Universidad de las Ciencias Informáticas

Facultad 2

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

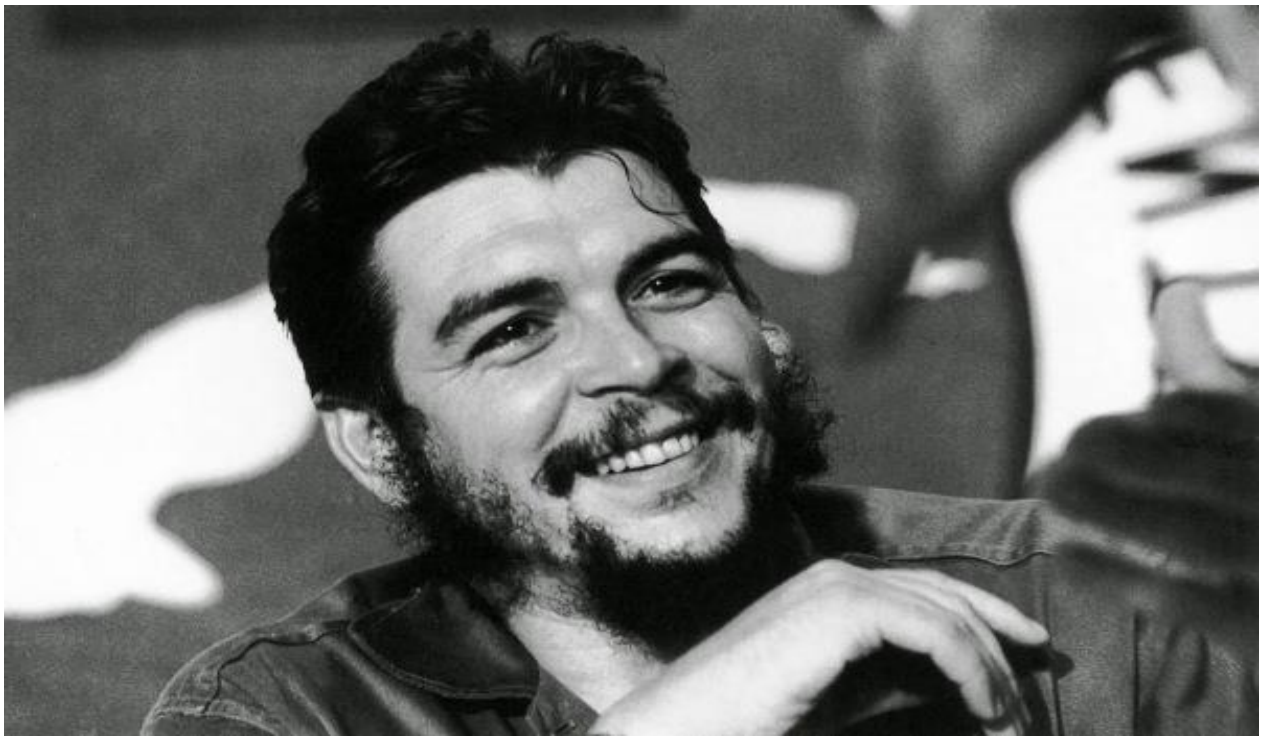
Sistema para la gestión de acciones tácticas significativas de los equipos de baloncesto

Autor: Jessica Pérez Almanza

Tutores: Ing. Vladimir Milián Núñez

Ing. Roberto Antonio Infante Milanés

La Habana, 2018



"Lo fundamental es que seamos capaces de hacer cada día algo nuevo, que perfeccione lo que hicimos el día anterior..."

Che

DECLARACIÓN DE AUTORÍA

Declaro ser autora del presente trabajo de diploma titulado: “Sistema para la gestión de acciones tácticas significativas de los equipos de baloncesto” y concedo a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo. Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____.

_____

Jessica Pérez Almanza

Autor

_____ _____

Ing. Vladimir Milián Núñez

Tutor

Ing. Roberto Antonio Infante Milanés

Tutor

DEDICATORIA

Este trabajo lo dedico a las dos personas más importantes en mi vida**, MIS PADRES.**

AGRADECIMIENTOS

A mis padres por todo el amor que me han brindado, mi madre por guiarme siempre por el buen camino, estar para mí en los momentos más difíciles, por haberme convertido en la persona que soy hoy, gracias a su excelente educación y por luchar cada día para brindarme siempre lo mejor. A mi padre por el sacrificio todos estos años para que su niña siempre este feliz, por su apoyo incondicional y por ser un excelente padre como los hay pocos en este mundo, los amo.

A mis tías por su apoyo en todo momento, su amor incondicional como si fuera una hija más.

A mi abuela por todo el cariño que me ha dado.

A mis primos que los adoro, por todo su amor y siempre estar presentes para su única prima.

A mi Joe por su enorme paciencia, por todo ese amor que me da, por estar siempre conmigo en las buenas y las malas, por toda su ayuda y comprensión, por todas las cosas lindas que hace por mi TE QUIERO.

A Dayme y Oscar por su cariño y apoyo.

A mi amiga Eliani, por ser como una hermana en estos 8 años, te quiero.

A Eileen por los días de tesis, de fiestas, de eventos, por estar siempre presente en mis momentos más tristes y en los más alegres como hoy. Te quiero

A Darío por toda su ayuda en el desarrollo de esta tesis.

A las amistades que hice durante estos 5 años, en especial a Pablo.

A mis compañeros de aula, por esos 5 años que compartimos juntos.

A mis compañeras de apartamento, aunque sea muy poco el tiempo que compartimos la pase genial con ustedes.

A mis tutores por toda su ayuda y dedicación.

A los profesores, en especial a mi profe de matemática Abel.

A todo aquel que contribuyó con mi formación y que de una manera u otra me ha ayudado a lograr este sueño.

Resumen

En el Baloncesto se utilizan las estadísticas como instrumento que posibilita al entrenador tomar decisiones durante el desarrollo de un partido, y para analizar con posterioridad la actuación del equipo. La presente investigación tiene como objetivo el desarrollo de un sistema para dispositivos móviles con sistema operativo Android, nombrada ManagerBasket. El sistema permite gestionar las acciones tácticas significativas de los equipos de baloncesto. ManagerBasket tiene como finalidad, brindarle al entrenador estadísticas claras de los jugadores, partidos, equipos y torneos, aportando información necesaria para un posterior análisis y toma de decisiones. El sistema está implementado en el lenguaje Java, utilizando el entorno de desarrollo Android Studio y como gestor de base de datos SQlite. Además, se tuvieron en cuenta todos los pasos y métodos que ofrece la metodología AUP-UCI. Luego de creadas las bases, se realiza la implementación de las funcionalidades obteniendo como resultados: una aplicación que agiliza la recogida de datos durante los partidos de baloncesto y es capaz de calcular de manera automática las estadísticas necesarias facilitando el trabajo a los entrenadores.

Palabras clave: dispositivos móviles, Android, equipos de baloncesto, estadísticas, toma de decisiones.

Abstract

In basketball statistics are used as an instrument that facilitates the process of decision making to the coach during the course of a game, and to analyze later the performance of the team. The fallowing research haves as an objective to develop a system for mobile devices with the Android operative system, named ManagerBasket. The system allows the management of meaningful tactical actions for basketball teams. ManagerBasket haves as a goal, to give the manager clear statistics of the players, games, teams and tournaments, adding necessary information for a later analysis and decision making. The system is implemented in the Java programing language, using the Integrated development environment Android Studio, and as a database gestor SQlite. Besides, it was taken in to account every step and method offered by the methodology AUP-UCI. After the creation of the bases, an implementation of the main functionalities is made, obtaining as a result: A app that makes an agile data collection, during basketball games, and is able to calculate on an automatic fashion the statistics needed, facilitating the work of the coaches.

Key words: mobile devices, Android, basketball teams, statistics, decision making.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

Las Tecnologías de la Información y la Comunicación (TIC) se han convertido en un recurso casi imprescindible para el desarrollo de la actividad social de manera general. El uso de la tecnología en el quehacer diario de las personas, tanto en lo laboral como en lo personal, ha estado condicionado por el propio desarrollo tecnológico. La expansión de las TIC en todos los ámbitos de la sociedad, se ha producido a gran velocidad y es un proceso que continua sin cesar, acrecentando cada vez más los aportes tecnológicos. (Diaz, y otros, 2011)

El surgimiento de la telefonía móvil es uno de los aportes de mayor impacto en la sociedad en los últimos años. Los teléfonos móviles han adquirido funcionalidades que van más allá de llamar o de enviar mensajes de texto y han incorporado múltiples acciones que permiten la interacción con los usuarios. Estos dispositivos reciben el nombre de teléfonos inteligentes o “smartphones” donde su funcionamiento y prestaciones se acercan más a pequeños ordenadores.

Actualmente, los teléfonos inteligentes, han llegado a convertirse en una herramienta básica para los usuarios, gracias a su capacidad y poder de procesamiento, calidad de la conectividad, movilidad, la disminución de costos, rapidez, aumento de los servicios y calidad de los mismos (Gil, y otros, 2017). Para su funcionamiento, estos dispositivos requieren de un sistema operativo (SO), entre los que se destacan Android, iOS y Windows Phone. 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 hace 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)

Android es un sistema operativo basado en el núcleo de Linux que abarca una amplia gama de dispositivos móviles y sus proyectos de desarrollo correspondientes liderados por Google. Es compatible con una gran variedad de hardware en el mercado (tablets y dispositivos celulares de marcas como: Motorola, Samsung, ZTE, Huawei, Ericsson por nombrar algunas) permitiendo al usuario elegir el dispositivo que mejor se ajusta a sus necesidades (Malave, y otros, 2011). Esto demuestra que Android se afianza como uno de los sistemas operativos más utilizados en dispositivos móviles.

Los smartphones están presentes en casi todo el acontecer científico, político, económico, social, cultural, deportivo, de la sociedad actual. En los últimos años se ha producido un avance considerable en el deporte, gracias a las tecnologías digitales y a las herramientas informáticas. Dichos avances permiten el desarrollo de aplicaciones que favorecen los procesos de registros y la interpretación de los datos, ofreciéndoles a los deportistas y entrenadores una retroalimentación inmediata de su rendimiento. (Carranza, y otros, 2011)

En el deporte, con el fin de mejorar el rendimiento de los jugadores y los equipos, se desarrollan diferentes técnicas y tácticas para el entrenamiento deportivo, utilizando los avances que ofrece la ciencia y la tecnología. Uno de esos deportes es el baloncesto, donde para formular cualquier nueva estrategia, se necesita un basamento estadístico del comportamiento de los jugadores y el equipo en cada una de las fases del juego. Los especialistas destacan las posibilidades que ofrece una planilla de evaluación-observación sistemática como técnica de medida para el análisis de la cuantificación y la evaluación táctica ofensiva, tanto individual como colectiva; permitiendo el control, análisis y evaluación de las numerosas variables que se corresponden con las acciones tácticas de un juego.

En la Universidad de las Ciencias Informáticas (UCI) el baloncesto es un deporte de alta masividad y de grandes atractivos para la comunidad universitaria. La utilización de las tecnologías en el baloncesto, facilita el trabajo tanto de los entrenadores como de los propios jugadores. Un estudio realizado en dicha universidad, tomando como muestra los trabajadores de la Dirección de Deportes, reveló que el 94 % poseen teléfonos inteligentes con Android (ver anexo 1).

Durante un torneo de baloncesto, la recogida de información acerca de las acciones tácticas, se torna compleja. En ocasiones pueden cometerse errores, debido a que es un deporte muy dinámico y existe un corto período de tiempo entre las anotaciones, cuando las acciones realizadas son continuas. Además, la recogida de forma manuscrita puede provocar pérdidas o deterioro en los documentos y su organización. Es por ello que los entrenadores han apostado por las tecnologías para facilitar el proceso de recogida de acciones tácticas en los torneos de baloncesto.

A partir de la problemática anteriormente expuesta se identifica el siguiente **problema:** ¿Cómo facilitar a los entrenadores la gestión de acciones tácticas significativas en los partidos de baloncesto para su posterior análisis y toma de decisiones?

Se identifica como **objeto de estudio** para esta investigación la gestión de acciones tácticas significativas en el baloncesto y como **campo de acción** el proceso de gestión de información tácticas significativas en los partidos de baloncesto mediante dispositivos móviles.

Para darle solución al problema planteado se propone como **objetivo general:** desarrollar un sistema informático para la gestión de las acciones tácticas significativas de los equipos de baloncesto durante un partido, para el sistema operativo Android

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

1. Análisis de la bibliografía técnica del baloncesto, para comprender las reglas del deporte y de las competiciones, así como las acciones tácticas significativas de un partido.

2. Análisis de sistemas existentes que permitan la recogida de acciones tácticas significativas en partidos de baloncesto, para identificar características o funcionalidades similares a las del sistema propuesto.
3. Análisis y selección de la metodología, las herramientas, las tecnologías y el lenguaje a utilizar para el desarrollo de la solución propuesta.
4. Especificación de los requisitos funcionales y no funcionales de la solución propuesta para establecer lo que el software debe hacer y bajo qué circunstancias debe hacerlo.
5. Implementación de la aplicación propuesta para dispositivos que operan con sistema operativo Android.
6. Validación de la solución mediante los métodos de prueba para comprobar el correcto funcionamiento de la misma.

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

Métodos teóricos:

- Análisis-Síntesis: permite analizar la información acerca de las metodologías, las tecnologías y herramientas a utilizar en el desarrollo del sistema, seleccionando los principales elementos y características. Además, posibilita el estudio de los documentos y la extracción de los aspectos más importantes sobre el registro de las acciones en un juego de baloncesto.
- Modelación: permite reflejar la estructura, relaciones y características de la solución a través de diagramas, facilitando también el diseño y la comprensión de las clases necesarias para la implementación del sistema.

Métodos empíricos:

- Entrevista (de forma no estructurada): para conocer las particularidades de la práctica de baloncesto en la UCI, el proceso de recogida de acciones tácticas significativas y las propiedades fundamentales de los recursos tecnológicos que poseen los entrenadores.

El presente documento está estructurado en introducción, desarrollo, conclusiones, bibliografía y anexos. El contenido del desarrollo se divide en 3 capítulos:

Capítulo 1. Fundamentos teóricos de la gestión de información de acciones tácticas significativas en el deporte: se realiza un estudio sobre los sistemas existentes en el mundo, enfocados en la recogida de resultados estadísticos en partidos de baloncesto. Se definen las herramientas, tecnologías y lenguajes de programación, así como la metodología para el desarrollo de la aplicación.

Capítulo 2. Sistema informático para la gestión de acciones tácticas significativas de los equipos de baloncesto: se presenta la propuesta de solución y se realiza el modelado de los procesos de negocio. Se determinan los requisitos funcionales y no funcionales que debe cumplir el sistema. Además, se define la arquitectura, el patrón arquitectónico de la aplicación y los patrones de diseño utilizados.

Capítulo 3. Implementación y validación del sistema: se describe el proceso de implementación, mediante los diagramas de componentes y de despliegue. 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 tácticas significativas en el deporte

El objetivo de este capítulo, es abordar los aspectos teóricos que servirán de soporte para la implementación del sistema. Se analizan soluciones similares vinculadas a la gestión de información estadística en los partidos de baloncesto, profundizando en las características de cada una. Se hace un estudio para definir la metodología, tecnologías, herramientas y lenguajes de programación a utilizar en la implementación de la solución propuesta.

1.1 Baloncesto

El baloncesto es un deporte de dos equipos, formado por 12 jugadores como máximo cada uno; 5 formarán el quinteto final y los restantes serán los suplentes. La función de este deporte es introducir el balón en una canasta o cesta. Los partidos están compuestos por cuatro períodos de 10 minutos cada uno. Si el partido finaliza en empate, se juegan tantas prórrogas de 5 minutos que sean necesarias hasta que un equipo gane, ya que no existe el empate ni las tandas de tiros especiales (equivalente a los penales en otros deportes). (Olivera, y otros, 1992; Federación Internacional de Baloncesto (FIBA), 2016)

Durante un partido, el entrenador es el encargado de cambiar a los jugadores y de llevar el control de las acciones tácticas significativas de su equipo. Una acción táctica es un movimiento realizado por un jugador ya sea en ataque, defensa o contraataque. Algunas de las acciones tácticas más importantes que son recogidas durante los partidos son las canastas (tiros libres, dos y tres puntos), las asistencias, los fallos, las pérdidas y las recuperaciones de rebotes, entre otras (Goldstein, 2011; Federación Internacional de Baloncesto (FIBA), 2016). Los entrenadores recogen estás acciones para evaluar el comportamiento de sus jugadores en cada una de las fases del juego, así como el comportamiento general del equipo.

