanca, utilizando la técnica: prueba de ruta o trayectoria básica, la cual permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba. (Pressman, 2010)

A continuación se muestra una prueba de caja blanca realizada al método setOnClickListener (ver figura 19) de la clase FragmentEquipo.java del módulo Equipo.

```
1  crear.setOnClickListener((view) -> {
           equipo = new Equipo();
           dbManager = new DBManager(getContext());
2      if (n.getText().toString().trim().length()==0 || nombreEnUso(n.getText().toString()))
3          if (n.getText().toString().trim().length()==0)
4              n.setError("Campo Vacio");
5          else (nombreEnUso(n.getText().toString()))
6              n.setError("Este equipo ya existe");
7      else {
           equipo.setNombre_equipo(n.getText().toString());
           dbManager.insertarEquipo(equipo);
           Intent intent = new Intent(getContext(), Equipo_Main.class);
           int id = dbManager.extraerId(equipo, null, null, null);
           Bundle bundle = new Bundle();
           bundle.putInt("idEquipo", id);
           intent.putExtras(bundle);

8          //llenar tabla estadistica-torneo
           est.setTorneo(dbManager.queryForIdTorneo(idTorneo));
           est.setEquipo(equipo);
           dbManager.insertarEstadisticaTorneo(est);

           startActivity(intent);
           getActivity().finish();
           mostrarListado();
           dialog.cancel();
9      }
10 });
```

Figura 19. Método setOnClickListener

Grafo de flujo del método setOnClickListener:

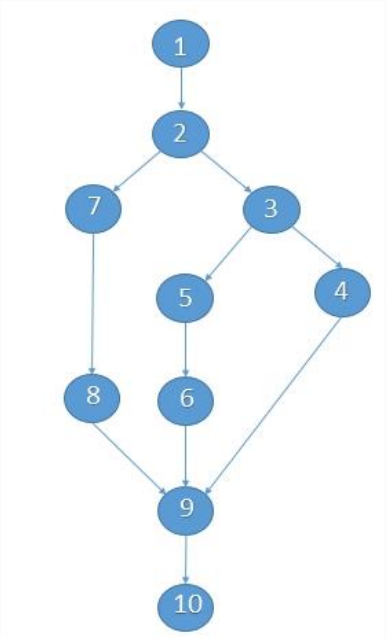


Figura 20. Grafo de flujo del método setOnClickListener

La complejidad ciclomática es una medición de software que proporciona una evaluación cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba de la ruta básica, el valor calculado por la complejidad ciclomática define el número de rutas independientes del conjunto básico de un programa. (Pressman, 2010)

La complejidad ciclomática V (G) del grafo se calcula en una de las siguientes tres formas: (Pressman, 2010)

1. $V(G) = E - N + 2$, donde E es el número de aristas del grafo de flujo y N el número de nodos del grafo de flujo
2. El número de regiones del grafo de flujo
3. $V(G) = P + 1$, P es el número de nodos predicados contenidos en el grafo de flujo

Complejidad ciclomática del grafo de flujo del método setOnClickListener

1. $V(G) = E - N + 2$
   $V(G) = 11 - 10 + 2$
   $V(G) = 3$
2. El grafo tiene 3 regiones
3. $V(G) = P + 1$
   $V(G) = 2 + 1$
   $V(G) = 3$

Por tanto la complejidad ciclomática del grafo de flujo de la figura 20 es 3, lo que representa que el grafo tiene 3 caminos básicos de rutas linealmente independientes:

1. 1-2-7-8-9-10
2. 1-2-3-5-6-9-10
3. 1-2-3-4-9-10

Se diseñaron tres casos de prueba, uno para cada camino y se probaron con el debugueo de la aplicación.

Tabla 5. Casos de prueba

Casos de prueba	Entradas	Resultados
Caso 1.Insertar equipo	Para un valor de **n** != vacío y el valor de **n** no existe en la clase modelo Equipo	Se inserta el equipo con el nombre introducido por el usuario
Caso 2. Campo nombre vacío	Para un valor de **n**=vacío	Se lanza un mensaje de error: “Campo vacío”
Caso 3. Existe equipo con ese nombre	Para un valor de **n** igual al nombre de otro equipo en la base de datos	Se lanza un mensaje de error: “Este equipo ya existe”

Después de realizada la prueba, no se detectó ninguna No Conformidad (NC). La primera NC (NC1) se detectó en el método onKeyDown de la clase PartidoMain.java que pertenece al módulo Partido (ver figura 21), consistió en un error de objeto con referencia nula de la variable duración, se había creado, pero no hacía referencia a ningún componente. Se solucionó inicializando la variable con el identificador del componente que le correspondía en la clase interfaz PartidoMain.xml.

Se detectó otra NC (NC2) en la clase EliminarPartido.java del módulo Partido, en el método setOnClickListener del botón eliminar, cuya funcionalidad era eliminar un partido. El error consistió en que al eliminar un partido no se actualizaban las estadísticas de los equipos que habían participado en él. Se solucionó modificando la consulta eliminarPartidoPorId que se utiliza en el método setOnClickListener. (Ver figura 22)

En pruebas de este tipo realizadas al resto de los módulos de la aplicación no se detectaron más NC.

```
140    public boolean onKeyDown(int keyCode, KeyEvent event) {
141        List<AccionPartido> acciones = new ArrayList<>();
142        if (keyCode == KeyEvent.KEYCODE_BACK && event.getRepeatCount() == 0) {
143            if(cont_atras ==1) {
144                if (estado.equals("nuevo")) {
145                    if (duracion.equals(tvCrono.getText())) {
146                        Toast.makeText(PartidoMain.this, "este es el fin", Toast.LENGTH_SHORT).show();
147                        Intent intent = new Intent(PartidoMain.this, TorneoMain.class);
148                        Bundle bundle = new Bundle();
149                        bundle.putInt("idTorneo", idTorneo);
150                        bundle.putInt("Pos", 1);
151                        intent.putExtras(bundle);
152                        startActivity(intent);
153                        finish();
154                    }
155                    if (!duracion.equals(tvCrono.getText())) {
156                        Toast.makeText(PartidoMain.this, "aun no ha acabado", Toast.LENGTH_SHORT).show();
157                        cronoPrinc.reiniciar();
158                        acciones = dbManager.listarAccionPartido(idPartido);
159                        dbManager.eliminarAcciones(acciones);
160                        dbManager.eliminarPartidoPorId(idPartido);
161                        Intent intent = new Intent(PartidoMain.this, TorneoMain.class);
162                        Bundle bundle = new Bundle();
163                        bundle.putInt("idTorneo", idTorneo);
164                        bundle.putInt("Pos", 1);
165                        intent.putExtras(bundle);
166                        startActivity(intent);
167                        finish();
168                    }
169                }else {
170                        Intent intent = new Intent(PartidoMain.this, TorneoMain.class);
171                        Bundle bundle = new Bundle();
172                        bundle.putInt("idTorneo", idTorneo);
173                        bundle.putInt("Pos", 1);
174                        intent.putExtras(bundle);
175                        startActivity(intent);
176                        finish();
177                }
178            }
179            if(cont_atras ==0){
180                cont_atras++;
181                Toast.makeText(this, "Presione una vez más para salir", Toast.LENGTH_SHORT).show();
182                TimerTask timerTask = () -> { cont_atras =0; };
188                Timer timer=new Timer();
189                timer.schedule(timerTask, 2000);
190            }
191            return false;
192        }
193        return super.onKeyDown(keyCode, event);
194    }
```