Actualmente existe una adaptación de este deporte llamada baloncesto 3x3, básicamente contempla las mismas reglas, aunque se han reducido las formalidades respecto a lo que se puede y lo que no se puede hacer. En esta variante solo se utiliza la mitad de la cancha y cada canasta vale un punto, excepto aquellas conseguidas desde más allá de la línea de triple, que valdrán dos. Los partidos de 3x3 tienen una duración de 10 minutos y se juega a 21 puntos con 2 de ventaja. (García, 2013)

En Cuba se desarrolla la Liga Superior de Baloncesto utilizando la variante uno de este deporte. Para el desarrollo de la presente investigación se analizan las reglas del mismo, debido a que el baloncesto 3x3 actualmente no está incorporado en todos los eventos deportivos. (García, 2013)

1.2 Acciones tácticas significativas en el baloncesto

En el Baloncesto al igual que otros deportes, la gestión de las acciones tácticas significativas se utilizan como instrumento para medir el rendimiento de los jugadores y del equipo en una competición y/o en la etapa de preparación. Los entrenadores realizan un análisis de las acciones recogidas durante un partido a través de diversos indicadores. Estos indicadores le permiten al entrenador llevar un control de su equipo, facilitando la toma de decisiones y el diseño de nuevas estrategias de juego. A continuación se muestran un conjunto de indicadores utilizados por entrenadores y técnicos, en la evaluación de los jugadores durante la actividad competitiva. (Sampaio, y otros, 2002)

1. Tiros de campo

Expresa la eficacia de un jugador en los tiros de campo y también el número de veces que lo consigue.

$PT = \frac{PA-TLA}{TCI}$, donde PT son los puntos por tiros, PA los puntos anotados, TLA los tiros libres anotados y TCI los tiros de campo intentados.

2. Tiros libres

La evaluación de los tiros libres se basa en el porcentaje de tiros libres y el número de tiros libres convertidos.

$ETL = \%TL * TLC$, donde ETL es la eficiencia en el tiro libre, %TL representa el porcentaje de tiros libres y TLC los tiros libres convertidos.

3. Faltas

En el apartado de las faltas simplemente se calcula la relación entre las faltas recibidas y las cometidas.

$F = \frac{FR}{FC}$, donde F es la relación de las faltas, FR las faltas recibidas y FC las faltas cometidas.

4. Control del balón

Está determinado por la relación entre las asistencias y las pérdidas de balón.

$CB = \frac{A}{PB}$, donde CB es el control del balón, A las asistencias y PB las pérdidas de balón.

1.3 Análisis de otras soluciones existentes

Actualmente existen numerosas aplicaciones informáticas que permiten la gestión de información en los partidos de baloncesto. Antes de comenzar con el desarrollo de la solución propuesta, se analizaron las características de los sistemas similares para conocer si alguno constituye una solución al problema planteado en la presente investigación. Para realizar esta valoración se tuvieron en cuenta varios aspectos, como: la gestión de jugadores, equipos y torneos; el cálculo automático de indicadores; código abierto; registro en tiempo real de acciones; la disponibilidad de una pizarra táctica; y que hayan sido desarrollados para el sistema operativo Android.

A continuación, se describen aplicaciones a nivel internacional que presentan objetivos similares a las necesidades del cliente.

Baloncesto Five On Court

Aplicación con sistema operativo Android, que obtiene las estadísticas de equipos de baloncesto, contando con la mayoría de eventos de un partido. Permite obtener las estadísticas en tiempo real de cada jugador. Posee un cronometro para llevar el tiempo del juego y capaz de parar el marcador con solo un clic. El sistema se presenta en tres lenguajes, inglés, español y catalán. Dispone de una pizarra táctica para el diseñar estrategias de jugadas. (Google Play, 2017)

Baloncesto Five On Court posibilita la recogida de una gran cantidad de datos, utilizando en este proceso una interfaz amigable y muy intuitiva. El sistema no brinda algunas de las funcionalidades que se desean con la solución propuesta, como la gestión de torneos y la de equipos. Es un software gratuito y no es de código abierto por lo que imposibilita la reutilización del mismo.

BasketBloc

Aplicación diseñada para que los entrenadores o los miembros del cuerpo técnico, puedan controlar las estadísticas de los partidos de baloncesto de su equipo u otro equipo. Este programa también permite ajustarse a la normativa del Baloncesto en Silla de Ruedas (BSR). Compatible con Android 3.0 o superior y recomendado para dispositivos con pantallas de 4 pulgadas o superior. Los períodos por partidos se especifican, al igual que las faltas por jugador y la cantidad de períodos que reiniciaran las faltas de equipo. Además, contiene un cronometro para controlar el tiempo del partido. Es posible añadir estadísticas personalizadas, permite asignar imágenes y colores a los equipos y jugadores para mejor la identificación. (Google Play, 2017)

Esta aplicación presenta características muy útiles y algunos elementos de la interfaz, como la personificación de los equipos y jugadores, para la identificación de cada uno. Estos elementos pueden ser tomados como ideas en la implementación de la propuesta de solución. La aplicación posee la ventaja de ajustarse a la normativa del BSR, por tanto, puede ser utilizada en Juegos Paralímpicos. No presenta una pizarra táctica para la creación de estrategias, componente necesario para el trabajo de

los entrenadores. BasketBloc permite gestionar los jugadores, pero presenta como limitante que no gestiona los torneos ni los equipos. El sistema no cumple con todas las funcionalidades que se desean y no es de código abierto.

BasketStats Lite

BasketStats permite recoger, almacenar y agrupar estadísticas de uno o varios equipos de baloncesto, para cualquier categoría y edad. Modificando los ajustes se pueden crear torneos personalizados, definiendo la cantidad de cuartos, jugadores en pista, minutos por cuarto o el máximo de faltas permitidas. Gracias a esta información se realiza un seguimiento exhaustivo de los jugadores, sus porcentajes, posiciones de tiro preferidas, rendimiento según cuarto y minutos jugados, además del resto de estadísticas más comunes (asistencias, robos, faltas, recuperaciones, tapones, rebotes.), y posteriormente exportarlas en formato CSV. La aplicación está disponible para smartphones y tablets, en múltiples idiomas. (Google Play, 2017)

Es una aplicación muy completa con respecto a las necesidades del cliente, pero presenta la limitante de no tener pizarra táctica para dibujar las estrategias de juego. Este sistema permite modificar los colores de los botones, observar las estadísticas por juegos o por períodos y especificar el máximo de faltas. Define la ubicación de almacenamiento de las estadísticas, y permite cargar aquellas que hayan sido guardadas anteriormente. Es un software muy eficiente para los entrenadores por las variadas funcionalidades que posee, pero no es posible acceder a su código fuente.

OmniStats BasketBall

Aplicación desarrollada por Alberto Ortiz para llevar todas las estadísticas de un partido de baloncesto, de “forma ágil y fluida”, y que puede ser utilizada tanto por profesionales de este deporte -entrenadores o delegados-, como por aficionados. Posee memoria interna, lo que facilita realizar comparativas con estadísticas anteriores sin necesidad de conexión a internet. Destaca por su simplicidad, cercanía al usuario y control de las estadísticas, disponible por el momento solo para Android y con versiones en inglés, italiano, francés y portugués. Entre sus ventajas, destaca que es capaz de sacar medias y acumulados por partidos**,** temporadas**,** jugadores**,** etc., además de estadísticas de cualquier tiempo (simples, mini/plus), diagramas de tiro**,** valoraciones**,** jugada a jugada y todo ello compartirlo con quien se quiera. (Google Play, 2017)

OmniStats BasketBall usa el formato vertical para la toma de datos, excepto en la visualización de las estadísticas, en donde podrá usar el dispositivo en cualquier posición, además genera las mismas en tres formatos distintos para su comodidad, html, csv y pdf. En la pantalla de la aplicación aparecen dos variantes de recogida de datos: acciones de tiro, se pulsa el jugador, la zona de tiro y si anota o falla; y las acciones que no son de tiro, seleccionando el jugador y el tipo de acción realizada. Todas las acciones introducidas se graban en tiempo real en la base de datos de la aplicación, donde el usuario puede observar todas las acciones de los partidos que ha recogido, pudiendo hacer comparativas de

jugadores por partido. Además de todas estas características, posee una variedad de funcionalidades muy útiles como: (Google Play, 2017)

- Fluidez a la hora de introducir los datos al tener todas las acciones en la misma pantalla.
- Correo activo las 24 horas del día para solucionar cualquier consulta que pueda tener.
- Configuración del partido a la normativa que tenga su comunidad: períodos, faltas, tiempo, distancia de línea de triples (6,25 – 6,75 o campo de minibasket), acciones con continuación.
- Sin límite en la numeración de jugadores (00-99) ni cantidad de jugadores para un mismo equipo.
- Añadir posiciones de tiro anotados o fallados desde la posición del campo con su sistema táctil intuitivo.
- Cualquier error se puede deshacer rápidamente, mediante su lista de acciones con filtros.
- Tiempo de juego visible en todo momento con paradas automáticas dependiendo de su configuración, usted podrá utilizarlo a su antojo.
- Puede sustituir a los jugadores de campo mediante dos métodos distintos, bien para cambios múltiples o cambios individuales.
- Estadísticas: rebotes, asistencias, tapones, balones perdidos o recuperados, pasos, dobles, violaciones de tiempo, zonas, faltas cometidas o recibidas y más.
- Diagramas de Tiro, por jugador, temporada, partido, tipos de partidos.

Este sistema es muy completo, que permite a los entrenadores del baloncesto llevar las estadísticas de su equipo, diseñar jugadas, y almacenar datos de otros equipos y partidos. La recogida de datos se realiza de forma sencilla y se exportan en diferentes formatos, elementos que pueden ser tomados como ideas en el desarrollo de la solución propuesta. Además, contiene variadas características que cumplen con todas las necesidades del cliente, pero presenta una limitante, no es una aplicación gratuita por lo que impide su adquisición.

Basketball Coach

Desarrollada por la firma Canica Apps. Se trata de una buena herramienta de trabajo para entrenadores, ya que permite convertir la pantalla del smartphone en una pizarra táctil para diseñar jugadas de ataque y defensa empleando diversos elementos gráficos (líneas, balón, jugadores, conos). Está dedicada exclusivamente al baloncesto y permite preparar las tácticas, diseñar los ejercicios y planificar los entrenamientos. Facilita el diseño de estrategias a partir de la definición de las posiciones de los jugadores en determinados instantes. Contiene un reproductor para visualizar las estrategias almacenadas y lograr una mejor interpretación. Almacena hasta veinticinco jugadas de ataque o defensa. (Google Play, 2017)

Facilita la creación de ejercicios de entrenamientos mediante la pizarra, definiéndolos entre cinco posibles tipos (ejercicios físicos, de defensa, de ataque, de técnica individual, de tiro y general). Planifica entrenamientos, asignando los ejercicios guardados y el tiempo específico que trabaja cada uno. Además, se puede utilizar libremente como una pizarra para explicar indicaciones. Requiere de Android 4.0.3 o superior. (Google Play, 2017)

Sistema muy sencillo, diseñado para esbozar jugadas a través de su pizarra táctica, la cual posee elementos como líneas, balón, jugadores y conos. Estos elementos se pueden tomar como ideas para la aplicación a desarrollar, son útiles en el momento de trazar una estrategia de juego en la pizarra táctica. La aplicación no admite el almacenamiento de estadísticas en tiempo real, no brinda la opción de gestionar jugadores, equipo o torneos. Es un sistema con muy pocas funcionalidades que no responde a las necesidades del cliente. Además, es gratuita y no permite acceder a su código fuente.

1.2.2 Resultados del estudio de las herramientas

Después del análisis realizado se determinó que ninguna de las aplicaciones antes mencionadas constituye por sí misma, una solución al problema planteado en la presente investigación y, en consecuencia, no se satisfacen las necesidades del cliente. Aunque estas aplicaciones brindan información a los usuarios, ninguna contiene en su totalidad funcionalidades que se correspondan con las necesidades de los entrenadores de baloncesto. Por estas razones no se selecciona ninguna de las aplicaciones y se decide desarrollar un software, que proporcione las funcionalidades necesarias para la gestión de acciones tácticas significativas de un juego de baloncesto en tiempo real.

1.4 Metodología de desarrollo de software

Las metodologías de desarrollo de software son indispensables para crear o actualizar software de calidad que cumpla con los requisitos de los usuarios; son una parte fundamental de la Ingeniería de software la cual denomina metodología a un conjunto de métodos coherentes y relacionados por unos principios comunes. Desarrollar un buen software depende de un gran número de actividades y etapas, donde el impacto de elegir la metodología para un equipo en un determinado proyecto es trascendental para el éxito del producto. (Rivas, y otros, 2015)

Las metodologías se dividen en dos grandes grupos las tradicionales y las ágiles. Las primeras imponen una disciplina de trabajo sobre el proceso de desarrollo de software, realizando una planificación total de todo el trabajo a realizar. Este grupo es más indicado para proyectos de gran tamaño con un equipo capaz de administrar un proceso complejo en varias etapas. Las tradicionales no resultan ser las más adecuadas cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o pueden variar.

Alternativamente, surgen las metodologías ágiles, caracterizadas por ser más orientadas al desarrollo de software, con bajos niveles de formalización en la documentación requerida y por ser, a diferencia de

las tradicionales, más adaptables a los cambios, requerir de pequeños grupos de trabajo y por ser apropiadas para entornos volátiles. (Álvarez, y otros, 2015)

La selección de un enfoque y en función de este la metodología a utilizar, dependen de las circunstancias y características específicas de cada proyecto de desarrollo de software. Para esto existen diversos métodos como el propuesto por Boehm y Turner, la matriz de evolución de metodología y el método de expertos. Se escoge el método de Boehm y Turner por su fácil uso y comprensión, además permite caracterizar el proyecto de software a partir de 5 criterios y estimar cuan ágil o prescriptivo debía ser el enfoque a utilizar.

A continuación, se describen los 5 criterios propuestos: (Boeras, y otros, 2012)

- Tamaño: este criterio se utiliza para representar el número de personas involucradas en el proyecto. Pueden tenerse en cuenta el nivel de complejidad que pueda presentarse en la comunicación entre los miembros del proyecto y los costos que pueden provocar cambios esperados.
- Criticidad: se utiliza para evaluar la naturaleza del daño ocasionado por defectos que no hayan sido detectados al producto. Su evaluación puede ser cualitativa.
- Dinamismo: representa la rapidez con la que pueden estar cambiando los requerimientos del proyecto.
- Personal: representa la proporción del personal con experiencia alta, media y baja. Los métodos orientados al plan no se ven afectados negativamente por este factor pues no interesa el nivel de experiencia con la que cuenten los miembros del equipo.
- 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.

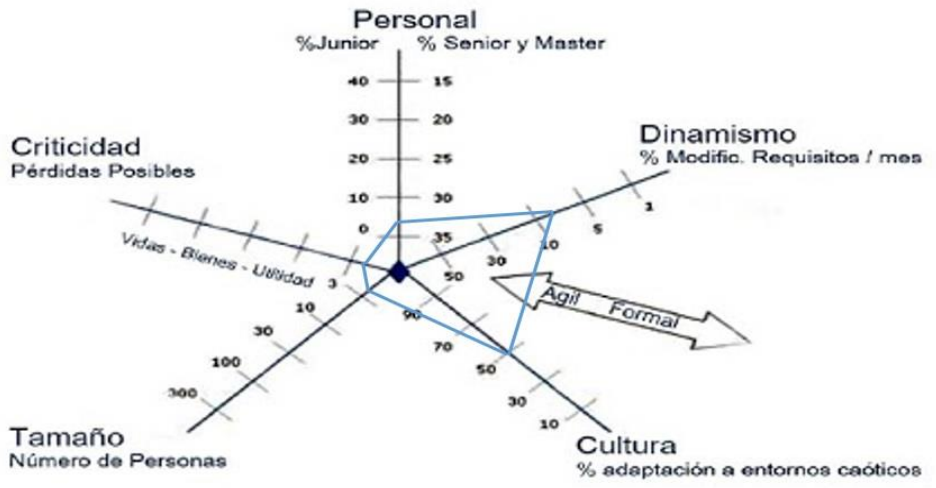


Figura 1: Aplicación del método de Boehm y Turner

Luego de haber analizado cada criterio mencionado anteriormente y en correspondencia con las características del equipo, se llega a la conclusión que se debe utilizar una metodología ágil. Dentro de las metodologías ágiles se encuentran Programación extrema (*eXtreme Programming*, XP por sus siglas en inglés) (Joskowicz, 2008) ,Proceso Unificado Ágil (*Agile Unified Process,* AUP por sus siglas en inglés*)* (Mori, y otros, 2010), entre otras.

AUP una versión simplificada del Proceso Racional Unificado (*Rational Unified Process*, RUP por sus siglas en inglés) se describe como una metodología fácil de entender para el desarrollo de aplicaciones software de negocio, utilizando técnicas ágiles y conceptos aun fieles a las de RUP. (Mori, y otros, 2010)

AUP, incluye o hace uso de las siguientes técnicas ágiles: (Mori, y otros, 2010)

- Desarrollo impulsado por pruebas.
- Desarrollo impulsado por modelos ágiles.
- Gestión de cambios ágiles.
- Refactorización de la base de datos para mejorar la productividad.

AUP, es una metodología que tiene la adopción de muchas de las técnicas ágiles de la metodología XP (Extreme Programming) y de las formalidades de RUP, teniendo como filosofía adaptarse a las necesidades del proyecto. AUP se preocupa especialmente por la gestión de riesgos. Propone que los elementos con alto riesgo obtengan prioridad en el desarrollo y sean abordados en etapas tempranas del mismo. (Mori, y otros, 2010)

El Proceso Unificado Ágil consta de cuatro fases que el proyecto atraviesa de forma secuencial. Dichas fases son, al igual que en el Proceso Unificado de Rational: (Mori, y otros, 2010)

1. **Iniciación**. El objetivo de esta fase es identificar el alcance inicial del proyecto, una arquitectura potencial para el sistema y obtener, si procede, financiación para el proyecto y la aceptación por parte de los promotores del sistema.
2. **Elaboración**. Mediante esta fase se pretende identificar y validar la arquitectura del sistema.
3. **Construcción**. El objetivo de esta fase consiste en construir software desde un punto de vista incremental basado en las prioridades de los participantes.
4. **Transición**. En esta fase se valida y despliega el sistema en el entorno de producción.

Con el fin de crear una metodología que se adapte al ciclo de vida definido por la actividad productiva en la UCI, se desarrolló una versión de la metodología de desarrollo de software AUP denominada AUP-UCI. Esta versión tiene como particularidades que se decide mantener para el ciclo de vida de los proyectos la fase de Inicio, pero modificando el objetivo de la misma y se unifican las restantes fases de la metodología de desarrollo de software AUP en una sola, nombrada Ejecución y agregándose también una nueva fase llamada Cierre. (Valdés, 2016)

Para el desarrollo de la aplicación se seleccionó la metodología AUP-UCI debido a que es una de las más utilizadas en la UCI. Además, el equipo de desarrollo es pequeño, contando solamente con un programador de mediana experiencia. Un factor importante de la selección es que ofrece una gestión de cambios ágil, permitiendo que el equipo de trabajo se adapte a nuevas circunstancias, que pueden ser provocadas por el cliente en el cambio de algunas de las funcionalidades.

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. (Martínez, y otros, 2017)

Para el desarrollo de este trabajo se seleccionó el escenario No 2, porque no es necesario incluir las responsabilidades de las personas que ejecutan las actividades. El objetivo primario del sistema es la gestión y presentación de información por lo que es recomendable utilizar este escenario.

1.5 Lenguajes, herramientas y tecnologías

Para el desarrollo de la solución propuesta se realizó un estudio de los lenguajes, herramientas y tecnologías a utilizar para el diseño, modelado e implementación de la aplicación. A continuación, se describen las características de cada uno de estos elementos.

1.5.1 Lenguajes

Lenguaje de programación

Java es un lenguaje de programación desarrollado por James Gosling y sus compañeros de Sun Microsystems al principio de la década de los 90, con el objetivo de desarrollar software altamente confiable. Este contiene una sintaxis sencilla, orientada a objetos que permite optimizar el tiempo y ciclo de desarrollo (compilación y ejecución). (García de Jalón, y otros, 2000)

Para la implementación del sistema se determinó utilizar como lenguaje de programación Java teniendo en cuenta que es el lenguaje propuesto para el desarrollo de aplicaciones móviles en la plataforma Android (Garrido, 2013). Se utilizó la versión 8 debido a que es compatible con la versión 3.0 del entorno de desarrollo integrado utilizado.

Lenguaje de modelado

El Lenguaje Unificado de Modelado (Unified Modeling Language, UML por sus siglas en inglés), es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un software. UML proporciona una forma estándar de escribir los planos de un sistema, cubriendo tanto los procesos del negocio y funciones del sistema, como esquemas de bases de datos y componentes de software reutilizables. (Rumbaugh, y otros, 2012)

Para representar las diferentes partes del sistema a desarrollar, se utilizan los diagramas de caso de uso, de componentes y de despliegue. Se utilizó la versión 1.0 del UML.

1.5.2 Herramientas y tecnologías

Entorno de desarrollo integrado

Android Studio es el entorno de desarrollo integrado (*Integrated Development Environment*, IDE por sus siglas en inglés) oficial de Android. Un entorno o ambiente de desarrollo para programas, que posee potentes herramientas de edición de código. Fue presentado por Google el 16 de mayo del 2013 en el congreso de desarrolladores Google I/O, con el objetivo de crear un entorno dedicado en exclusiva a la programación de aplicaciones para dispositivos Android. Este IDE es multiplataforma permitiendo su instalación de forma sencilla tanto en Windows como en Linux o Mac. (Robledo, 2015)
Para el desarrollo de la aplicación se utilizó Android Studio en su versión 3.0, se selecciona este IDE porque es el entorno de desarrollo integrado oficial para la plataforma Android (Invarato Menéndez, 2014). Además posee características, entre las cuales destacan: (Robledo, 2015)

- Estructura simple y organizada para los proyectos.
- Entorno de desarrollo más robusto, pero más simple, fácil e intuitivo.
- Permite visualizar en vivo el aspecto de nuestra aplicación con respecto al estilo.
- Fácil creación de proyectos para cada uno de los dispositivos que emplean Android como Sistema Operativo.

Sistema gestor de base de datos

SQLite es una biblioteca que implementa un motor de base de datos SQL (Structure Query Language, Lenguaje Estructurado de Consultas) transaccional independiente. La biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo Debe su popularidad a que es de código abierto, consume muy pocos recursos y no necesita servicios instalados para su funcionamiento. A continuación algunas de sus características: (Hwaci-Applied Software Research, 2017)

- Tamaño: tiene pequeña memoria y necesita de una biblioteca única para acceder a bases de datos, lo que lo hace ideal para aplicaciones de bases de datos incorporadas.
- Portabilidad: se ejecuta en múltiples plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.
- Costo: es de dominio público, por tanto, es libre de utilizar para cualquier propósito sin costo y se puede redistribuir libremente.
- No requiere de configuración: de la forma en que fue creado y diseñado, no necesita ser instalado.
- No es necesario prender, reiniciar o apagar un servidor, e incluso configurarlo; esta cualidad permite que no haya un administrador de base de datos para crear las tablas, vistas, asignar permisos o bien la adopción de medidas de recuperación de servidor por cada caída del sistema.

Se seleccionó SQLite porque es el gestor de base dato que utiliza Android Studio (Invarato Menéndez, 2014). Además, permite crear bases de datos muy ligeras, de forma que la aplicación podrá interactuar con los datos incluso desde dispositivos con pocas prestaciones.

Se utiliza la librería de **Java Object Relational Mapping Lite** (ORMLite) que proporciona algunas funcionalidades simples y livianas para objetos persistentes de Java en bases de datos SQL, al tiempo que se evita la complejidad y la sobrecarga de paquetes de ORM más estándar. (Lightweight Object Relational Mapping (ORM) Java Package, 2018)

ORMLite es fácil de usar y ofrece las siguientes características: (Lightweight Object Relational Mapping (ORM) Java Package, 2018)

- Configuración simple de clases agregando anotaciones Java.
- Flexible QueryBuilder para construir fácilmente consultas simples y complejas.
- Admite MySQL, PostgreSQL, Microsoft SQL Server, H2, Derby, HSQLDB y SQLite, y se puede ampliar a bases de datos adicionales con relativa facilidad.
- Autogenera SQL para crear y eliminar tablas de la base de datos.
- Soporte para la configuración de tablas y campos sin anotaciones.
- Admite llamadas nativas a las Interfaces de Programación de Aplicaciones o Application Programming Interface (API por sus siglas en inglés) de la base de datos SQLite de Android.

Herramientas de modelado UML

Visual Paradigm 8.0 es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Agiliza la construcción de aplicaciones con calidad y a un menor coste. Posibilita la generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos, así como ingeniería inversa de bases de datos. (Díaz, y otros, 2011)

Conclusiones del capítulo

Luego de un estudio realizado sobre el baloncesto se conocieron las reglas de este deporte, así como las acciones tácticas significativas de un partido. El análisis de aplicaciones que gestionara acciones tácticas significativas en los partidos de baloncesto determinó, que ninguna de estas satisface las necesidades del cliente y se decide realizar ManagerBasket. Se definió le metodología de desarrollo de software AUP-UCI, para guiar el desarrollo de la propuesta de solución de la presente investigación. Se seleccionó el lenguaje de programación Java 8, como entorno de desarrollo Android Studio en su versión 3.0 y SQLite como sistema gestor de bases de datos con la librería Java Object Relational Mapping Lite (ORM Lite) y el Visual Paradigm para el modelado de los artefactos y la arquitectura del sistema.

Capítulo 2. Sistema informático para la gestión de acciones tácticas significativas de los equipos de baloncesto

En el presente capítulo se realiza el diseño del sistema informático para la gestión de acciones tácticas significativas de los equipos de baloncesto, siguiendo la metodología AUP-UCI. Se presenta el modelo de dominio donde se analizan las entidades y conceptos relacionados con la aplicación. Se especifican los requisitos funcionales y no funcionales con los que debe cumplir el sistema, además de mostrar el diagrama de casos de uso y la descripción de los actores. Se definen los estándares de codificación utilizados para el desarrollo del sistema.

2.1 Propuesta de solución

En el baloncesto, los entrenadores recogen las acciones tácticas significativas durante un partido para su posterior análisis y toma de decisiones. Este proceso lo realizan de forma manual, lo que puede provocar en ocasiones pérdidas de la información, deterioro de la misma, y desorganización. Además, el cálculo de indicadores se realiza luego de terminado el partido impidiéndole a los entrenadores obtener resultados inmediatos de los jugadores. Estos resultados brindan información sobre el comportamiento de los jugadores en cada una de las fases del juego, y del equipo en general.

Para facilitar el trabajo de los entrenadores se propone como solución al problema presentado en la investigación, el desarrollo de una aplicación móvil para la plataforma Android, nombrada ManagerBasket. Su objetivo es el registro de datos en tiempo real durante los partidos, el diseño de estrategias de juego y el cálculo de los principales indicadores; proporcionando una mayor rapidez en la recogida de datos.

La solución propuesta permite gestionar torneos, equipos y jugadores, obteniendo detalles de cada uno de ellos. ManagerBasket posee una pizarra táctica que permite el diseño de estrategias de juego. La recogida de las acciones tácticas significativas durante un partido se realiza en tiempo real de forma rápida y sencilla. Luego de almacenada esta información se realizan los cálculos correspondientes y se muestran los resultados de los indicadores, brindándole al entrenador información necesaria para la toma de decisiones. En el menú de la aplicación se brinda la información de los torneos, equipos y jugadores, además permite acceder a las estadísticas durante el partido y después de finalizado. La información se almacena en una base de datos que puede ser compartida mediante bluetooth y wifi.

Figura 2: Representación de la propuesta de solución

2.2 Modelo conceptual

Un modelo conceptual o también conocido como modelo de dominio, es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. Su objetivo fundamental es organizar y representar, de manera semiformal, el conocimiento de un área o campo específico asociado a un sistema de gestión o de información. Para crear un modelo conceptual es suficiente con una buena definición y explicación de conceptos o entidades de negocio y de sus relaciones. (Departamento Nacional de Planeación, 2015)

En la Figura 3 se muestra el modelo conceptual de ManagerBasket.

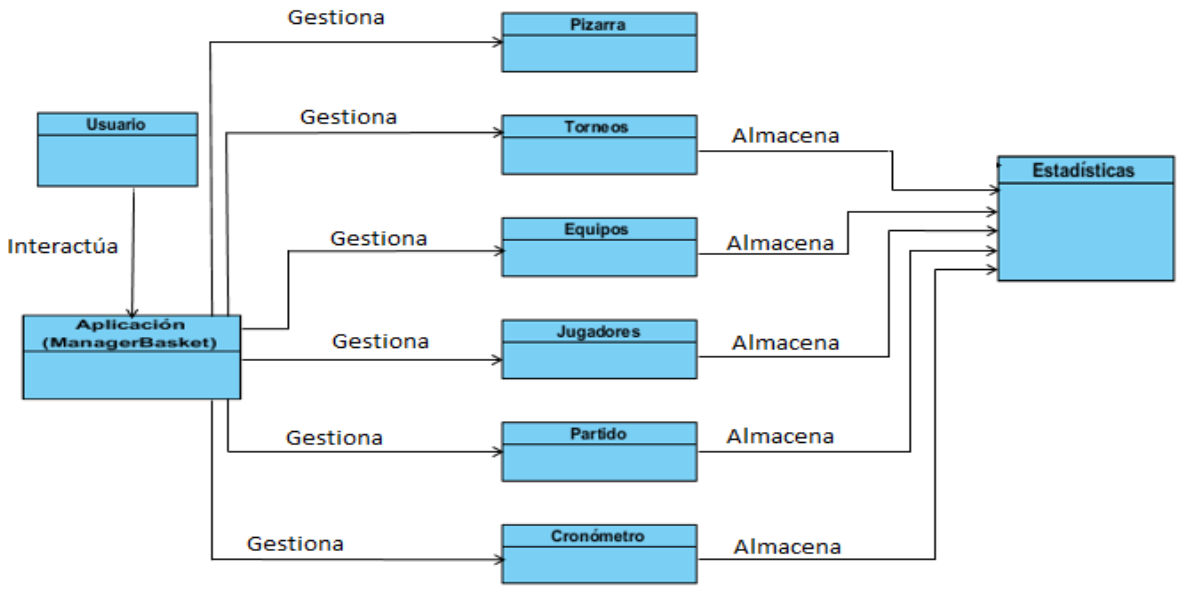


Figura 3: Modelo Conceptual

A continuación, se presenta una descripción de los conceptos involucrados en el modelo conceptual:

Usuario: persona que interactúa con la aplicación.

Dispositivo Móvil: entidad que hace referencia al medio donde se encuentra instalada la solución propuesta.

Torneo: hace referencia a los eventos de competiciones de baloncesto.

Equipo: es el colectivo integrado por varios jugadores.

Jugador: persona que juega en un partido de baloncesto.

Partido: forma de competición donde se enfrentan dos equipos.

Estadísticas: hace referencia a los indicadores que se calculan a partir de los datos registrados por el usurario. Se muestran estadísticas tanto de los jugadores como de los equipos y partidos.

Cronómetro: se encarga de controlar el tiempo del partido.

Pizarra: permite diseñar las estrategias de juego.

2.3 Requisitos funcionales del sistema

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Estos requisitos definen una función del sistema o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. (Sommerville, 2008)

Tabla 1: Listado de requisitos funcionales

Número del requisito	Nombre del requisito	Descripción
RF 1	**Gestionar torneos**	**Permite al usuario gestionar la información referente a los torneos.**
RF1.1	Insertar torneo.	Permite al usuario añadir a la base de datos los elementos de un nuevo torneo: (nombre, fecha de inicio y de fin del torneo).
RF1.2	Mostrar detalles torneo	Muestra los resultados de los partidos jugados en el torneo.
RF1.3	Listar torneos.	Lista todos los torneos existentes en la base de datos y los muestra al usuario.