Figura 21. Método onKeyDown

```
eliminar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        dbManager = new DBManager(EliminarPartido.this);
        dbManager.eliminarPartidoPorId(idPartido);
        Intent intent= new Intent(EliminarPartido.this, TorneoMain.class);
        Bundle bundle = new Bundle();
        bundle.putInt("idTorneo", idTorneo);
        bundle.putInt("Pos", 1);
        intent.putExtras(bundle);
        startActivity(intent);
        finish();
    }
});
```

Figura 22. Método setOnClickListener

3.4.3 Pruebas de sistema

Las pruebas de sistema se llevan a cabo cuando todo el desarrollo ha sido culminado y se tiene una versión preliminar del sistema que saldrá a producción. Esta etapa de pruebas consiste en probar un sistema integrado con el objeto de comprobar el cumplimiento de requisitos especificados. Las pruebas de sistema se desarrollan utilizando casos de prueba funcionales y no funcionales. (Campos, 2015)

Algunos de los casos de prueba que fueron diseñados para realizar estas pruebas de caja negra al sistema, se muestran en las siguientes tablas, (ver tablas 6, 7, 8 y 9), el resto puede ser consultado en el Anexo 2. Pruebas de Sistema.

Tabla 6. Caso de prueba Listar Jugadores

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Listar Jugadores	Permite al usuario listar los jugadores creados en el equipo seleccionado.	Muestra una lista de todos los jugadores creados por el usuario en el equipo seleccionado.	Seleccionar en la vista principal, un Torneo previamente creado. Seleccionar en la vista Torneo, el equipo.

Tabla 7. Caso de prueba Eliminar Jugador

Escenario	Descripción	Respuesta del sistema	Flujo central

EC 1.2 Eliminar Jugador	Permite al usuario eliminar el jugador seleccionado	Elimina el jugador seleccionado por el usuario.	Seleccionar en la vista principal, un Torneo previamente creado Seleccionar en la vista Torneo, el equipo Seleccionar en la vista Equipo, el jugador que desea eliminar Seleccionar Eliminar Jugador

Tabla 8. Caso de prueba Listar Equipo

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 2 Listar equipo	Permite al usuario listar los equipos creados dentro del torneo seleccionado	Muestra un listado con los equipos creados en el torneo seleccionado	Seleccionar en la vista principal, un Torneo previamente creado

Tabla 9. Caso de prueba Eliminar Equipo

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 2.1 Eliminar equipo	Permite al usuario eliminar el equipo seleccionado	Elimina el equipo seleccionado	Seleccionar en la vista principal, un Torneo previamente creado Seleccionar el equipo que el usuario desee eliminar Seleccionar Eliminar Equipo

Después de realizadas las pruebas al sistema fueron detectadas cinco NC

NC3: cuando se importa un equipo, no se importan los jugadores creados en ese equipo.

NC4: cuando la base de datos está vacía y se crea un jugador, no se muestra en el listado de jugadores.

NC5: en la pantalla principal, cuando se inicia un partido fuera de un torneo, los datos de este partido no se guardan en la base de datos.

NC6: en la vista Equipo cuando hay equipos creados, sale el mensaje “no hay equipos creados”.

NC7: en la vista Equipo, si el equipo fue importado, al seleccionar atrás la aplicación muestra la vista del torneo donde se creó el equipo por primera vez.

Las NC detectadas fueron solucionadas en las iteraciones de las pruebas (ver figura 23). Además, se probó la solución propuesta en varias ocasiones, formando el cliente una parte significativa de las pruebas y obteniéndose así un resultado satisfactorio.

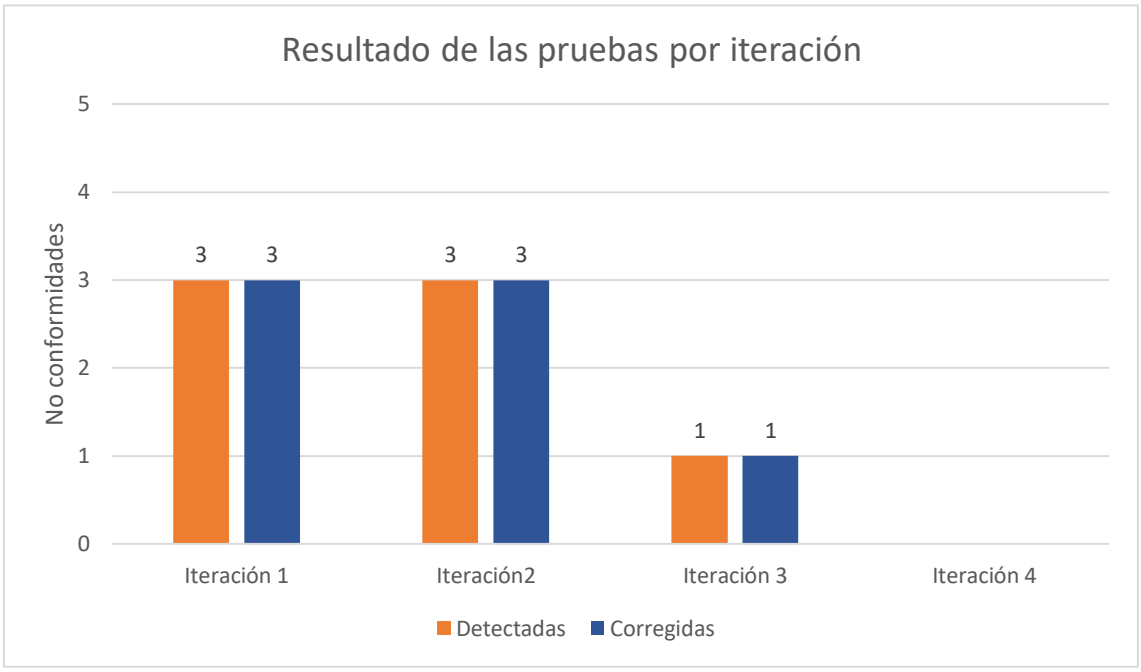


Figura 23. Resultado de las pruebas por iteración

Las no conformidades detectadas en cada una de las iteraciones de las pruebas realizadas se pueden ver en la (tabla 10).