RF1.4	Modificar torneo	Permite al usuario modificar un torneo seleccionado de la lista.
RF1.5	Eliminar torneo.	Permite al usuario eliminar un torneo.
RF 2	**Gestionar jugadores**	**Permite al usuario gestionar la información referente a los jugadores.**
RF2.1	Insertar jugador.	Permite al usuario añadir a la base de datos los elementos de un nuevo jugador: (nombre, apellido, posición y el equipo al que pertenece).
RF2.2	Listar jugadores.	Lista todos los jugadores existentes en la base de datos.
RF2.3	Modificar jugador	Permite al usuario modificar un jugador seleccionado de la lista.
RF2.4	Eliminar jugador.	Permite al usuario eliminar un jugador.
RF2.5	Mostrar detalles jugador	Muestra la información referente a un jugador seleccionado de la lista.
RF2.6	Exportar estadísticas a formato PDF	Exporta las estadísticas de un jugador a un archivo con extensión PDF.
RF 3	**Gestionar equipos.**	**Permite al usuario gestionar la información referente a los equipos.**
RF3.1	Insertar equipo.	Permite al usuario añadir a la base de datos el nombre de un nuevo equipo.
RF3.2	Listar equipos.	Lista todos los equipos existentes en la base de datos.
RF3.3	Modificar 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 la información referente a un equipo.
RF 4	**Crear nuevo partido**	**Permite al usuario crear un nuevo partido a partir de un torneo y equipo previamente creados.**
RF4.1	Mostrar cronómetro de juego	Genera un cronómetro que lleva el control del tiempo del partido.
RF4.2	Insertar acción	Permite al usuario insertar una acción significativa del partido.
RF4.3	Mostrar pizarra táctica	Muestra una pizarra táctica que permite al entrenador dar indicaciones al equipo.
RF4.4	Calcular estadísticas	Calcula automáticamente las estadísticas del partido en tiempo real a partir de las acciones insertadas.
RF4.5	Exportar estadísticas a formato PDF	Exporta las estadísticas de un partido a un archivo con extensión PDF.

2.4 Requisitos no funcionales del sistema

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener (Sommerville, 2008). Se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento.

A continuación los requisitos no funcionales de ManagerBasket:

Usabilidad:

- El tiempo para la introducción de datos en la aplicación no debe ser limitado.
- El sistema no debe ser adaptable a la rotación automática de la pantalla.

Apariencia e Interfaz:

- La aplicación debe tener una interfaz intuitiva, de manera que permita al usuario sin experiencia interactuar fácilmente con el sistema.

- Todos los textos y mensajes en pantalla aparecerán en idioma español.

Hardware:

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

Software:

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

2.5 Definición de los casos de usos

Los casos de uso son una técnica para especificar el comportamiento de un sistema. Se entienden como una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios. (Gutiérrez, 2011)

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

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

2.5.1 Actor del sistema

Actor	Descripción
Usuario	Persona que interactúa con la aplicación. Es el encargado de introducir la información de los jugadores, equipos, torneos y partidos.

2.5.2 Diagrama de casos de uso del sistema

Un diagrama de casos de uso del sistema (CUS) muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. (Camacho, 2013)

El diagrama de casos de uso del sistema se muestra en la Figura 4.

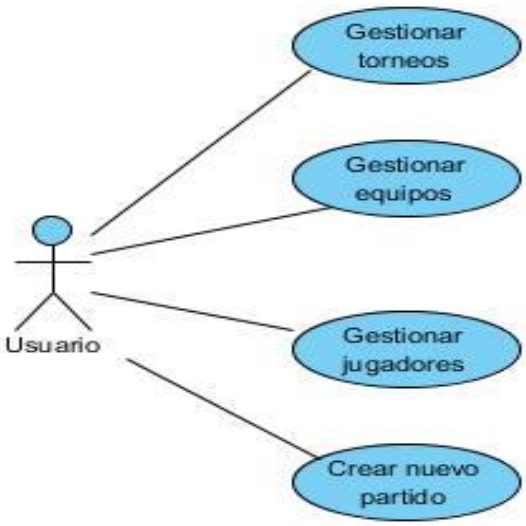


Figura 4: Diagrama de casos de uso del sistema

2.5.3 Descripción de los casos de usos del sistema

A continuación, se muestra la descripción del CUS1. Gestionar Torneos; el resto de las descripciones, pueden ser consultadas en el Anexo 1. Descripción de CUS.

Tabla 2: Descripción del CUS Gestionar torneos

Nombre	Gestionar torneos
Objetivo	Gestionar los torneos
Actor	Usuario
Resumen	El caso de uso inicia cuando el usuario desea insertar, listar, editar, ver detalles o eliminar torneos
Complejidad	Media
Prioridad	Media
Precondiciones	El usuario selecciona la opción Torneos
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 mínimo 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 torneos Si el usuario decide editar algún torneo Ver Sección 3: Editar torneo Si el usuario decide eliminar torneos Ver Sección 4: Eliminar torneos Si el usuario decide ver los detalles de algún torneo Ver Sección 5: Mostrar detalles torneo

Sección 1: Listar torneos		
Flujo básico Listar torneos		
Actor		**Sistema**
1	Selecciona la opción Torneos del menú principal	
2		Muestra una interfaz con una lista de todos los torneos existentes en la base de datos. Las opciones que se brindan son: Insertar un torneo (ver la sección 2 Insertar Torneo) Editar torneo (ver la sección 3 Editar torneo) Eliminar torneo (ver la sección 4 Eliminar torneos) Mostrar detalles del torneo (ver la sección 5 Mostrar detalles torneo)
Sección 2 Insertar torneo		
Flujo básico Insertar torneo		

Actor		Sistema
1	Selecciona la opción que permite insertar un nuevo torneo	
2		Muestra la interfaz Insertar torneo que permite introducir los valores referentes al torneo a insertar
3	Selecciona el botón Aceptar	
4		Valida los valores introducidos del torneo y muestra un mensaje de información “Torneo Insertado”
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 la opción que permite editar el torneo	
2		Muestra la interfaz Editar que permite editar los valores referentes al torneo
3	Selecciona el botón Aceptar	

4		Valida los nuevos valores introducidos del torneo y muestra un mensaje de información "El Torneo ha sido editado"
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. 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 los detalles del torneo.

<table>
<tr><td colspan="4">Sección 5. Mostrar detalles torneo</td></tr>
<tr><td colspan="4">Flujo básico Mostrar detalles torneo</td></tr>
<tr><td colspan="2">Actor</td><td colspan="2">Sistema</td></tr>
<tr><td>1</td><td>Hace clic sobre uno de los torneos listados</td><td colspan="2"></td></tr>
<tr><td>2</td><td></td><td colspan="2">Se muestra una interfaz con el listado de los partidos</td></tr>
<tr><td colspan="2">Relaciones</td><td>CU incluidos</td><td>-</td></tr>
<tr><td colspan="4">Prototipo elemental de interfaz gráfica de usuario</td></tr>
<tr><td colspan="4"> 
</td></tr>
</table>

2.6 Arquitectura de software

La estructura del sistema y la forma en la que sus componentes trabajan juntos, son elementos claves para el desarrollo de un sistema. La arquitectura de software permite modelar estos elementos proporcionando una visión más detalla del sistema. Según Pressman (Pressman, 2010) “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”.

Para el desarrollo de la aplicación se empleará la arquitectura en tres capas (ver Figura 5), su objetivo es dividir el sistema en partes diferenciadas, de tal forma que cada capa solo se comunique con la inferior. Esta arquitectura posibilita la reducción de costos por tiempo, debido a que cada capa puede ser tratada de forma independiente, lo que permite avanzar de manera segura en el desarrollo de la aplicación. En el caso que sean necesario realizar algún cambio, solo se modifica el componente necesario sin tener que revisar todo el código. Estas capas se denominan:

- **Capa de presentación**: se refiere a la presentación del programa frente al usuario, esta presentación debe cumplir su propósito con el usuario final, una presentación fácil de usar y amigable.
- **Capa de lógica de negocio**: se encuentran los programas que son ejecutados, recibe las peticiones y posteriormente envía las respuestas tras el proceso. Esta capa tiene comunicación con la de presentación ya que se tienen que comunicar para recibir las solicitudes y presentar los resultados.
- **Capa de datos**: se encarga de hacer las transacciones con la base de datos para consultar o insertar información al sistema. La comunicación de esta capa con la capa de lógica de negocio se refiere a que la capa de datos es la que le enviara información a la capa de negocio.

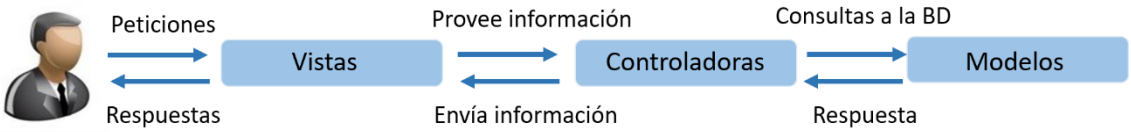


Figura 5: Arquitectura del sistema ManagerBasket

2.7 Patrón arquitectónico

Son patrones que ofrecen soluciones a problemas de arquitectura de software. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades e interrelaciones. (Departamento de Lenguajes y Sistemas Informáticos, 2013)

El patrón arquitectónico utilizado para el desarrollo de la aplicación es Modelo-Vista-Controlador (MVC), que separa los datos, la lógica de negocio y la interfaz de usuario (ver Figura 6). Este patrón logra una división de las diferentes partes que conforman la aplicación; permitiendo la actualización y mantenimiento del software de una forma sencilla y en un reducido espacio de tiempo. El MVC sugiere la separación del software en tres componentes:

- **Modelo:** es la representación de la información que maneja la aplicación. El modelo en si son los datos puros que puestos en contexto del sistema proveen de información al usuario o a la aplicación misma.
- **Vista:** es la representación del modelo en forma gráfica disponible para la interacción con el usuario, contienen los layouts, resources y widgets. Los layouts son ficheros XML de las diferentes pantallas de la interfaz gráfica. Los resources son los recursos que utiliza la aplicación y los widgets son elementos visuales que pueden mostrarse en la pantalla del dispositivo.
- **Controlador:** es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información y modificando el modelo en caso de ser necesario. Esta capa contiene

los activities, adapters y los fragments. Los primeros representan el componente principal de la interfaz gráfica de una aplicación Android. Los adapters son los responsables de adaptar los datos a los componentes de las vistas y los fragments son una porción de la interfaz de usuario que puede añadirse o eliminarse de una interfaz de forma independiente al resto de elementos de la actividad.

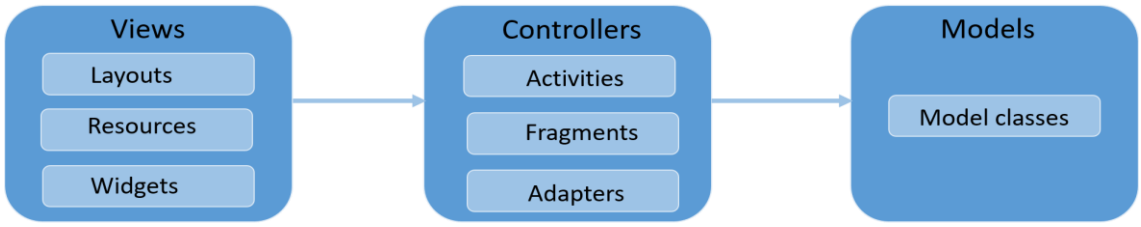


Figura 6: Patrón arquitectónico del sistema ManagerBasket

2.8 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Los patrones de diseño facilitan la reutilización de arquitecturas y diseños de software exitosos. (Pavón, 2010)

2.8.1 Patrones GRASP

Los patrones GRASP (General Responsibility Assignment Software Patterns o Patrones Generales de Software para Asignación de Responsabilidades) son una colección de principios de diseño orientados a objetos que guían la asignación de responsabilidades sobre los objetos. (Trellini, 2015)

Los siguientes patrones fueron utilizados en ManagerBasket:

Experto: permite asignar responsabilidades a las clases**.** El experto en la información es la clase que tiene la información necesaria para cumplir la responsabilidad. (Macario, 2014)

Este patrón está presente en las clases Torneos.java, Equipo.java y Jugador.java, donde cada de ellas contiene toda la información de los torneos, equipos y jugadores respectivamente (ver Figura 7).

```java
@DatabaseTable (tableName = "jugador")
public class Jugador {

    public static final String ID = "id";
    public static final String Nombre ="nombre";
    public static final String Apellido= "apellido";
    public static final String Numero = "numero";
    public static final String Posicion = "posicion";
    public static final String Equipo = "equipo";

    @DatabaseField (generatedId = true, columnName = ID) private int id;
    @DatabaseField (columnName = Nombre) private String nombre;
    @DatabaseField (columnName = Apellido) private String apellido;
    @DatabaseField (columnName = Numero) private int numero;
    @DatabaseField ( columnName = Posicion) private String posicion;
    @DatabaseField (columnName = Equipo) private String equipo;

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getNombre() { return nombre; }
    public void setNombre(String nombre) { this.nombre = nombre; }
    public String getApellido() { return apellido; }
    public void setApellido(String apellido) { this.apellido = apellido; }
    public int getNumero() { return numero; }
    public void setNumero(int numero) { this.numero = numero; }
    public String getPosicion() { return posicion; }
    public void setPosicion(String posicion) { this.posicion = posicion; }
    public String getEquipo() { return equipo; }
    public void setEquipo(String equipo) { this.equipo = equipo; }
}
```

Figura 7: Utilización del patrón Experto en la aplicación

Creador: el patrón creador nos ayuda a identificar quién debe ser el responsable de la creación de una nueva instancia de una clase. (Macario, 2014)

Este patrón se evidencia en las clases ListarTorneo.java, ListarEquipo.java y ListarJugador.java, donde cada una es la responsable de la creación de un nuevo objeto de tipo torneo, equipo o jugador respectivamente (ver Figura *8*).

```java
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab_agregar_equipo);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(ListarEquipo.this, AgregarEquipo.class);
        startActivity(intent);
        finish();
    }
});
```

Figura 8: Utilización del patrón Creador en la aplicación

Bajo Acoplamiento: el acoplamiento mide el grado en que una clase está conectada a otra, tiene conocimiento de otra o de alguna manera, depende de otra. Permite mantener un bajo acoplamiento para lograr, entre otras cosas, alta reutilización (Macario, 2014).

Este patrón está se evidencia en la relación que existe entre las clases Cronometro.java y PartidoMain.java (ver Figura 9), donde esta última utiliza la clase Cronometro.java para controlar el tiempo del partido. La clase Cronometro.java se encarga de todos los métodos referentes al cronómetro y cualquier cambio en ella no afecta la clase PartidoMain.java.

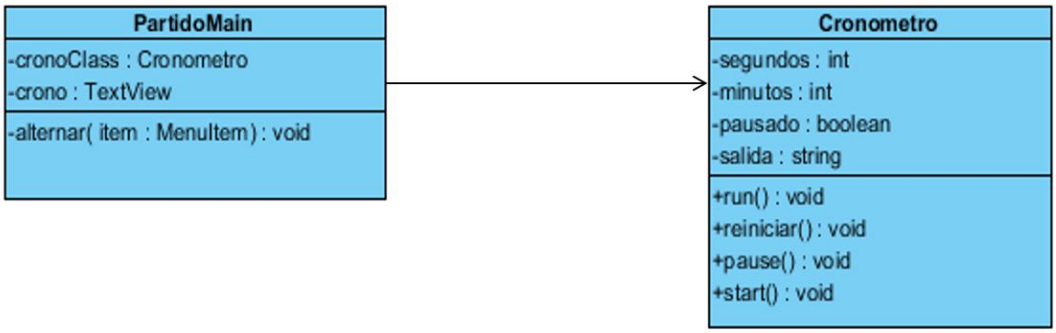


Figura 9: Utilización del patrón Bajo Acoplamiento en la aplicación.

Alta Cohesión: la cohesión mide el grado en que están relacionadas las responsabilidades de una clase. Permite mantener la complejidad de una clase en niveles manejables. (Macario, 2014)

Alta Cohesión se evidencia en la clase Pizarra.java, la cual se dedica a realizar solo la tarea para la cual fue creada (ver Figura 10). La clase PartidoMain.java permite mediante el método de la clase Pizarra.java visualizar la pizarra táctica.