Tabla 10.Resultado de las pruebas

Número	**Iteración**	**Observación**
NC1	1	Detectada y corregida en la iteración 1
NC2	1	Detectada y corregida en la iteración 1
NC3	1	Detectada y corregida en la iteración 1
NC4	2	Detectada y corregida en la iteración 2

NC5	2	Detectada y corregida en la iteración 2
NC6	2	Detectada y corregida en la iteración 2
NC7	3	Detectada y corregida en la iteración 3

3.4.4 Pruebas de rendimiento

Las pruebas de rendimiento se diseñan para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado. (Pressman, 2010)

Las pruebas de rendimiento a TableStatisticsHB se le realizaron mediante dispositivos móviles con diferentes características (ver tabla 11) y mediante el emulador Koplayer (ver Anexo 4. Pruebas de rendimiento), se tuvo en cuenta los siguientes parámetros:

- Dimensiones y resolución de pantalla
- Frecuencia de trabajo del microprocesador
- Cantidad de núcleos del microprocesador
- Memoria de acceso aleatorio (Random Access Memory, por sus siglas en inglés RAM)
- Memoria interna del dispositivo
- Versión del sistema operativo Android

Tabla 11. Visualización de la aplicación en distintos dispositivos móviles

Nombre del dispositivo móvil	**Dimensiones y resolución de pantalla**	**Cantidad de núcleos y frecuencia del microprocesador**	**RAM**	**SO**	**Memoria interna**	
Samsung Galaxy J5 Prime	5" pulgadas 720 x 1280 pixels	Procesador quad-core a 1.4GHz	512Mb	Android 6.0.1	16Gb	

Xiaomi Redmi Note 4	5.5" pulgadas 1280 x 1920 pixels	Procesador octa-core a 2.0Ghz	3Gb	Android 7.0.0	32Gb	
Alcatel OT-4033X (POP C3)	4" pulgadas 480 x 800 pixels	Procesador dual-core a 1.3Ghz	512Mb	Android 4.2.2	4Gb	

Conclusiones del Capítulo

En este capítulo fue descrito el proceso de implementación de TableStatisticsHB a través de los artefactos utilizados para llevar a cabo la aplicación; los diagramas de componentes, despliegue y paquetes. Además, se describió el estándar de codificación empleado en la escritura del código fuente del sistema. Se presentó la estrategia de prueba seleccionada, que abarcó la realización de las pruebas unitarias y las pruebas al sistema como métrica para erradicar los fallos y medir el correcto funcionamiento y eficiencia del producto.

CONCLUSIONES GENERALES

1. Al finalizar la presente investigación y una vez implementado el sistema informático de registro y control temporal de acciones de un partido de balonmano por los árbitros de mesa, se puede concluir que:
2. El análisis de la bibliografía técnica del balonmano, principalmente del reglamento y de la planilla oficial de anotación, permitió conocer las acciones significativas registradas por los árbitros de mesa durante un partido.
3. El estudio de las aplicaciones existentes a nivel nacional e internacional, enfocadas a la gestión de información de los partidos de balonmano, demostró que ninguna cumple con las necesidades del cliente, por lo que se hizo necesario la implementación de TableStatisticsHB.
4. El análisis de las herramientas y tecnologías para el desarrollo de aplicaciones móviles sobre la plataforma Android, posibilitó la selección de las más idóneas teniendo en cuenta las características de TableStatisticsHB.
5. Teniendo en cuenta la complejidad del sistema a desarrollar, el dominio de las herramientas y tecnologías, el tiempo de desarrollo y que es un solo desarrollador se seleccionó una metodología de desarrollo ágil, y dentro de las ágiles se escogió AUP-UCI.
6. Se diseñó una solución para la recogida de información durante los partidos de balonmano por los árbitros de mesa, agilizando así este proceso, lo que permitió el cálculo automático de estadísticas y la gestión de jugadores, equipos y torneos.
7. Los requerimientos identificados en las entrevistas con los clientes, permitieron la implementación de las funcionalidades de TableStatisticsHB, haciendo uso de estándares y buenas prácticas del diseño y desarrollo de sistemas informáticos.
8. Los métodos de pruebas utilizados para validar TableStatisticsHB, permitieron la identificación de no conformidades existentes en la aplicación y su posterior corrección.

RECOMENDACIONES

Luego de implementado el sistema TableStatisticsHB, la autora del presente trabajo recomienda:

Añadir en próximas versiones, la funcionalidad de compartir la base de datos desde la propia aplicación.

REFERENCIAS BIBLIOGRÁFICAS

Android Corporation. Academia Android. [En línea] 2017. [Citado el: 11 de dociembre de 2017.] http://academiaandroid.com/introduccion-a-orm-y-greendao/.

Boeras, Mairelys, Cabrera, Laritza y Llano, Eileén. *Aplicando el método de Boehm y Turner.* 2012.

Bustacara, Cesar Julio.*Diseño de Software Basado en Patrones.* 2011.

Campos, Cindy. *Las pruebas en el desarrollo del software.* México : s.n., 2015.

Cano, Andres Felipe.*Niveles de prueba.* 2013.

Cárdenas, Néstor Raúl y Arias, Jaime Eduardo.*Patrones Arquitectónicos de Software.* 2010.

Federación Internacional de Balonmano. Estatutos y Regulaciones. *International Handball Federation.* [En línea] Julio de 2016. [Citado el: 16 de Enero de 2018.] http://www.ihf.info/files/Uploads/NewsAttachments/0_New-Rules%20of%20the%20Game_GB.pdf.

Fernández, Yenisleidy y Díaz, Yanette. *Patrón Modelo-Vista-Controlador.* La habana : s.n., 2012.

Figueroa, Robert, Solís, Camilo y Cabrera, Armando. *Metodologías Tradicionales vs Metodologías Ágiles.* 2014.

García, Ivan González. *Herramientas tecnológicas para el análisis del juego de balonmano en tiempo real.* 2016.

Google Play. Google Play. *Estadísticas Balonmano.* [En línea] 2017. [Citado el: 1 de octubre de 2017.] https://play.google.com/store/apps/details?id=com.whattheappz.handball..

Google Play. HB-ALL Handball Statistics. [En línea] 2017. [Citado el: 9 de octubre de 2017.] https://play.google.com/store/apps/details?id=de.hball.core.

Google Play. Scoreboard handball. [En línea] 2017. [Citado el: 9 de ocutubre de 2017.] https://play.google.com/store/apps/details?id=it.alecs.puntihandball&hl=es.

Gutierrez, Demián. *Métodos de desarrollo de software.* 2011.

International Data Corporation. IDC. International Data Corporation. IDC Analyze the Future. Smartphone OS. [En línea] 2017. [Citado el: 12 de 10 de 2017.] http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

Jacobson, Ivar, Booch, Grady y Rumbaugh, Jame. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000. ISBN 84-7829-036-2.

Letelier, Paticio y Penadés, Carmen. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* 2013.

Malave, Kristel. *"Android" el sistema operativo de Google para dispositivos moviles.* 2011.

Martínez, Ernesto y Olavarrieta, Dario Marcel. *Sistema para el análisis de acciones tácticas significativas de los equipos de balonmano.* 2017.

Martínez, Francisco Javier.*Guía de construcción de software en JAVA con patrones de diseño.* 2011.

Martínez, Ivette. *Diseño con patrones.* 2015.

Martínez, José Miguel y Silva, Camilo Andrés. *Ingeniería de requerimientos. Guía metodológica para el levantamiento y análisis de Requerimientos de software en base a procesos de negocio.* 2010.

Montiel, Daniel Ponsoda. *Introducción a SQlite.* 2008.

Obregón, María de los Angeles y Lambert, Heimer.*Sistema informático para el análisis del comportamiento de acciones tácticas significativas de los equipos de balonmano.* 2010.

Pavón, Juan. *Introducción al lenguajes java.* 2013.

Pressman, Roger S. *Ingeniería del software. Un enfoque práctico.* 2010.

Rio, Sergio Sánchez. *Metodologías de Análisis y Diseño. Modelo conceptual.* 2011.

Rocha, Marcelo. *Modelado de datos.* 2011.

Rodríguez, Manuel José García.*Estudio comparativo entre las metodologías ágiles y las metodologías tradicionales para la gestión de proyectos de software.* 2015.

Ruiz, Francisco y Lopez, Patricia. *INGENIERÍA DEL SOFTWARE I. Tema11.Arquitectura Lógica del Sistema.* 2015.

Sánchez, Tamara Rodríguez. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.

Sierra, Antonio. *UML (Unified Medling Language) Lenguaje Unificado de Modelado.* 2013.

Sommerville, Ian. *Ingeniería de software. Séptima edición.* 2005.

Sparx System. Sparx System. [En línea] 2018. [Citado el: 20 de marzo de 2018.] http://www.sparxsystems.com.ar/about.html.

Trellini, Ariel. 2015. *Arquitectura y diseño de sistemas.* 2015.

Universidad del Valle. 2008. *Patrones GoF (Gang of Four).* 2008.

Valdés, David. 2016. *Aplicación para la gestión de perfiles en sistemas operativos Android.* 2016.

BIBLIOGRAFÍA

Android Corporation. Academia Android. [En línea] 2017. [Citado el: 11 de dociembre de 2017.] http://academiaandroid.com/introduccion-a-orm-y-greendao/.

Aragón, Licesio J. Rodríguez. *Sistemas operativos y aplicaciones. Infromática básica.* 2017.

Avello Huala, Luis Fernado. *Modelo de comunicación punto a punto para aplicaciones colaborativas en dispositivos móviles.* Santiago de Chile : s.n., 2010.

Blanco, Carlos. *Patrones de Diseño. Ingenieria de software1.* 2010.

Boeras, Mairelys, Cabrera, Laritza y Llano, Eileén. *Aplicando el método de Boehm y Turner.* 2012.

Bustacara, Cesar Julio. *Diseño de Software Basado en Patrones.* 2011.

Campos, Cindy. *Las pruebas en el desarrollo del software.* México , 2015.

Cano, Andres Felipe. *Niveles de prueba.* 2013.

Cárdenas, Néstor Raúl y Arias, Jaime Eduardo. *Patrones Arquitectónicos de Software.* 2010.

De Silva Fariña, Antonio. *Desarrollo de aplicaciones Android utilizando la tecnologías Wifi-Direct.* 2016.

Developers. [En línea] [Citado el: 2018 de febrero de 10.] MODELO DE COMUNICACIÓN PUNTO A PUNTO PARA APLICACIONES COLABORATIVAS EN DISPOSITIVOS MÓVILES.2018

Federación Internacional de Balonmano. Estatutos y Regulaciones. *International Handball Federation.* [En línea] Julio de 2016. [Citado el: 16 de Enero de 2018.] http://www.ihf.info/files/Uploads/NewsAttachments/0_New-Rules%20of%20the%20Game_GB.pdf.

Fernández, Yenisleidy y Díaz, Yanette. *Patrón Modelo-Vista-Controlador.* La habana : s.n., 2012.

Figueroa, Robert, Solís, Camilo y Cabrera, Armando. *Metodologías Tradicionales vs Metodologías Ágiles.* 2014.

García, Ivan González. *Herramientas tecnológicas para el análisis del juego de balonmano en tiempo real.* 2016.

Google Play. Google Play. *Estadísticas Balonmano.* [En línea] 2017. [Citado el: 1 de octubre de 2017.] https://play.google.com/store/apps/details?id=com.whattheappz.handball..

Google Play. HB-ALL Handball Statistics. [En línea] 2017. [Citado el: 9 de octubre de 2017.] https://play.google.com/store/apps/details?id=de.hball.core.

Google Play. Scoreboard handball. [En línea] 2017. [Citado el: 9 de ocutubre de 2017.] https://play.google.com/store/apps/details?id=it.alecs.puntihandball&hl=es.

Gutierrez, Demián. *Métodos de desarrollo de software.* 2011.

Handball Manager 13 v1.0. [En línea] [Citado el: 1 de noviembre de 2017.] http://apk-dl.com/handball-manager-13/.

Indicadores para el control y evaluación de la preparación técnico táctica en el balonmano. **Aróstica Villa, Orelvis, Sembrano Rodríguez, Carlos Rafael y Hurtado Rodríguez, Jacinto.** 2016, Vol. 12.

International Data Corporation. IDC. International Data Corporation. IDC Analyze the Future. Smartphone OS. [En línea] 2017. [Citado el: 12 de 10 de 2017.] http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

Jacobson, Ivar, Booch, Grady y Rumbaugh, Jame. *El Proceso Unificado de Desarrollo de Software.* Madrid , 2000. ISBN 84-7829-036-2.

Letelier, Paticio y Penadés, Carmen. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* 2013.

Malave, Kristel. *"Android" el sistema operativo de Google para dispositivos moviles.* 2011.

Martínez, Ernesto y Olavarrieta, Dario Marcel. *Sistema para el análisis de acciones tácticas significativas de los equipos de balonmano.* 2017.

Martínez, Francisco Javier. *Guía de construcción de software en JAVA con patrones de diseño.* 2011.