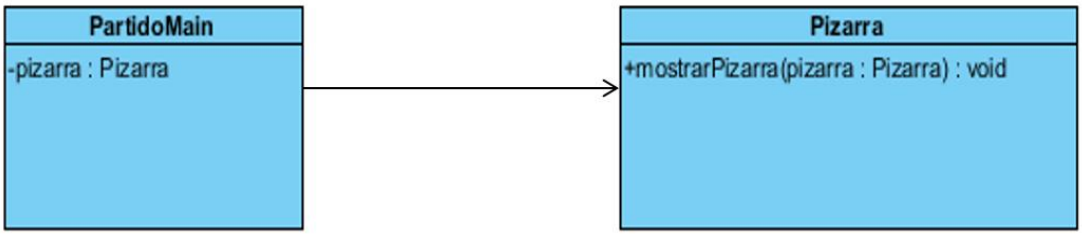


Figura 10: Utilización del patrón Alta Cohesión en la aplicación

Controlador: sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Permite manejar un evento del sistema**.** Un evento del sistema es un evento generado por un actor externo. (Macario, 2014)

Este patrón se encuentra en las clases AgregarJugador.java, AgregarEquipo.java y AgregarTorneo.java donde cada clase recibe los datos del jugador, equipo y torneo respectivamente. Un ejemplo de este patrón en la aplicación se puede observar en la (Figura 11).

```
private void insertarJugador() {
    if (retornarEquipo(spinnerequipos.getSelectedItem().toString()) == null){
        jugador_comprobar = dbManager.getNoJugadorForDatosAux(nombre.getText().toString(), apellido.getText().toString());
    }
    else {
        jugador_comprobar = dbManager.getNoJugadorForDatos(nombre.getText().toString(),
                apellido.getText().toString(), retornarEquipo(spinnerequipos.getSelectedItem().toString()));
    }

    if ((nombre.getText().toString().trim().length() > 0) &&
            (apellido.getText().toString().trim().length() > 0)  && spinnerposicion.getSelectedItemId() != 0)
    {
        if (jugadorEditado == false) {

            if (jugador_comprobar!=null){
                showMessageDialog("Ya existe este jugador");
            }
            else {
            jugador.setNombre(nombre.getText().toString());
            jugador.setApellido(apellido.getText().toString());
            jugador.setPosicion(spinnerposicion.getSelectedItem().toString());
```

Figura 11: Utilización del patrón Controlador en la aplicación

2.8.2 Patrones GoF

Los autores Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, se conocen colectivamente como Gang of Four (GOF). Estos autores publicaron un libro titulado Patrones de diseño**,** donde se contemplan 3 tipos de patrones: patrones de creación, estructurales y de comportamiento. En la aplicación se utilizó el patrón estructural decorador.

Adapter: este patrón permite adaptar dos componentes que trabajan de manera diferente pero que necesitan un traductor para que ambas puedan funcionar en conjunto. (Bustacara, 2011)

Este patrón está presente en la clase PartidoAdapter.java, donde permite adaptar una vista a la que el cliente espera. Utiliza la interfaz card_partido.xml y fragment_torneo_detalles.xml, para visualizar la lista de los partidos (ver Figura 12).

```
public static class PartidoViewHolder extends RecyclerView.ViewHolder {
    // Campos respectivos de un item
    public TextView nombre_partido;
    public TextView getFecha;
    public TextView marcador;


    public PartidoViewHolder(View v) {
        super(v);
        nombre_partido = (TextView) v.findViewById(R.id.nombre_partido_card);
        getFecha = (TextView) v.findViewById(R.id.fecha_torneo);
        marcador = (TextView) v.findViewById(R.id.marcador_final);
    }
}
```

Figura 12: Utilización del patrón Adapter en la aplicación

Builder: separa la construcción de un objeto complejo de su representación, de modo que el mismo proceso de construcción pueda crear diferentes representaciones de este objeto. (Bustacara, 2011)

Este patrón se encuentra en todos los diálogos de ManagerBasket. En la Figura 13 se muestra un ejemplo de este patrón en la clase DetallesJugador.java.

```
private void showDeleteDialogDelete(){
    final AlertDialog alertDialog = new AlertDialog.Builder( context: this)
            .setTitle("Confirmacion")
            .setMessage("Desea eliminar al equioo seleccionado")
            .setPositiveButton( text: "Si", (dialog, which) → {
                    dbManager.eliminarEquipoForID(equipoId);
                    setResult(RESULT_OK);
                    for (int i= 0; i< aux.size(); i ++){
                        aux.get(i).setEquipo(null);
                    }
                    Intent intent = new Intent( packageContext: DetallesEquipo.this, ListarEquipo.class);
                    startActivity(intent);
                    finish();
            })
            .setNegativeButton( text: "No",  listener: null)
            .create();
    alertDialog.show();
}
```

Figura 13: Utilización del patrón Builder en la aplicación

Singleton: asegura que sólo se cree una instancia de la clase y provee un punto global de acceso a ésta. Este patrón es útil cuando se quiere tener solamente un objeto único instanciado. (Universidad del Valle, 2008)

Este patrón está presente en la clase DBManager.java, la cual se crea una sola instancia de esta. Mediante ella se puede acceder a la base de datos de la aplicación desde cualquier clase que lo necesite (ver Figura 14).

```
public class DBManager {

    static private DBManager instance;
    private DBHelper dbHelper;
    private Dao<Jugador, Integer> jugadorDao;

    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: Utilización 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 por otros desarrolladores y alcanzar una uniformidad en el mismo. A continuación, se listan los elementos pertenecientes al estándar de codificación definido para el desarrollo de la aplicación:

- Dejar espacio de separación dentro del código, para que sea entendible.

- Los nombres de las variables deben ser cortos y significativos.
- Utilizar el estilo de escritura lowerCamelCase y snake _case.

Además de estos elementos antes mencionados se utilizó para el desarrollo de ManagerBasket los estándares de Java y Android.

2.10 Modelo de datos

El modelo de datos es un conjunto de conceptos que permiten describir los datos, las relaciones que existen entre ellos, sus propiedades y restricciones (Blanco, 2008). La Figura 15 muestra el modelo de datos de la aplicación.

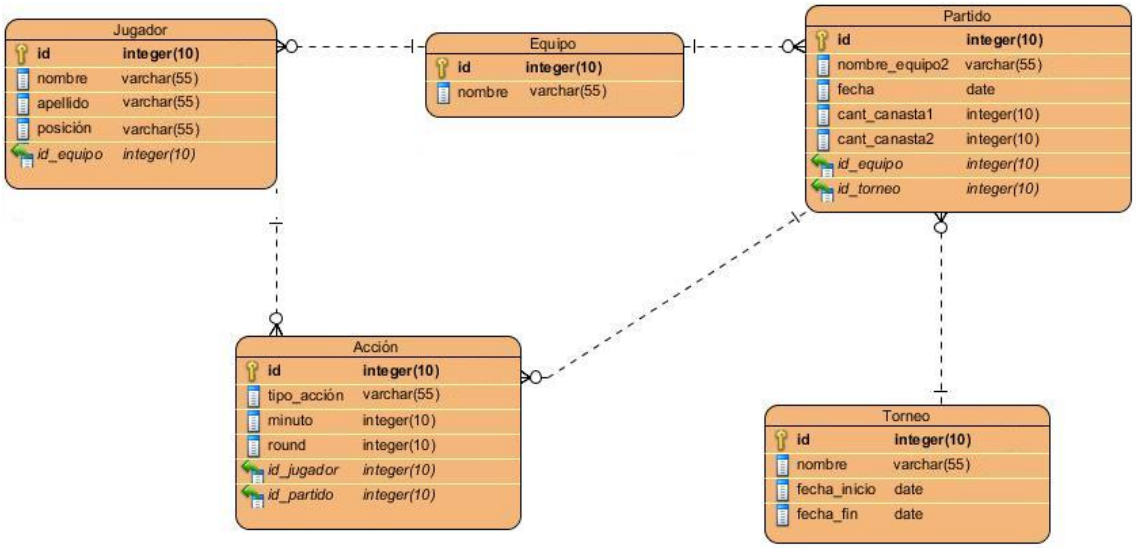


Figura 15: Modelo físico de la base de datos

A continuación se muestra la descripción de la tabla Partido del modelo físico de la base de datos, el resto se encuentran en el Anexo 2: Descripción de las tablas de la base de datos.

Tabla 3: Descripción de la tabla Partido

Partido		
Almacena los datos referentes a un partido		
Atributo	**Tipo**	**Descripción**
Id	Integer	Representa el elemento que identifica un partido
Nombre_equipo2	Varchar	Almacena el nombre del equipo adversario
Fecha	Date	Almacena la fecha del partido
cant_canasta1	Integer	Almacena la cantidad de canastas del equipo
cant_canastas2	Integer	Almacena la cantidad de canastas del equipo adversario

id_equipo	Integer	Almacena el identificador del equipo
id_torneo	Integer	Almacena el identificador del torneo

Conclusiones del capítulo

En este capítulo se describió la propuesta de solución, definiéndose los artefactos correspondientes a las etapas de análisis y diseño de la aplicación, según la metodología AUP-UCI. Se identificaron y describieron los requisitos funcionales y no funcionales. Con el modelado del diagrama de casos de uso del sistema y con sus especificaciones, se logró una descripción detallada del sistema. Se definió la arquitectura en tres capas y el modelo-vista-controlador como patrón arquitectónico, garantizando una mayor organización. Se especificaron los estándares de codificación permitiendo el desarrollo de un código reutilizable y legible. Además, se explicaron los patrones de diseño utilizados y se diseñó la base de datos de la solución propuesta.

Capítulo 3. Implementación y validación del sistema

En el presente capítulo se exponen aspectos asociados al proceso de implementación de la aplicación. Como parte de este proceso se muestra el diagrama de componentes, el cual contribuye a la organización del subsistema. El diagrama de despliegue en el que se visualiza la situación física del sistema y el diagrama de paquetes para representar las dependencias entre los paquetes que componen el modelo. Se documentan los resultados de las pruebas de software para verificar el correcto funcionamiento de la aplicación.

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. (Villar, y otros, 2016)

3.1.1 Diagrama de componentes

El diagrama de componentes describe la descomposición física del sistema de software en componentes, muestra un conjunto de componentes y sus relaciones de manera gráfica. (Universidad de Madrid, 2009)

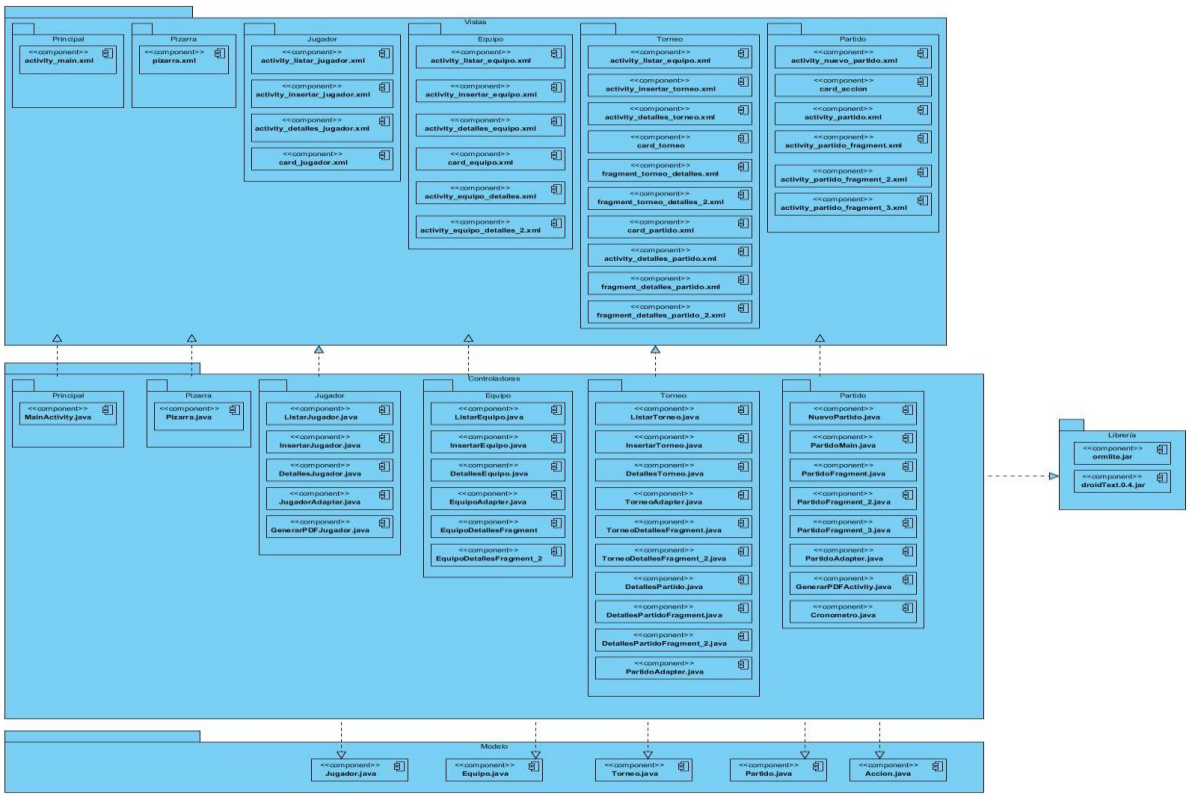


Figura 16: Diagrama de componentes

A continuación, se muestra la descripción de los paquetes:

Vistas: contienen toda la interfaz de la aplicación distribuidas en 6 paquetes. Cada paquete contiene sus vistas.

- Principal: contiene la interfaz principal de la aplicación MainActivity.xml
- Pizarra: contiene la vista referente a la pizarra táctica activity_pizarra.xml
- Jugador: almacena todas las vistas del gestionar jugador y card_jugador.xml que permite mostrar la lista de jugadores.
- Torneo: almacena todas las vistas del gestionar torneos, card_torneo.xml, card_partido.xml que permite mostrar la lista de torneos y partdios; y un activity_detalles_partido.xml que muestra información de los partidos. Además, contiene 4 fragment, dos de ellos brindan información de los torneos, y los restantes información de los partidos.
- Equipo: almacena todas las vistas del gestionar equipo y un card_equipo.xml que permite mostrar la lista de equipos. También contiene dos fragment para los detalles de los equipos.
- Partido: almacena todas las vistas de un partido. Contiene activity_nuevo_partido.xml que permite insertar un nuevo partido, activity_partido.xml la vista cuando comienza el partido. Además, contiene tres fragment nombrados activity_partido_fragment.xml que contiene la lista de jugadores del equipo, activity_partido_fragment_2.xml muestra todas las acciones insertadas en el partido, activity_partido_fragment_3.xml interfaz que muestra en tabla todas las estadísticas del partido y un card_accion.xml que permite mostrar la lista de acciones.

Controladoras: almacena todas las clases que realizan la parte lógica de la aplicación y el trabajo con la base de datos, distribuidas en 6 paquetes.

- Principal: contiene la clase principal de la aplicación MainActivity.java
- Pizarra: contiene la clase Pizarra.java que le permite al usuario dibujar las estrategias.
- Jugador: almacena todas las clases del gestionar jugador, además de la clase JugadorAdapter.java que se encarga de adaptar la vista del jugador.
- Torneo: almacena todas las clases del gestionar torneos y un DetallesPartido.java, la clase TorneoAdapter.java y PartidoAdapter.java que se encarga de adaptar la vista del torneo y todas las clases de los fragmant.
- Equipo: almacena todas las clases del gestionar equipo, la clase EquipoAdapter.java que se encarga de adaptar la vista del equipo y todas las clases de los fragmant.
- Partido: almacena todas las clases de un partido. Contiene la clase principal del partido que se encarga se crear un nuevo partido, la clase PartidoMain.java que recoge todos los datos para empezar el partido y Cronometro.java que su función es la de controlar el tiempo. Además de

todas las clases de los fragmant, se encuentra AccionAdapter.java para adaptar la vista de las acciones y GenerarPDFActivity encargada de generar los documentos en formato PDF.

Modelo: almacena todas las clases modelo de la base de datos, Jugador.java, Equipo.java, Torneo.java, Partido.java y Accion.java

Librerías: almacena todas las librerías utilizadas en la aplicación, ormlite.jar permite hacer consultas a la base de datos; droidText.0.4.jar para exportar la información almacenada sobre los partidos.

3.2 Diagrama de paquetes

El diagrama de paquete permite dividir un modelo para agrupar y encapsular sus elementos en unidades lógicas individuales. En general, pueden tener una interfaz (métodos de clases e interfaces exportadas) y una realización de éstas interfaces (clases internas que implementan dichas interfaces). (Gutiérrez, 2009)

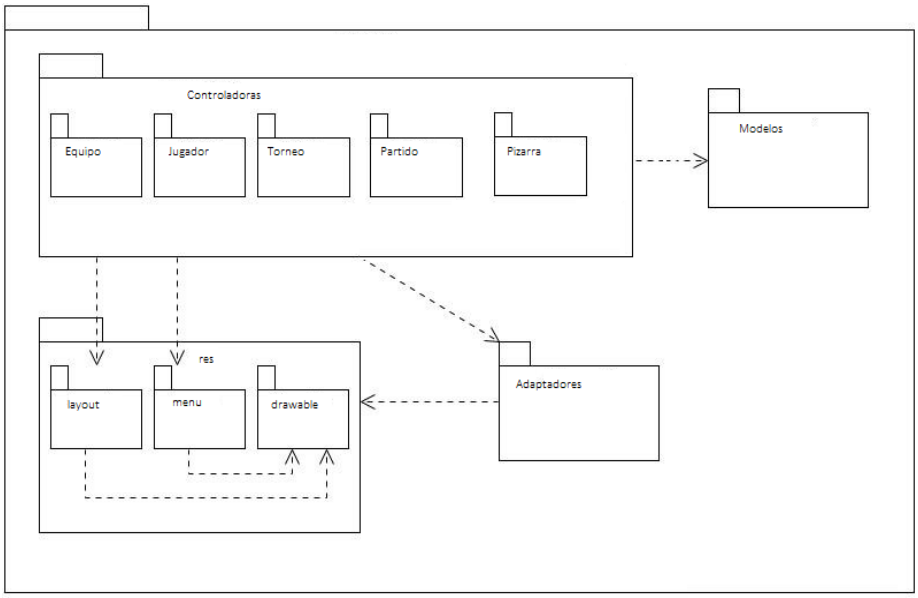


Figura 17: Diagrama de paquetes

A continuación, se muestra la descripción de los paquetes:

Controladoras: contiene todas las clases controladoras de la aplicación, contiene dentro cinco paquetes donde cada uno almacena distintas clases controladoras.

Equipo: almacena las clases controladoras referentes a la gestión de los equipos.

Jugador: almacena las clases controladoras referentes a la gestión de los jugadores.

Torneo: almacena las clases controladoras referentes a la gestión de los torneos.

Partido: almacena las clases controladoras referentes a la gestión de partidos.

Pizarra: almacena las clases controladoras referentes a la pizarra táctica.

Modelos: contiene las clases entidades que se utilizan como clases de acceso a datos.

Adaptadores: contiene las clases que sirven como adaptadores entre las clases controladoras y los layouts para poder realizar las acciones de listar equipos, jugadores, torneos y partidos en la aplicación.

Res: almacena otros paquetes en los cuales se almacenan los elementos que componen la interfaz de la aplicación.

Layouts: contiene todos los archivos XML que conforman las vistas de la aplicación.

Menú: contiene los archivos XML que componen todos los menús con que cuenta la aplicación.

Drawable: contiene todas las imágenes que son cargadas desde los archivos XML de la aplicación.

3.3 Diagrama de despliegue

Los diagramas de despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, generalmente tiene algo de memoria y a menudo, capacidad de procesamiento. Los nodos se utilizan para modelar la topología del hardware sobre el que se ejecuta el sistema. (Universidad de Madrid, 2009)

Figura 18: Diagrama de despliegue

Dispositivo Móvil: representa el hardware donde se ejecuta el sistema. La base de datos que utiliza es interna, representada por el fichero basketManager.db, donde se almacena toda la información con que trabaja la aplicación. El dispositivo debe contar con los requisitos no funcionales mencionados en el epígrafe 2.4.

3.4 Pruebas de software

Las pruebas son un conjunto de actividades que pueden planearse por adelantado y realizarse de manera sistemática, asegurando el correcto cumplimiento de las funcionalidades del producto. Las pruebas de software se han convertido en un factor determinante para logar un software con la mayor calidad posible. (Pressman, 2010)

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, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados. Una estrategia debe incluir pruebas de bajo nivel, que son necesarias para verificar que un pequeño segmento de código fuente se implementó correctamente, así como pruebas de alto nivel, que validan las principales funciones del sistema a partir de los requerimientos del cliente. Este objetivo se logra mediante una serie de pasos de prueba, primero se analizan las pruebas de unidad e integración, luego las pruebas de validación y por ultimo las pruebas del sistema. (Pressman, 2010)

Según Pressman existen cuatro niveles de prueba:

- Pruebas de unidad
- Pruebas de integración
- Pruebas de validación
- Pruebas del sistema

Las pruebas a la aplicación fueron realizadas en dos de los cuatro niveles mencionados anteriormente: pruebas de unidad y pruebas del sistema. Se comenzó por analizar pequeños elementos del software y luego el programa como un todo.

Para el nivel de pruebas de unidad se realizaron pruebas de caja blanca. Este tipo de pruebas se basa en el examen cercano de los detalles de procedimiento. Las rutas lógicas a través del software y las colaboraciones entre componentes se ponen a prueba al revisar conjuntos específicos de condiciones y/o bucles. (Pressman, 2010)

En el nivel de pruebas del sistema se desarrollaron las pruebas de caja negra. Una prueba de caja negra examina algunos aspectos fundamentales de un sistema con poca preocupación por la estructura lógica interna del software. (Pressman, 2010)

3.4.2 Pruebas de unidad

Las pruebas de unidad se enfocan en la lógica de procesamiento interno y de las estructuras de datos dentro de las fronteras de un componente. Se concentran en probar cada componente individualmente para asegurar que funcione de manera apropiada como unidad. (Campos, 2015)

A la aplicación se le realizó la prueba de caja blanca utilizando el método de ruta o trayectoria básica, propuesto por primera vez por Tom McCabe. Esta prueba permite obtener 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. (Pressman, 2010)

A continuación, se muestra una prueba de caja blanca realizada al método onKeyDown()

de la clase AgregarJugador.java.

```
1  public boolean onKeyDown(int keyCode ) {
2      if (keyCode == KeyEvent.KEYCODE_BACK ) {
3          if (nameTeam != null) {
               List<Equipo> n = dbManager.getEquipoForNombre(nameTeam);
               Intent intent = new Intent( packageContext: AgregarJugador.this, DetallesEquipo.class);
               Bundle b = new Bundle();
4              b.putInt("id", n.get(0).getId());
               intent.putExtras(b);
               startActivity(intent);
               finish();
5          } else if (flag && from.equals("detallesJugador")) {
               Intent intent = new Intent( packageContext: AgregarJugador.this, DetallesJugador.class);
               Bundle b = new Bundle();
               b.putBoolean("eq", true);
6              b.putInt("id", jugador_a_editar.get(0).getId());
               intent.putExtras(b);
               startActivity(intent);
               finish();
7          } else if (!flag && from.equals("detallesJugador")) {
               //Si se entro a editar desde los detalles de un jugador del listado general de jugadores
               Intent intent = new Intent( packageContext: AgregarJugador.this, DetallesJugador.class);
               Bundle b = new Bundle();
8              b.putInt("id", jugador_a_editar.get(0).getId());
               intent.putExtras(b);
               startActivity(intent);
               finish();
9          } else if (from.equals("listarJugador")) {
               Intent intent = new Intent( packageContext: AgregarJugador.this, ListarJugador.class);
10             startActivity(intent);
               finish();
               return true; }
11         return true;}
12     return super.onKeyDown(keyCode );
```

Figura 19: Método onKeyDown

Así queda el grafo de flujo de este método:

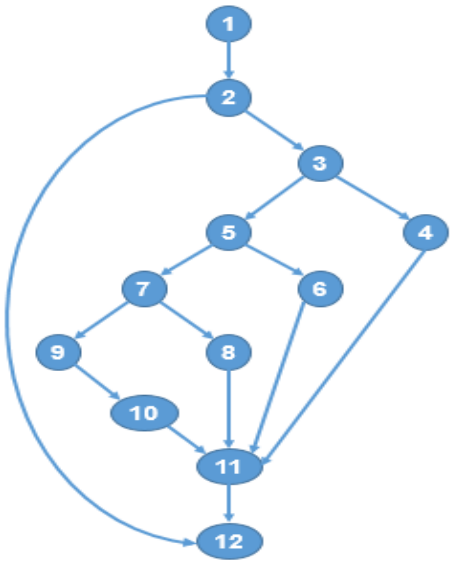


Figura 20: Grafo de flujo del método onKeyDown

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 se calcula en una de tres formas:

1. El número de regiones del gráfico de flujo corresponde a la complejidad ciclomática.
2. La complejidad ciclomática V(G) para un gráfico de flujo G se define como $V(G) = E - N + 2$,donde E es el número de aristas del gráfico de flujo y N el número de nodos del gráfico de flujo.
3. La complejidad ciclomática V(G) para un gráfico de flujo G también se define como $V(G) = P + 1$,donde P es el número de nodos predicado contenidos en el gráfico de flujo G.

En el gráfico de flujo de la figura 20, la complejidad ciclomática puede calcularse usando cada uno de los algoritmos recién indicados:

1. El gráfico de flujo tiene 5 regiones.
2. $V(G) = E - N + 2$
   $V(G) = 15 - 12 + 2$
   $V(G) = 5$
3. $V(G) = P + 1$
   $V(G) = 4 + 1$
   $V(G) = 5$

Por tanto, la complejidad ciclomática del gráfico de flujo en la figura 20 es 5. Se determinaron cinco caminos básicos de rutas linealmente independientes.

- 1,2,12
- 1,2,3,4,11,12
- 1,2,3,5,6,11,12
- 1,2,3,5,7,8,11,12
- 1,2,3,5,7,9,10,11,12

Se diseñaron cuatro casos de prueba, uno para cada camino.

Camino 1,2,12

Caso de pruebas: Obtener vista AgregarJygador

Entrada: Para un valor KeyCode != KeyEvent.KEYCODE_BACK

Resultado: Vista AgregarJugador

Camino 1,2,3,4,11,12

Caso de pruebas: Obtener vista DetallesEquipo

Entrada: Para un valor KeyCode == KeyEvent.KEYCODE_BACK**,** nameTeam != null

Resultado: Vista DetallesEquipo

Camino 1,2,3,5,6,11,12

Caso de pruebas: Obtener vista DetallesJugador

Entrada: Para un valor KeyCode == KeyEvent.KEYCODE_BACK**,** nameTeam == null y (flag=true && from.equals("DetallesJugador"))

Resultado: Vista DetallesJugador

Camino 1,2,3,5,7,8,11,12

Caso de pruebas: Obtener vista DetallesJugador

Entrada: Para un valor KeyCode == KeyEvent.KEYCODE_BACK**,** nameTeam==null, (flag=false && !from.equals("DetallesJugador")) y (flag= false && from.equals("DetallesJugador"))

Resultado: Vista DetallesJugador

Camino 1,2,3,5,7,9,10,11,12

Caso de pruebas: Obtener vista ListarJugador

Entrada: Para un valor KeyCode == KeyEvent.KEYCODE_BACK**,** nameTeam==null, (flag=false && !from.equals("DetallesJugador")), (flag= false && from.equals("DetallesJugador")) y from.equals("ListarJugador")

Resultado: Vista ListarJugador

Luego de la prueba realizada no se detectaron No Conformidades (NC) para este caso. En el resto del código se detectaron dos NC, la primera fue en el método eliminarUltimaAccion() de la clase PartidoMain.java que consistía en un error de objeto de referencia nula cuando se eliminaba la última acción. Se le dio solución agregando una condición después de recoger todas las acciones almacenadas en la base de datos *(*ver Figura 21). La segunda NC encontrada fue en el método insertarJugador() de la clase AgregarJugador.java cuando insertaba un jugador en la base de datos no verificaba si el jugador existe. Se le dio solución agregando una condición para que comprobara si los datos del jugador ya estaban registrados en la base de datos (ver Figura 22).

```
private void elimiarUltimaAccion(MenuItem item) {
    accionesEliminar = dbManager.getListAcciones();
    if (accionesEliminar != null) {
        pos = accionesEliminar.size() - 1;
        if (accionesEliminar.get(pos).getPartido().getId() == id_part) {
            dbManager.eliminarAccionForID(accionesEliminar.get(pos).getId());

            if (accionesEliminar.get(pos).getTipoAccion().equals("Tiro Libre")) {
                suma = suma - 1;
                marcador_1.setText("" + suma);
            }
```

Figura 21: Método eliminarUltimaAccion