Martínez, Ivette. 2015. *Diseño con patrones.* 2015.

Martínez, José Miguel y Silva, Camilo Andrés. *Ingeniería de requerimientos. Guía metodológica para el levantamiento y análisis de Requerimientos de software en base a procesos de negocio.* 2010.

Ministerio de Modernización. *Estándares de aplicaciones móviles.* 2012.

Montiel, Daniel Ponsoda. *Introducción a SQlite.* 2008.

Obregón, María de los Angeles y Lambert, Heimer. *Sistema informático para el análisis del comportamiento de acciones tácticas significativas de los equipos de balonmano.* 2010.

Pavón, Juan. *Introducción al lenguajes java.* 2013.

Pressman, Roger S. *Ingeniería del software. Un enfoque práctico.* 2010.

Real Federación Española de Balonmano. *Handball. Reglas de juego.* 2005.

RIcardo Sanchez, Jairo Hechevarria. Validacion de escalas. [En línea] 2004. www.google.com.

Rio, Sergio Sánchez. *Metodologías de Análisis y Diseño. Modelo conceptual.* 2011.

Rocha, Marcelo. *Modelado de datos.* 2011.

Rodríguez, Manuel José García. *Estudio comparativo entre las metodologías ágiles y las metodologías tradicionales para la gestión de proyectos de software.* 2015.

Ruiz, Francisco y Lopez, Patricia. *INGENIERÍA DEL SOFTWARE I. Tema11.Arquitectura Lógica del Sistema.* 2015.

Sánchez, Tamara Rodríguez. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.

Sierra, Antonio. *UML (Unified Medling Language) Lenguaje Unificado de Modelado.* 2013.

Sommerville, Ian.*Ingeniería de software. Séptima edición.* 2005.

Sparx System. Sparx System. [En línea] 2018. [Citado el: 20 de marzo de 2018.] http://www.sparxsystems.com.ar/about.html.

Técnicas de programación . **Sánchez, Esther Guerra.** 2009.

Trellini, Ariel. *Arquitectura y diseño de sistemas.* 2015.

Universidad del Valle. *Patrones GoF (Gang of Four).* 2008.

Valdés, David. *Aplicación para la gestión de perfiles en sistemas operativos Android.* 2016.

ANEXOS

Anexo 1. Descripción de los CUS

Tabla 12. CUS Gestionar Torneos

Nombre	Gestionar Torneo	
Objetivo	Gestionar los torneos	
Actor	Usuario	
Resumen	El caso de uso inicia cuando el usuario desea insertar, listar, editar, ver detalles o eliminar un torneo	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El usuario selecciona o crea un torneo en la vista principal Torneo	
Postcondiciones	Se inserta, lista, edita o elimina los torneos	
Flujo de eventos		
Flujo básico Gestionar torneos		
Actor		**Sistema**
Indica que desea insertar, listar, editar, ver detalles o eliminar un torneo		Si el usuario decide listar los torneos (ver sección 1. Listar torneos) Si el usuario decide insertar un nuevo torneo (ver sección 2. Insertar torneo) Si el usuario decide editar algún torneo (ver sección 3. Editar torneo) Si el usuario decide seleccionar un torneo (ver sección 4. Mostrar detalles torneo) Si el usuario decide eliminar torneo (ver sección 5. Eliminar torneo)
Sección 1 Listar torneo		
Flujo básico Listar torneo		
Actor		**Sistema**
1	Al ir a la vista Torneo en la pantalla	

	principal	
2		Muestra una interfaz con una lista de todos los torneos creados Las opciones que se brindan son: -insertar un torneo (ver la sección 2 Insertar Torneo) -editar un torneo (ver la sección 3 Editar Torneo) -mostrar detalles de un torneo (ver la sección 4 Mostrar detalles torneo) -eliminar torneo (ver la sección 5 Eliminar Torneo)
Sección 2 Insertar torneo		
Flujo básico Insertar torneo		
Actor		**Sistema**
1	Selecciona crear un torneo	
2		Muestra la interfaz Insertar torneo que permite introducir los valores referentes al torneo a insertar: -nombre -duración de los partidos en el torneo -rama (F o M) -categoría
3	Selecciona el botón Aceptar	
4		Valida los valores introducidos del torneo y lo muestra en esa misma pantalla, al cerrar la interfaz Insertar torneo
Flujo Alterno Insertar torneo		
Actor		**Sistema**
1	En caso de hacer clic en el botón Aceptar y haber dejado algún campo en blanco	
2		El sistema muestra el mensaje de error “Todos los

		campos son obligatorios”
Sección 3 Editar torneo		
Flujo básico Editar torneo		
Actor		**Sistema**
1	Selecciona un torneo	
2		El sistema muestra los equipos creados en ese torneo
3	Selecciona la opción que permite editar el torneo	
4		Muestra la interfaz Editar torneo que permite editar los valores referentes al torneo: -nombre -duración de los partidos en el torneo -rama (F o M) -categoría
3	Selecciona el botón Aceptar	
4		Valida los nuevos valores introducidos del torneo y muestra el torneo con sus nuevos valores
Flujo alterno Editar torneo		
Actor		**Sistema**
1	En caso de dar clic en el botón Aceptar y haber dejado algún campo en blanco	
2		El sistema muestra el mensaje de error “Todos los campos son obligatorios”
Sección 4 Mostrar detalles torneo		
Flujo básico Mostrar detalles torneo		
Actor		**Sistema**
1	Selecciona en la vista principal un	

	torneo previamente creado	
2		Muestra los resultados de los equipos y de los partidos jugados en el torneo
Sección 5 Eliminar torneo		
Flujo básico Eliminar torneo		
Actor		**Sistema**
1	Selecciona la opción que permite eliminar torneo	
2		Muestra un mensaje de información: “Este torneo se eliminará”
3	Selecciona el botón Confirmar	
4		Elimina el torneo seleccionado y actualiza la lista
Flujo alterno Eliminar torneo		
Actor		
1	Una vez seleccionado el torneo presiona el botón cancelar	
2		Cancela la petición y muestra nuevamente el listado de los torneos
Relaciones	**CU incluidos**	**-**
Prototipo elemental de interfaz gráfica de usuario		