```
private void insertarJugador() {
    if ((nombre.getText().toString().trim().length() > 0) &&
            (apellido.getText().toString().trim().length() > 0) && spinnerposicion.getSelectedItemId() != 0
            && spinnerequipos.getSelectedItemId() != 0) {
        if (retornarEquipo(spinnerequipos.getSelectedItem().toString()) == null) {
            jugador_comprobar = dbManager.getNoJugadorForDatosAux(nombre.getText().toString(), apellido.getText().toString());
        } else {
            jugador_comprobar = dbManager.getNoJugadorForDatos(nombre.getText().toString(),
                    apellido.getText().toString(), retornarEquipo(spinnerequipos.getSelectedItem().toString()));
        }
```

Figura 22: Método insertarJugador

3.4.3 Pruebas al 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 objetivo de comprobar el cumplimiento de los requisitos. Las pruebas se hacen con un enfoque desde el punto de vista del usuario. (Campos, 2015)

Esta prueba se desarrolló desde un dispositivo móvil. Para la realización de las mismas se hizo necesario el diseño de casos de pruebas que verificaran el comportamiento que correcto de la aplicación. A continuación, se muestran algunos de los casos de prueba creados para iniciar las pruebas de caja negra al sistema, el resto puede ser consultado en el Anexo 2.

Tabla 4: Caso de prueba Listar jugadores

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Listar Jugadores	Permite al usuario listar los jugadores creados.	Muestra una lista de todos los jugadores creados por el usuario.	1- Seleccionar del menú principal la opción "Jugador"

Tabla 5: Caso de prueba Listar equipos

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 2.1 Listar Equipos	Permite al usuario listar los equipos creados.	Muestra una lista de todos los equipos creados por el usuario.	1- Seleccionar del menú principal la opción "Equipos"

Tabla 6: Caso de prueba Listar torneos

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 3.1 Listar Torneos	Permite al usuario listar los torneos creados.	Muestra una lista de todos los torneos creados por el usuario.	1- Seleccionar del menú principal la opción "Torneo"

Después de realizadas las pruebas al sistema fueron detectadas siete NC:

- NC3: el sistema permitía seguir introduciendo acciones, aunque el tiempo del partido se encontrará detenido.
- NC4: el cronómetro no se detenía al terminar un período del juego.
- NC5: al insertar un equipo se repetía en la lista de equipos el nombre del último elemento almacenado
- NC6: al eliminar una acción no se actualizaba la lista de acciones.
- NC7: al salir de un partido antes de terminar el tiempo establecido, se almacenaban las acciones en la base de datos.
- NC8: el sistema no actualizaba los detalles del jugador luego de terminado un partido.
- NC9: no se actualizaban los marcadores al eliminar una acción.

Las no conformidades detectadas fueron solucionadas en las iteraciones de pruebas. A continuación, se muestran en la (tabla 7) las no conformidades detectadas en cada una de las iteraciones de las pruebas realizadas.

Tabla 7: Resultados de las pruebas

No. de Iteraciones	Total de no conformidades	Resueltas
1ra Iteración	5	NC1,NC2,NC3,NC4,NC5
2da Iteración	4	NC6, NC7, NC8, NC9
3ra Iteración	0	0

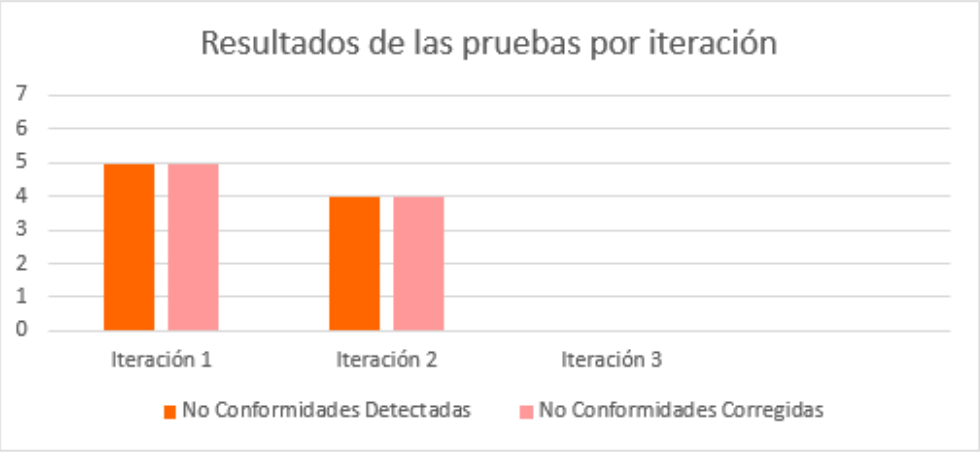


Figura 23: Resultados de las pruebas

3.4.4 Pruebas de rendimiento

La prueba de rendimiento se diseña para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado (Pressman, 2010). A la aplicación se le realizó la prueba de rendimiento con el objetivo de medir la utilización de los recursos. Se simuló un dispositivo con el emulador de Android Studio, permitiendo definir determinados aspectos de hardware de los dispositivos emulados y crear varias configuraciones para probar diferentes permutaciones de hardware. Los aspectos a tener en cuenta para esta prueba fueron los siguientes:

- Memoria de acceso aleatorio (Random Access Memory, RAM)
- Memoria interna del dispositivo
- Cantidad de núcleo del procesador

A continuación las pruebas realizadas a ManagerBasket, la figura 24 muestra las características mínimas que debe tener el dispositivo para la ejecución de la aplicación.

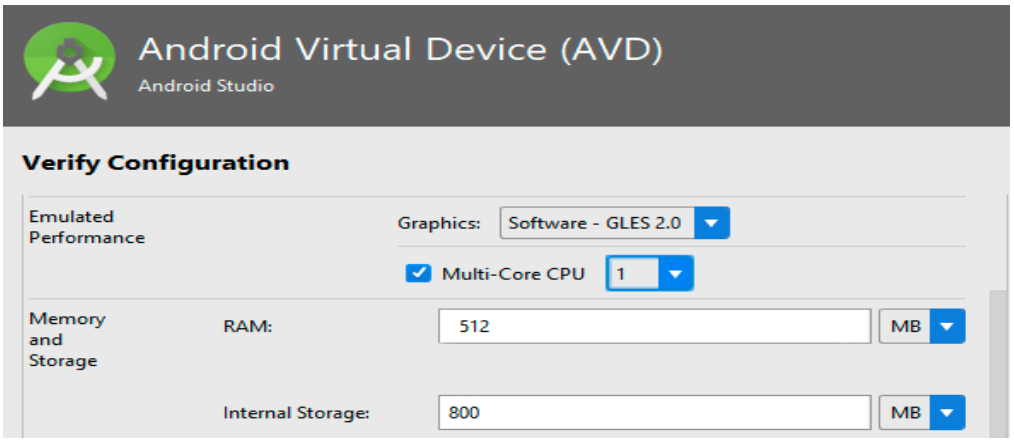


Figura 24: Visualización de ManagerBasket en emulador

Conclusiones del Capítulo

En este capítulo se explicó a partir de los resultados del diseño, la implementación de ManagerBasket. Se definió la distribución física y lógica de la arquitectura del sistema y sus conexiones mediante el diagrama de componentes y el diagrama de despliegue. Las pruebas realizadas al sistema permitieron validar y comprobar el funcionamiento de la aplicación.

CONCLUSIONES GENERALES

Luego de realizada la presente investigación, se concluye que:

1. El estudio de la bibliografía técnica del baloncesto, posibilitó conocer e identificar las principales acciones tácticas significativas registradas durante un partido y los indicadores a medir por los entrenadores, que deben estar presentes en la solución propuesta, para su posterior análisis y toma de decisiones.
2. El estudio del estado actual de las aplicaciones informáticas existentes que permiten la gestión de información estadísticas en los partidos de baloncesto, corroboró que ninguna de las aplicaciones analizadas cumple en su totalidad con las necesidades del cliente, por lo que se hizo necesario el desarrollo de ManagerBasket.
3. La selección de las herramientas y tecnologías está condicionada por la plataforma Android, para la cual se desarrolla ManagerBasket, seleccionando las más acorde con el desarrollo de aplicaciones móviles: Java como lenguaje de programación, Android Studio como entorno de desarrollo y SQLite como gestor de base de datos.
4. La aplicación del método de Boehm y Turner dio como resultado las metodologías ágiles y dentro de este grupo se seleccionó AUP-UCI, para guiar el desarrollo de la solución propuesta.
5. Se diseñó una aplicación que agiliza el proceso de recogida de información durante los partidos de baloncesto, permitiendo la gestión de jugadores, equipos y torneos, y además el cálculo automático de estadísticas.
6. Se validó ManagerBasket aplicando las pruebas de caja blanca y caja negra, obteniendo resultados satisfactorios, garantizando así el cumplimiento de los requisitos planteados.

RECOMENDACIONES

La autora de la presente investigación recomienda:

- Añadir para próximas iteraciones o versiones del producto, funcionalidades de análisis de los datos.
- Extender el uso de este tipo de aplicación para otros deportes como voleibol o fútbol.
- Extender el uso de este tipo de aplicación para otros sistemas operativos.

.

Referencias

Álvarez, Ariel y Gutiérrez, Roberto. *API para el trabajo con llamadas de datos en el Sistema Operativo Android.* La Habana, 2015.

Blanco, Miguel Ángel. *Generador del Modelo Relacional y Esquemas de Base de Datos a partir del modelo Entidad/Relación.* Madrid , 2008.

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

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

Camacho, Jonathan. *Diagrama de clases, Diagrama de casos de uso.* 2013.

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

Carranza, Marta y Maza, Gaspar. *Deporte, actividad física e inclusión social. Una guía para la intervención social a través de las actividades deportivas.* 2011.

Departamento de Lenguajes y Sistemas Informáticos. *Patrones Arquitectónico.* Sevilla, 2013.

Departamento Nacional de Planeación. *MANUAL DE ELABORACIÓN DE MODELOS CONCEPTUALES.* Bogotá, 2015.

Diaz, Juliet; Pérez, Adriana y Florido, René. *Impacto de las tecnologías de las información y las comunicaciones (TIC) para disminuir la brecha digital en le sociedad actual.* Cultivos Tropicales, 2011. págs. 81-90.

Díaz, Abel; Camejo, Lianet y Quintana, Yoandri. *Diseño de la Base de Datos para Sistemas de Digitalización y Gestión de Medias.* La Habana, 2011.

Federación Internacional de Baloncesto (FIBA). Federación Internacional de Baloncesto (FIBA). [En línea] 2016. [Citado el: 2018 de febrero de 12.] http://www.fiba.basketball/es/basic-rules.

García de Jalón, Javier , y otros. *Aprende Java como si estuvieras en primero.* Navarra, 2000.

García, Mar. *Un juego.* 2013.

Garrido, Juan. *TFC Desarrollo de Aplicaciones Móviles.* 2013.

Gil, Salvador Alejandro Tovar y KIM, Hyun Sook Lee. *La evolución de la telecomunicación hacia una sociedad conectada.* 2017. págs. 27-48, Revista de Investigación en Ciencias de la Administración.

Goldstein, Sidney. *BIBLIA DEL ENTRENADOR DE BALONCESTO. Una guía sistemática y exahustiva del entrenamiento.* s.l. : Paidotribo, 2011. 9788480196604.

Google Play. Baloncesto Five On Court. [En línea] 2017. https://play.google.com/store/apps/details?id=app.android.fiveoncourt.

Google Play. Basketball Coach. [En línea] 2017. https://play.google.com/store/apps/details?id=com.canica.apps.basket&hl=es_419.

Google Play. BasketBloc. [En línea] 2017. https://play.google.com/store/apps/details?id=basketbloc.basketball..

Google Play. BasketStats Lite. [En línea] 2017. file:///D:/Escuela/mi%20tesis/aplicaciones%20oko/BasketStats%20Lite%20-%20Aplicaciones%20de%20Android%20en%20Google%20Play.htm.

Google Play. OmniStats BasketBall. [En línea] 2017. http://www.efeemprende.com/noticia/omnistats-baskerball-app-estadisticas-baloncesto/.

Gutiérrez, Demián. *Casos de Uso, Diagramas de Casos de Uso.* Venezuela, 2011.

Gutiérrez, Demián. *UML Diagramas de Paquetes*, 2009.

International Data Corporation IDC. Analyze the Future. Smartphone OS. [En línea] 2017. [Citado el: 16 de noviembre de 2017.] http://www.idc.com/prodserv/smartphone-os-market-share.jsp..

Invarato, Ramón. *Android 100%.* 2014.

Joskowicz, José. *Reglas y Prácticas en eXtreme Programming.* Vigo, España, 2008.

Lightweight Object Relational Mapping (ORM) Java Package. OrmLite - Lightweight Object Relational Mapping (ORM) Java Package. [En línea] 2018. [Citado el: 4 de enero de 2018.] http://ormlite.com/.

Macario, Usaola. *Patrones GRASP.* 2014.

Malave, Kristel y Beauperthuy, José Luis. *Android el sistema operativo de google para dispositivos móviles.* 2011.

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

Mori, José Germán. *USABILIDAD EN METODOLOGÍAS ÁGILES.* Madrid, España, 2010.

Olivera, Javier y TICÓ, Jordi. *Análisis funcional del baloncesto como deporte de equipo.* 1992. págs. 34-46.

Pavón, Juan. *Patrones de diseño orientado a objetos.* Madrid, 2010.

Pressman, Roger S. *Un efoque Práctico.* 2010.

Rivas, Carlos Ignacio, y otros. *Metodologías actuales de desarrollo de software.* México, 2015.

Robledo, David. *Desarrollo de Aplicaciones para Android I.* España, 2015.

Rumbaugh, James y Jacobson, Ivan. *Lenguaje Unificado de Modelado.* 2012.

Sampaio, Jaime; Janeira, Manuel Antonio y Brandao, Eurico. *Evaluación del jugador en los partidos de baloncesto.* Portugal, 2002.

Hwaci-Applied Software Research. Sitio Oficial de SQLite. [En línea] 2017. [Citado el: 20 de diciembre de 2017.] https://www.sqlite.org.

Sommerville, Ian. *Ingeniería de software.* 2008.

Trellini, Ariel. *Arquitectura y Diseño de Sistemas.* 2015.

Universidad de Madrid. *Diseño Basado en Componentes. UML aplicado al diseño basado en componentes.* Madrid, 2009.

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

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

Villar, Aray y Hernández, José Javier. *Herramienta para el procesamiento en lote de documentos digitales para el sistema REPXOS.* La Habana, 2016.

Bibliografía

Álvarez, Ariel y Gutiérrez, Roberto. *API para el trabajo con llamadas de datos en el Sistema Operativo Android.* La Habana, 2015.

Bernal, Gabriela. *El desarrollo tecnológico, una perspectiva social y humanista.* 2006.

Blanco, Miguel Ángel. *Generador del Modelo Relacional y Esquemas de Base de Datos a partir del modelo Entidad/Relación.* Madrid, 2008.

Blanco,Paco, y otros. *Metodología de desarrollo ágil para sistemas móviles Introducción al desarrollo con Android y el iPhone.* Madrid , 2016.

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

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

Camacho, Jonathan . *Diagrama de clases, Diagrama de casos de uso.* 2013.

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

Carranza, Marta y Maza, Gaspar. *Deporte, actividad física e inclusión social. Una guía para la intervención social a través de las actividades deportivas.* 2011.

Contreras, Gabriel Rubén. *Tecnologías Móviles.2015.*

Departamento de Lenguajes y Sistemas Informáticos. *Patrones Arquitectónico.* Sevilla, 2013.

Departamento Nacional de Planeación. *MANUAL DE ELABORACIÓN DE MODELOS CONCEPTUALES.* Bogotá, 2015.

Diaz, Juliet; Pérez, Adriana y Florido Bacallao, René. *Impacto de las tecnologías de las información y las comunicaciones (TIC) para disminuir la brecha digital en le sociedad actual.* Cultivos Tropicales, 2011. págs. 81-90.

Díaz, Abel; Camejo, Lianet y Quintana, Yoandri. *Diseño de la Base de Datos para Sistemas de Digitalización y Gestión de Medias.* La Habana, 2011.

Federación Internacional de Baloncesto (FIBA). 2016. Federación Internacional de Baloncesto (FIBA). [En línea] 2016. [Citado el: 2018 de febrero de 12.] http://www.fiba.basketball/es/basic-rules.

García de Jalón, Javier , y otros. 2000. *Aprende Java como si estuvieras en primero.* Navarra, 2000.

García, Mar. *Un juego.* 2013.

Garrido, Juan. *TFC Desarrollo de Aplicaciones Móviles.* 2013.

Gil, Salvador Alejandro Tovar y KIM, Hyun Sook Lee. 2017. *La evolución de la telecomunicación hacia una sociedad conectada.* 2017. págs. 27-48, Revista de Investigación en Ciencias de la Administración.

Goldstein, Sidney. *BIBLIA DEL ENTRENADOR DE BALONCESTO. Una guía sistemática y exahustiva del entrenamiento.* Paidotribo, 2011. 9788480196604.

Gómez Fuentes,María del Carmen. *MATERIAL DIDÁCTICO.NOTAS DEL CURSO ANÁLISIS DE REQUERIMIENTOS.* 2011.

Google Play. 2017. Baloncesto Five On Court. [En línea] 2017. https://play.google.com/store/apps/details?id=app.android.fiveoncourt.

Google Play. 2017. Basketball Coach. [En línea] 2017. https://play.google.com/store/apps/details?id=com.canica.apps.basket&hl=es_419.

Google Play. 2017. BasketBloc. [En línea] 2017. https://play.google.com/store/apps/details?id=basketbloc.basketball..

Google Play. 2017. BasketStats Lite. [En línea] 2017. file:///D:/Escuela/mi%20tesis/aplicaciones%20oko/BasketStats%20Lite%20-%20Aplicaciones%20de%20Android%20en%20Google%20Play.htm.

Google Play. 2017. OmniStats BasketBall. [En línea] 2017. http://www.efeemprende.com/noticia/omnistats-baskerball-app-estadisticas-baloncesto/.

Gutiérrez, Demián. 2011. *Casos de Uso, Diagramas de Casos de Uso.* Venezuela, 2011.

Gutiérrez, Demián. 2009. *UML Diagramas de Paquetes.* 2009.

International Data Corporation IDC. 2017. Analyze the Future. Smartphone OS. [En línea] 2017. [Citado el: 16 de noviembre de 2017.] http://www.idc.com/prodserv/smartphone-os-market-share.jsp..

Invarato Menéndez, Ramón. *Android 100%.* 2014.

Joskowicz, José. *Reglas y Prácticas en eXtreme Programming.* Vigo, España, 2008.

Lightweight Object Relational Mapping (ORM) Java Package. 2018. OrmLite - Lightweight Object Relational Mapping (ORM) Java Package. [En línea] 2018. [Citado el: 4 de enero de 2018.] http://ormlite.com/.

Macario, Usaola. *Patrones GRASP.* 2014.

Malave, Kristel y Beauperthuy, José Luis. *Android el sistema operativo de google para dispositivos móviles.* 2011.

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

Mori, José Germán. *USABILIDAD EN METODOLOGÍAS ÁGILES.* Madrid, España, 2010.

Olivera, Javier y TICÓ, Jordi. *Análisis funcional del baloncesto como deporte de equipo.* 1992. págs. 34-46.

Pavón, Juan. *Patrones de diseño orientado a objetos.* Madrid, 2010.

Peinado, Federico . *Patrón Modelo-Vista-controlador.*

Pressman, Roger S. *Un efoque Práctico.* 2010.

Rivas, Carlos Ignacio, y otros. *Metodologías actuales de desarrollo de software.* México, 2015.