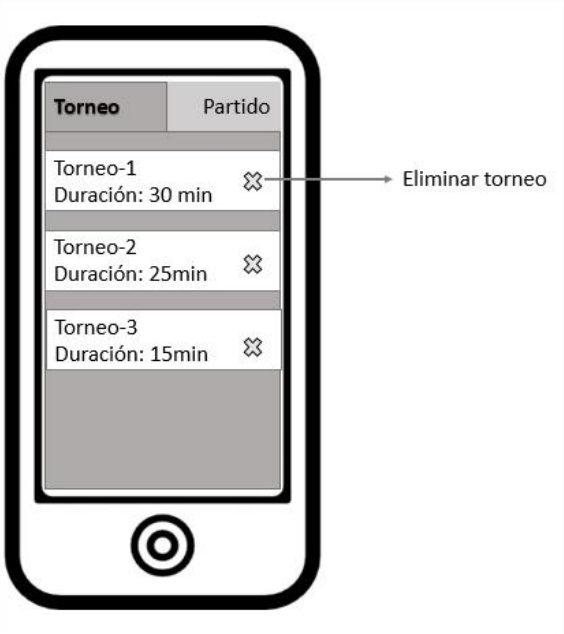

Torneo
Partido
Torneo-1
Duración: 30 min
Torneo-2
Duración: 25min
Torneo-3
Duración: 15min
Eliminar torneo

Tabla 13. CUS Gestionar Equipos

Nombre	Gestionar Equipo
Objetivo	Gestionar los equipos
Actor	Usuario
Resumen	El caso de uso inicia cuando el usuario desea insertar, listar, editar, ver detalles o eliminar un equipo
Complejidad	Media
Prioridad	Media
Precondiciones	El usuario selecciona o crea un equipo dentro de la vista Torneo
Postcondiciones	Se inserta, lista, edita o elimina los equipos

Flujo de eventos	
Flujo básico Gestionar equipos	
Actor	**Sistema**
Indica que desea insertar, listar, editar, ver detalles o eliminar un equipo	Si el usuario decide listar los equipos (ver sección 1. Listar equipos) Si el usuario decide insertar un nuevo equipo (ver sección 2. Insertar equipo) Si el usuario decide editar algún equipo (ver sección 3. Editar equipo) Si el usuario decide eliminar equipo (ver sección 4. Eliminar equipo) Si el usuario selecciona un equipo (ver sección 5. Mostrar detalles equipo)

Sección 1: Listar equipo		
Flujo básico Listar equipo		
Actor		**Sistema**
1	Selecciona un torneo	
2		Muestra una interfaz con una lista de todos los equipos existentes en ese torneo.

		Las opciones que se brindan son: -insertar un equipo (ver la sección 2 Insertar Equipo) -editar un equipo (ver la sección 3 Editar Equipo) -eliminar equipo (ver la sección 4 Eliminar Equipo)
Sección 2 Insertar equipo		
Flujo básico Insertar equipo		
Actor		**Sistema**
1	Selecciona un equipo	
2		Muestra la interfaz Insertar equipo que permite introducir el nombre del equipo a insertar
3	Selecciona el botón Aceptar	
4		Valida el campo nombre del equipo y lo muestra en esa misma pantalla, al cerrar la interfaz Insertar equipo
Flujo Alterno Insertar equipo		
Actor		**Sistema**
1	En caso de hacer clic en el botón Aceptar y haber dejado el campo nombre en blanco	
2		El sistema muestra el mensaje de error “Todos los campos son obligatorios”
Sección 3 Editar equipo		
Flujo básico Editar equipo		
Actor		**Sistema**
1	Selecciona la opción que permite editar el equipo	
2		Muestra la interfaz Editar equipo que permite editar el nombre del equipo
3	Selecciona el botón Aceptar	

4		Valida el campo nombre del equipo y muestra el equipo con el nuevo valor
Flujo alterno Editar equipo		
Actor		**Sistema**
1	En caso de dar clic en el botón Aceptar y haber dejado el campo nombre en blanco	
2		El sistema muestra el mensaje de error “Todos los campos son obligatorios”
Sección 4 Eliminar equipo		
Flujo básico Eliminar equipo		
Actor		**Sistema**
1	Selecciona la opción que permite eliminar equipo	
2		Muestra un mensaje de información: “Este equipo se eliminará”.
3	Selecciona el botón Confirmar	
4		Elimina el equipo seleccionado y actualiza la lista.
Flujo alterno Eliminar equipo		
Actor		
1	Una vez seleccionado el equipo presiona el botón cancelar	
2		Cancela la petición y muestra nuevamente el listado de los equipos.
Sección 5 Mostrar detalles equipo		
Flujo básico Editar equipo		
Actor		**Sistema**
1	Selecciona un equipo	

2		Muestra los jugadores insertados en ese equipo
Relaciones	**CU incluidos**	**-**
Prototipo elemental de interfaz gráfica de usuario		

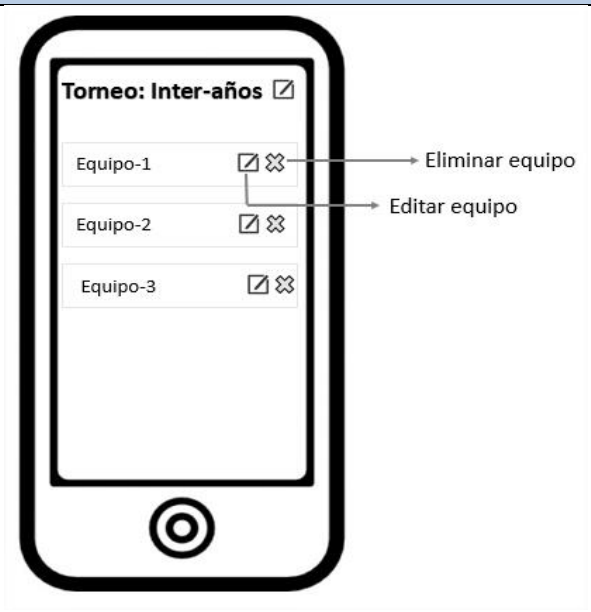
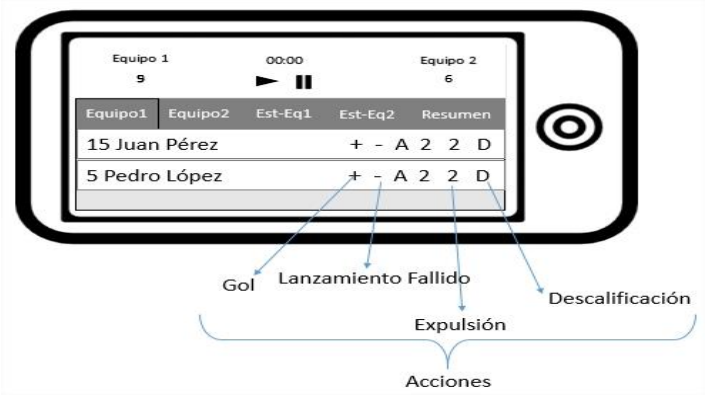


Tabla 14. CUS Registrar Partido

Nombre	Crear nuevo Partido	
Objetivo	Registrar un partido	
Actores	Usuario	
Resumen	El caso de uso inicia cuando el usuario desea iniciar un nuevo partido	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El usuario selecciona la opción Nuevo Partido	
Postcondiciones	Se inserta y registran las acciones de un partido	
Flujo de eventos		
Flujo básico Iniciar Partido		
Actor		**Sistema**
Indica que desea iniciar un nuevo partido		Si el usuario decide iniciar un nuevo partido. Ver Sección 1. Crear nuevo partido Si el usuario decide insertar acción del partido. Ver Sección 2. Insertar acción Si el usuario decide ver las acciones del partido. Ver Sección 3. Registro de acciones del partido. Si el usuario decide ver las estadísticas del partido. Ver Sección 4. Estadísticas.
Sección 1 Crear nuevo partido		
Flujo básico Crear nuevo partido		
Actor		**Sistema**
1	Selecciona la opción Nuevo Partido del menú principal o dentro de un torneo	
2		Muestra la interfaz Insertar Partido que permite introducir los valores referentes al partido a insertar: -fecha

		-equipo1 -equipo2 -grupo
3	Selecciona el botón Aceptar	
		Valida los valores introducidos del partido y muestra una interfaz con el listado de los jugadores de los equipos que juegan, el marcador de goles, cronómetro del partido, cronómetro de amonestaciones y los quipos que se enfrentan. Las opciones que se brindan son: Insertar acción (ver sección 2: Insertar Acción.) Registro de estadísticas (ver Sección 3: Registro de Estadísticas) Resumen general del partido (ver Sección 4: Resumen general del partido) Exportar planilla ver Sección 5: Exportar planilla
Sección 2 Insertar acción		
Flujo básico Insertar acción		
Actor		**Sistema**
1	Selecciona el jugador que realizó la acción a insertar.	
2		Inserta la acción y la muestra en el registro de estadísticas del equipo al que pertenece el jugador y en el resumen general del partido.
Sección 3 Registrar estadísticas		
Flujo básico Registro de estadísticas del partido		
Actor		**Sistema**
1	Selecciona la vista estadísticas del equipo que desee	

2		Muestra un listado de estadísticas calculadas a partir de las acciones insertadas (ver Sección 2: Insertar Acción) del partido.
Sección 4 Generar resumen		
Flujo básico Estadísticas		
Actor		**Sistema**
1	Selecciona la vista Resumen	
2		Muestras un resumen con todas las acciones insertadas (ver Sección 2: Insertar Acción)
Sección 5 Exportar planilla		
Flujo básico Estadísticas		
Actor		**Sistema**
1	Una vez finalizado el partido, presiona el botón exportar	
2		Exporta la planilla oficial del partido seleccionado a PDF
Relaciones	**CU incluidos**	**-**
Prototipo elemental de interfaz gráfica de usuario		



Anexo 2. Casos de prueba

Tabla 15. Caso de prueba Insertar Equipo

Escenario	Descripción	Nombre	Respuesta del sistema	Flujo central
EC 3.1 Insertar equipo	Permite al usuario crear un nuevo equipo.	V	Inserta en la base de datos el equipo creado por el usuario, para ello debe llenar el campo Nombre.	Seleccionar del menú principal un torneo Selecciona el botón Crear Equipo ubicado en la esquina inferior derecha Introduce los datos del equipo Selecciona el botón Aceptar
EC 3.2 Insertar equipo	Validar que no existen campos vacíos	I vacío	En el caso en que el campo Nombre esté vacío muestra el mensaje “Debe introducir el nombre del equipo”.	Seleccionar del menú principal un torneo Selecciona el botón Crear Equipo ubicado en la esquina inferior derecha Introduce los datos del equipo Selecciona el botón Aceptar
EC 3.3 Insertar equipo	Validar que no existen campos inválidos	I F@cultad	No permite introducir caracteres inválidos.	Seleccionar del menú principal un torneo Selecciona el botón Crear Equipo ubicado en la esquina inferior derecha Introduce los datos del equipo Selecciona el botón Aceptar

Tabla 16. Caso de prueba Insertar Jugador

Escenario	Descripción	Nombre	Apellido	Dorsal	Respuesta del sistema	Flujo central
EC 4.1 Insertar jugador	Permite al usuario crear un nuevo jugador.	V	V	V	Inserta en la base de datos el jugador creado por el usuario, para ello debe llenar los campos Nombre Apellido Dorsal Muestra el mensaje "Jugador Insertado"	Seleccionar del menú principal un torneo Selecciona dentro de un torneo un equipo Dentro del equipo selecciona el botón Crear Jugador ubicado en la esquina inferior derecha Introduce los datos del jugador Selecciona el botón Aceptar
EC 4.2 Insertar Jugador		I (vacío)	V	V	En el caso de que los campos: Nombre, Apellido o Dorsal estén vacíos, el sistema muestra un mensaje "Todos los campos son obligatorios"	Seleccionar del menú principal un torneo Selecciona dentro de un torneo un equipo Dentro del equipo selecciona el botón Crear Jugador ubicado en la esquina inferior derecha Introduce los datos del jugador Selecciona el botón Aceptar
		V	I(vacío)	V		
		V	V	I(vacío)		

EC 4.3 Insertar jugador	Validar que no existen campos inválidos	I Fr@nk	V	V	No permite introducir caracteres inválidos	Seleccionar del menú principal un torneo Selecciona dentro de un torneo un equipo Dentro del equipo selecciona el botón Crear Jugador ubicado en la esquina inferior derecha Introduce los datos del jugador Selecciona el botón Aceptar
		V	I G4rc1A	V		

Tabla 17. Caso de prueba Insertar Torneo

Escenario	**Descripción**	**Nombre**	**Duración de Partidos**	**Respuesta del sistema**	**Flujo central**

EC 5.1 Insertar torneo	Permite al usuario crear un nuevo torneo.	V	V	Inserta en la base de datos el torneo creado por el usuario, para ello debe llenar los campos Nombre Duración de Partidos Muestra el mensaje "Torneo Insertado"	Seleccionar en la pantalla principal, la vista torneo. Seleccionar el botón Crear Torneo ubicado en la esquina inferior derecha. Insertar los datos del torneo. Selecciona el botón aceptar.
EC 5.2 Insertar torneo	Validar que no existen campos vacíos	I vacío	V	Muestra el mensaje "Todos los campos son obligatorios"	Seleccionar en la pantalla principal, la vista torneo. Seleccionar el botón Crear Torneo ubicado en la esquina inferior derecha. Insertar los datos del torneo.
		V	I Vacío		

					Selecciona el botón aceptar.
EC 5.3 Insertar torneo	Validar que no existen campos inválidos	I Ju3go$	V	No permite introducir estos caracteres. La duración de un partido debe ser un valor entero.	Seleccionar en la pantalla principal, la vista torneo. Seleccionar el botón Crear Torneo ubicado en la esquina inferior derecha. Insertar los datos del torneo. Selecciona el botón aceptar.
		V	I -1		