Robledo, David. *Desarrollo de Aplicaciones para Android I.* España, 2015.

Rumbaugh, James y Jacobson, Ivan. *Lenguaje Unificado de Modelado.* 2012.

Sampaio, Jaime; Janeira, Manuel Antonio y Brandao, Eurico. 2002. *Evaluación del jugador en los partidos de baloncesto.* Portugal, 2002.

Hwaci-Applied Software Research. Sitio Oficial de SQLite. [En línea] 2017. [Citado el: 20 de diciembre de 2017.] https://www.sqlite.org.

Sommerville, Ian. 2008. *Ingeniería de software.* 2008.

Tinoco, Oscar ; Rosales, Pedro Pablo y Salas, Julio. *Criterios de selección de metodologías de desarrollo de software.* 2010.

Trellini, Ariel. *Arquitectura y Diseño de Sistemas.* 2015.

Universidad de Madrid. *Diseño Basado en Componentes. UML aplicado al diseño basado en componentes.* Madrid, 2009.

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

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

Villar Machado, Aray y Hernández Benítez, José Javier. *Herramienta para el procesamiento en lote de documentos digitales para el sistema REPXOS.* La Habana, 2016.

ANEXOS

Anexo 1: Uso de Android en muestra tomada

A continuación se muestra el uso del sistema operativo Android en una muestra tomada en la Universidad de las Ciencias Informáticas (ver Figura 22).

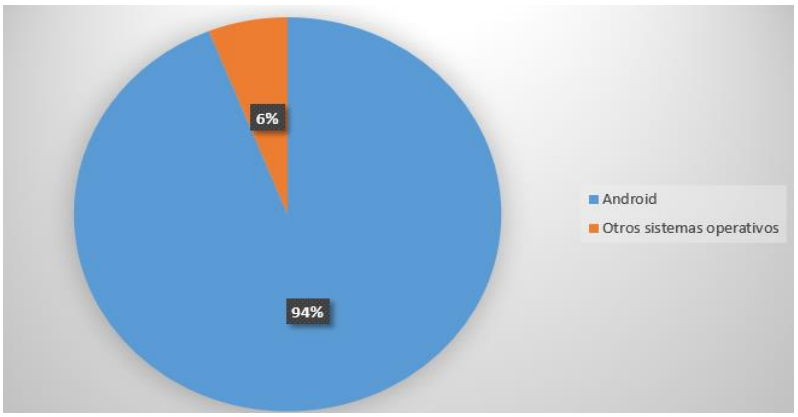


Figura 25: Uso del sistema operativo Android en teléfonos inteligentes

Anexo 2: Descripción de los CUS

Tabla 8: Descripción del CUS Gestionar equipos

Nombre	Gestionar equipos
Objetivo	Gestionar los equipos
Actor	Usuario
Resumen	El caso de uso inicia cuando el usuario desea insertar, listar, editar, ver detalles o eliminar equipos
Complejidad	Media
Prioridad	Media
Precondiciones	El usuario selecciona la opción equipos
Postcondiciones	Se inserta, lista, edita o elimina los equipos
Flujo de eventos	
Flujo básico Gestionar equipo	
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 decide ver los detalles de algún equipo Ver Sección 5: Mostrar detalles equipo
Sección 1: Listar equipos		
Flujo básico Listar equipos		
Actor		**Sistema**
1	Selecciona la opción Equipos del menú principal	
2		Muestra una interfaz con una lista de todos los equipos existentes en la base de datos. Las opciones que se brindan son: Insertar un equipo (ver la sección 2 Insertar equipo) Editar cada equipo (ver la sección 3 Editar equipo) Eliminar equipo (ver la sección 4 Eliminar equipo) Mostrar detalles del equipo (ver la sección 5 Mostrar detalles equipo)
Sección 2 Insertar equipo		
Flujo básico Insertar equipo		
Actor		**Sistema**
1	Selecciona la opción que permite insertar un nuevo equipo	
2		Muestra la interfaz Insertar equipo que permite introducir los valores referentes al equipo a insertar
3	Selecciona el botón Aceptar	
4		Valida los valores introducidos del equipo y muestra un mensaje de información "Equipo Insertado"
Flujo alterno Insertar equipo		
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 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 los valores referentes al equipo
3	Selecciona el botón Aceptar	
4		Valida los nuevos valores introducidos del equipo y muestra un mensaje de información "El Equipo ha sido editado"
Flujo alterno Editar equipo		
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 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 Mostrar detalles equipo		
Actor		**Sistema**
1	Selecciona uno de los equipos listados	
2		Se muestra una interfaz con la información referente al equipo seleccionado.
Relaciones	**CU incluidos**	**-**
Prototipo elemental de interfaz gráfica de usuario		

Tabla 9: Descripción del CUS Gestionar jugadores

Nombre	Gestionar jugadores	
Objetivo	Gestionar los jugadores	
Actor	Usuario	
Resumen	El caso de uso inicia cuando el usuario desea insertar, listar, editar, ver detalles o eliminar jugadores	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El usuario selecciona la opción Jugadores	
Postcondiciones	Se inserta, lista, edita o elimina los jugadores	
Flujo de eventos		
Flujo básico Gestionar jugadores		
Actor		**Sistema**
Indica que desea insertar, listar, editar, ver detalles 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 Si el usuario decide ver los detalles de algún jugador Ver Sección 5: Mostrar detalles jugador Si el usuario decide exportar las estadísticas a PDF. Ver Sección 6: Exportar estadísticas a PDF
Sección 1: Listar jugador		
Flujo básico Listar jugador		
Actor		**Sistema**
1	Selecciona la opción Jugadores del	

	menú principal	
2		Muestra una interfaz con una lista de todos los jugadores existentes en la base de datos. 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) Mostrar detalles del jugador (ver la sección 5 Mostrar detalles Jugador) Exportar estadísticas a PDF (ver la sección 6 Exportar estadísticas a PDF)
Sección 2 Insertar jugador		
Flujo básico Insertar jugador		
Actor1		**Sistema**
1	Selecciona la opción que permite insertar un nuevo jugador	
2		Muestra la interfaz Insertar jugador que permite introducir los valores referentes al jugador a insertar
3	Selecciona el botón Aceptar	
4		Valida los valores introducidos del jugador y muestra un mensaje de información “Jugador Insertado"
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
3	Selecciona el botón Aceptar	
4		Valida los nuevos valores introducidos del jugador y

		muestra un mensaje de información "El Jugador ha sido editado"
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.
Sección 5 Mostrar detalles jugador		
Flujo básico Mostrar detalles jugador		
Actor		**Sistema**
1	Hace clic sobre uno de los jugadores listados	
2		Se muestra una interfaz con la información referente al jugador seleccionado.
Sección 6 Exportar estadísticas a formato PDF		
Flujo básico Exportar estadísticas a formato PDF		
Actor		**Sistema**
1	Una vez seleccionado un jugador, presiona el botón exportar	
2		Exporta las estadísticas del jugador seleccionado a formato PDF

Relaciones	CU incluidos	-
Prototipo elemental de interfaz gráfica de usuario		

Tabla 10: Descripción del CUS Crear nuevo partido

Nombre	Crear nuevo partido	
Objetivo	Registrar un partido	
Actor	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. Exportar estadísticas a formato PDF. Ver la sección 6: Exportar estadísticas a formato PDF.
Sección 1: Crear nuevo partido		
Flujo básico Crear nuevo partido		
Actor		**Sistema**
1	Selecciona la opción Nuevo Partido del menú principal	

2		Muestra la interfaz Insertar Equipo que permite introducir los valores referentes al partido a insertar
3	Selecciona el botón Aceptar	
		Valida los valores introducidos del partido y muestra una interfaz con el listado de los jugadores del equipo, la pizarra del partido, que incluye marcador de canastas, cronómetro del partido, y los equipos que se enfrentan. Las opciones que se brindan son: Insertar acción (ver sección 2: Insertar Acción.) Registro de acciones (ver Sección 3: Registro de Acciones del Partido.) Estadísticas del partido (ver Sección 4: Estadísticas.) Exportar estadísticas a formato PDF (ver la sección 6: Exportar estadísticas a formato PDF)
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 acciones
Sección 3 Registro de acciones del partido		
Flujo básico Registro de acciones del partido		
Actor		**Sistema**
1	Selecciona la opción acciones	
2		Muestra el listado de todas las acciones insertadas (ver Sección 2: Insertar Acción) del partido.
Sección 4 Estadísticas		
Flujo básico Estadísticas		
Actor		**Sistema**
1	Selecciona la opción estadísticas	
2		Muestras las estadísticas del partido calculadas a partir de las acciones insertadas (ver Sección 2: Insertar Acción)
Sección 5 Exportar estadísticas a PDF		
Flujo básico Exportar estadísticas a PDF		

Actor		Sistema
1	Una vez terminado el partido, presiona el botón exportar	
2		Exporta las estadísticas del partido seleccionado a formato PDF
Relaciones	**CU incluidos**	**-**
Prototipo elemental de interfaz gráfica de usuario		

Anexo 3: Descripción de las tablas de la base de datos

Tabla 11: Descripción de la tabla Jugador

Jugador		
Almacena los datos referentes a un jugador		
Atributo	**Tipo**	**Descripción**
Id	Integer	Representa el elemento que identifica un jugador
Nombre	Varchar	Almacena el nombre del jugador
Apellido	Varchar	Almacena el apellido del jugador
Posición	Varchar	Almacena la posición del jugador
id_equipo	Integer	Almacena el identificador del equipo al que pertenece el jugador

Tabla 12: Descripción de la tabla Equipo

Equipo
Almacena los datos referentes a un equipo

Atributo	Tipo	Descripción
Id	Integer	Representa el elemento que identifica un equipo
Nombre	Varchar	Almacena el nombre del equipo

Tabla 13: Descripción de la tabla Acción

Acción		
Almacena los datos referentes a las acciones		
Atributo	**Tipo**	**Descripción**
Id	Integer	Representa el elemento que identifica una acción
Tipo_acción	Varchar	Almacena el tipo de acción que se realizó
Minuto	Integer	Almacena el minuto cuando fue insertada una acción
Round	Integer	Almacena el tiempo cuando fue insertada la acción
id_jugador	Integer	Almacena el identificador del jugador
id_partido	Integer	Almacena el identificador del partido donde fue insertada la acción

Tabla 14: Descripción de la tabla Torneo

Torneo		
Almacena los datos referentes a un torneo		
Atributo	**Tipo**	**Descripción**
Id	Integer	Representa el elemento que identifica un torneo
Nombre	Varchar	Almacena el nombre del torneo
fecha_inicio	Date	Almacena la fecha de inicio del torneo
fecha_fin	Date	Almacena la fecha de fin del torneo

Anexo 4: Diseño de casos de prueba

Tabla 15: Caso de prueba eliminar jugador

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.2 Eliminar Jugador	Permite al usuario eliminar uno de los jugadores creados.	Elimina el jugador seleccionado por el usuario.	1-Seleccionar del menú principal la opción "Jugador" 2-Seleccionar el jugador que desea eliminar 3-Seleccionar la opción eliminar de la barra superior 4-Confirmar la eliminación

Tabla 16: Caso de prueba eliminar equipo

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 2.2 Eliminar Equipo	Permite al usuario eliminar uno de los equipos creados.	Elimina el equipo seleccionado por el usuario.	1-Seleccionar del menú principal la opción "Equipos" 2-Seleccionar el equipo que desea eliminar 3-Seleccionar la opción eliminar de la barra superior 4-Confirmar la eliminación

Tabla 17: Caso de prueba eliminar torneo

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 2.2 Eliminar Torneo	Permite al usuario eliminar uno de los torneos creados.	Elimina el torneo seleccionado por el usuario.	1-Seleccionar del menú principal la opción "Torneos" 2-Seleccionar el torneo que desea eliminar 3-Seleccionar la opción eliminar de la barra superior 4-Confirmar la eliminación

Tabla 18: Caso de prueba editar jugador

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.3 Editar Jugador	Permite al usuario editar uno de los jugadores creados.	Edita el jugador seleccionado por el usuario.	1-Seleccionar del menú principal la opción "Jugador" 2-Seleccionar el jugador que desea editar 3-Seleccionar la opción editar de la barra superior

Tabla 19: Caso de prueba editar equipo

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 2.3 Editar Equipo	Permite al usuario editar uno de los equipos creados.	Edita el equipo seleccionado por el usuario.	1-Seleccionar del menú principal la opción "Equipo" 2-Seleccionar el equipo que desea editar 3-Seleccionar la opción editar de la barra superior

Tabla 20: Caso de prueba editar torneo

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 3.3 Editar Torneo	Permite al usuario editar uno de los torneos creados.	Edita el torneo seleccionado por el usuario.	1-Seleccionar del menú principal la opción "Torneo" 2-Seleccionar el torneo que desea editar 3-Seleccionar la opción editar de la barra superior

Tabla 21: Caso de prueba insertar equipo

Escenario	Descripción	Nombre	Foto	Respuesta del sistema	Flujo central

EC 4.1 Insertar equipo	Permite al usuario crear un nuevo equipo.	V	V	1- Inserta en la base de datos el equipo creado por el usuario, para ello debe llenar los campos • Nombre • Logo 2- Muestra el mensaje “Equipo Insertado”	1-Seleccionar del menú principal la opción "Equipos" 2-Selecciona el botón Agregar ubicado en la esquina inferior derecha 3-Introduce los datos del equipo 4-Selecciona el botón Aceptar
		V	V vacío		
EC 4.2 Insertar equipo	Validar que no existen campos vacíos	I vacío	V	1- En el caso en que el campo Nombre esté vacío muestra el mensaje “Debe introducir el nombre del equipo”	1-Seleccionar del menú principal la opción " Equipos" 2-Selecciona el botón Agregar ubicado en la esquina inferior derecha 3-Introduce los datos del Equipo 4-Selecciona el botón Aceptar
		V	V vacío		
EC 4.3 Insertar equipo	Validar que no existen campos inválidos	I F@cultad	V	1- No permite introducir caracteres inválidos	1-Seleccionar del menú principal la opción " Equipos" 2-Selecciona el botón Agregar ubicado en la esquina inferior derecha 3-Introduce los datos del Equipo 4-Selecciona el botón Aceptar
		V	V		

Tabla 22: Caso de prueba insertar jugador

Escenario	Descripción	Nombre	Apellido	Posición	Equipo	Foto	Respuesta del sistema	Flujo central
EC 5.1 Insertar jugador	Permite al usuario crear un nuevo jugador.	V	V	V	V	V	1- Inserta en la base de datos el jugador creado por el usuario, para ello debe	1- Seleccionar del menú principal la opción Jugadores 2- Selecciona

		V	V	V	V	V vacío	llenar los campos -Nombre -Apellido -Posición -Equipo 2- Muestra el mensaje “Jugador Insertado”	el botón Agregar ubicado en la esquina inferior derecha 3-Introduce los datos del jugador 4- Selecciona el botón Aceptar
EC 5.2 Insertar jugador	Validar que no existen campos vacíos	I vacío	V	V	V	V	1- En el caso en que los campos Nombre, Apellido, Equipo y Posición estén vacíos muestra el mensaje “Todos los campos son obligatorios”	1- Seleccionar del menú principal la opción "Jugadores " 2- Selecciona el botón Agregar ubicado en la esquina inferior derecha 3-Introduce los datos del jugador 4- Selecciona el botón Aceptar
		V	I Vacío	V	V	V		
		V	V	V vacío	V	V		
		V	V	V	V vacío	V		
		V	V	V	V	V vacío		
		V	V	V	V	V		
EC 5.3 Insertar jugador	Validar que no existen campos inválidos	I Fr@nk	V	V	V	V	1- No permite introducir caracteres inválidos	1- Seleccionar del menú principal la opción "Jugadores " 2- Selecciona el botón Agregar ubicado en la esquina inferior derecha 3-Introduce los datos del jugador 4- Selecciona el botón Aceptar
		V	I G4rc1A	V	V	V		
		V	V	V	V	V		
		V	V	V	V	V vacío		

Tabla 23: Caso de prueba insertar torneo

Escenario	Descripción	Nombre	Fecha de Inicio	Fecha de Fin	Respuesta del sistema	Flujo central
EC 6.1 Insertar torneo	Permite al usuario crear un nuevo torneo.	V	V	V	1- Inserta en la base de datos el torneo creado por el usuario, para ello debe llenar los campos • Nombre • Fecha de Inicio • Fecha de Fin 2- Muestra el mensaje “Torneo Insertado”	1-Seleccionar del menú principal la opción "Torneos" 2-Selecciona el botón Agregar ubicado en la esquina inferior derecha 3-Introduce los datos del torneo 4-Selecciona el botón Aceptar
EC 6.2 Insertar torneo	Validar que no existen campos vacíos	I vacío	V	V	1- Muestra el mensaje “Todos los campos son obligatorios”	1-Seleccionar del menú principal la opción "Torneos" 2-Selecciona el botón Agregar ubicado en la esquina inferior derecha 3-Introduce los datos del torneo 4-Selecciona el botón Aceptar
		V	V	V		
		V	I Vacío	V		
		V	V	I vacío		
EC 6.3 Insertar torneo	Validar que no existen campos inválidos	I Ju3go$	V	V	1- No permite introducir estos caracteres. 2- Muestra el mensaje “Para los torneos por grupos la cantidad de equipos debe ser par”	1-Seleccionar del menú principal la opción "Torneos" 2-Selecciona el botón Agregar ubicado en la esquina inferior derecha 3-Introduce los datos del torneo
		V	V	V		
		V	I 26/04/17	V		
		V	V	I 20/04/17		

					3- Muestra el mensaje “La duración mínima de un partido es de 2 minutos” 4- Muestra el mensaje “La fecha de fin debe ser mayor que la fecha de inicio	4-Selecciona el botón Aceptar