Anexo3. Descripción de las tablas de la base de datos

Tabla 18. Descripción de la tabla Jugador

Jugador		
Almacena toda la información referente a un jugador		
Atributo	**Tipo**	**Descripción**
id	integer	Etiqueta única que identifica al objeto jugador en la tabla
nombre	varchar	Nombres del jugador
apellido	varchar	Apellidos del jugador
dorsal	integer	Dorsal del jugador
equipo_id	integer	Atributo que identifica el equipo al que pertenece el jugador

Tabla 19. Descripción de la tabla Acción-Partido

Acción_Partido		
Almacena la información referente a una acción de un jugador en un partido		
Atributo	**Tipo**	**Descripción**
id	integer	Etiqueta única que identifica la acción del jugador en el partido
tipo de acción	integer	Tipo de acción (gol, lanzamiento fallido, falta)
minuto	integer	Minuto en que sucedió la acción
partido_id	integer	Atributo que identifica el partido en el que se hizo la acción
jugador_id	integer	Atributo que identifica el jugador que realizó la acción

Tabla 20. Descripción de la tabla Equipo

Equipo		
Almacena la información referente a un equipo		
Atributo	**Tipo**	**Descripción**
id	integer	Etiqueta única que identifica al objeto equipo en la tabla
nombre		Nombre del equipo
grupo	varchar	Grupo al que pertenece el equipo

Tabla 21. Descripción de la tabla Torneo

Torneo		
Almacena la información referente a un torneo		
Atributo	**Tipo**	**Descripción**
id	integer	Etiqueta única que identifica al objeto en la tabla
nombre	varchar	Nombre del torneo
duración_partido	integer	Duración de los partidos en ese torneo
rama	varchar	Género de los jugadores
categoría	varchar	Categoría del torneo

Tabla 22. Descripción de la tabla Partido

Partido		
Almacena la información referente a un partido		
Atributo	**Tipo**	**Descripción**
id	integer	Etiqueta única que identifica al objeto partido en la tabla

equipo_id1	integer	Etiqueta única que identifica a un equipo de los que participa en el partido
equipo_id2	integer	Etiqueta única que identifica a un equipo de los que participa en el partido
fecha	varchar	Fecha en que se realiza un partido
cant_goles_Eq1	integer	Cantidad de goles del equipo 1
cant_goles_Eq2	integer	Cantidad de goles del equipo 2
torneo_id	integer	Atributo que identifica el torneo al que pertenece el partido
hora_i	varchar	Hora en que se inicia el partido
hora_f	varchar	Hora en que finaliza el partido

Anexo 4. Pruebas de rendimiento

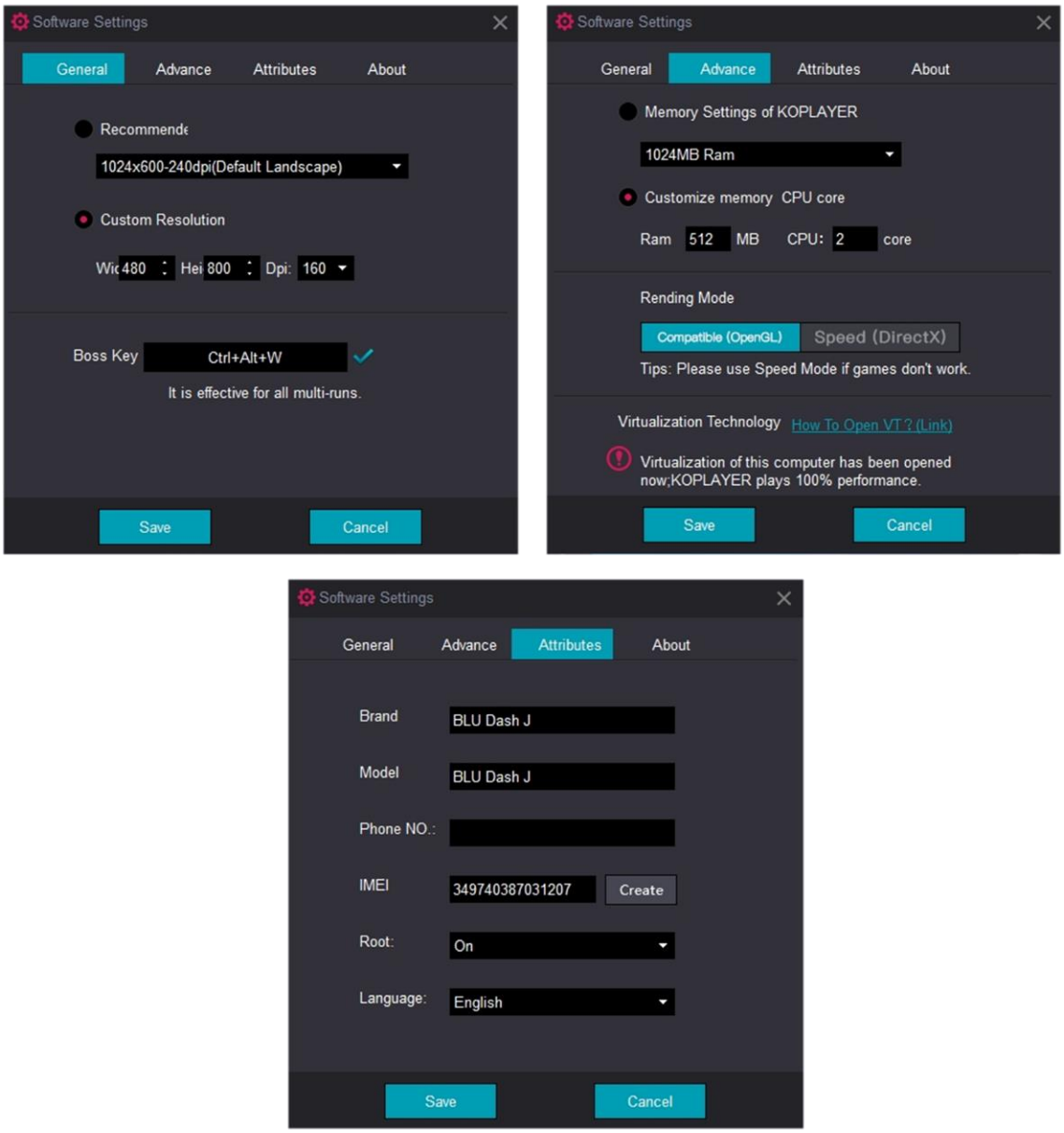


Figura 24. Configuración del emulador Koplayer para la prueba de rendimiento realizada a la aplicación

Figura 25. Visualización de la aplicación en el emulador Koplayer