UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA INFORMÁTICA

DESARROLLO DE UNA APLICACIÓN DE ANÁLISIS ESTADÍSTICO PARA EL APOYO DE ENTRENADORES DE LA NBA
DEVELOPMENT OF A STATISTIC ANALYSIS APPLICATION FOR NBA COACHES AID

Realizado por
JOSE LUIS RUIZ CASADO

Tutorizado por
RAFAEL MARCOS LUQUE BAENA
MIGUEL ÁNGEL MOLINA CABELLO

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

MÁLAGA, Septiembre 2020

Resumen:

Este trabajo fin de grado trata de solventar un problema que afecta a los entrenadores de la liga de baloncesto norteamericana: el poco tiempo de preparación para un partido de liga regular. Debido a la gran cantidad de partidos que se disputan a lo largo de la temporada regular es muy común que los equipos jueguen varios partidos en una semana, incluso en un intervalo de tres días se suelen jugar dos partidos.

La solución que se ofrece en este trabajo fin de grado es el diseño de un sistema software de apoyo a los entrenador y analistas, permitiendo facilitar el estudio de los rivales. El sistema está compuesto por un servicio web que se encargará de, en primer lugar, extraer la información de los partidos de la liga mediante técnicas de minería de datos y posteriormente analizará la información extraída para, mediante el análisis de los datos con un modelo de aprendizaje automático, generar información para poder interpretar la manera de jugar de un equipo y sus jugadores. Esta información junto a las estadísticas calculadas será almacenada en una base de datos, sobre la que posteriormente se realizarán consultas para mostrar de manera adecuada la información sobre cada jugador o equipo.

Desde una aplicación móvil diseñada para tabletas se conectará al servicio web y se obtendrá la información que será mostrada de tal manera que facilite el trabajo de los entrenadores pudiendo tener la información útil para poder preparar un partido.

Palabras claves: Minería de datos, Aprendizaje Automático, Liga de baloncesto, Apoyo a entrenadores.

Abstract:

This final year dissertation tries to solve a recurrent problem that affects the coaches of the American basketball league: the short preparation time period for a regular league match. Due to the large number of games that are played throughout the regular season it is very common for teams to play several games in a row, even in a three-day interval in which they usually play two games.

By designing a software system that, in support of coaches and analysts, facilitates the study of rivals we try to solve the problem. The system is made up of a web service that extract the information of the league's matches through data mining techniques and subsequently analysing the information extracted for, by analysing the data with a machine learning model, generate statistics to interpret the way of playing a team and its players. This information together will be stored in a database in which queries will be made to retrieve and properly display information about each player or team.

From a mobile application designed for tablets, it will be connected to the web service and the information will be displayed for each team and their players, facilitating the coach's work in the previous preparation of the matches.

Keywords: Data Mining, Machine Learning, Basketball League, Coach Support.

Índice de contenidos

CAPÍTULO 1

Introducción

La Asociación Nacional de Baloncesto [1], en adelante NBA (*National Basketball Association*), es la liga privada de baloncesto profesional de los Estados Unidos que fue fundada en 1946. Al ser una liga gestionada por una entidad privada las plazas son elegidas por la propia NBA de manera que se asignan en función de ciudades de Norte América y una entidad o grupo financiero crea o compra una marca, a esto se le conoce como franquicia. En la actualidad la liga está formada por treinta franquicias, y se dividen en dos conferencias, la este y la oeste, las cuales se encuentran a su vez divididas en tres divisiones de cinco equipos cada una. Estas divisiones se realizan por la localización geográfica de la ciudad que representa cada franquicia.

A día de hoy la NBA es considerada como la mejor liga de baloncesto del mundo, y una de las ligas deportivas con mayor impacto. Esto se debe principalmente a dos factores: la calidad de los jugadores y de los equipos es muy superior a la del resto y la inversión económica que recibe la liga y los equipos. En 2014 se firmó un nuevo contrato televisivo con las empresas *ESPN* y *TNT* por nueve años de veinticuatro billones de dólares, esto supuso un gran crecimiento en la capacidad financiera de los equipos. Esta inversión repercute de manera directa en las franquicias, ya que se llevan parte de ella debido a los derechos televisivos que poseen sobre los partidos que juegan.

Dicha inversión tuvo una importante repercusión en toda la liga, tanto es así que ese mismo año los estadios de cada franquicia se equiparon con sistemas software para el seguimiento de los jugadores durante los partidos, mediante cámaras. Esto supuso un importante cambio de paradigma en el análisis del baloncesto. El sistema de seguimiento permite obtener las coordenadas de los diez jugadores sobre la cancha y las coordenadas y velocidad de la pelota, de manera que permite un estudio más exhaustivo del impacto de los jugadores sobre el juego. Esto también implicó que las franquicias NBA incorporasen un departamento propio dedicado al análisis de datos.

1.1. Motivación

Tradicionalmente para analizar un partido de un jugador o de un equipo se utilizan estadísticas sobre cuatro aspectos positivos del propio partido cómo son los puntos anotados, las asistencias, rebotes atrapados (en ataque y en defensa) y los tiros libres realizados.

A partir de estos cuatro registros se pueden obtener estadísticas cómo el porcentaje de acierto de los tiros realizados por un jugador o por un equipo. Estás estadísticas son conocidas como *boxscore*, ya que son añadidas en una tabla, pero tienen un problema y es que no tienen en cuenta muchos aspectos del juego, sobre todo defensivos, por lo que no representan el impacto real del jugador en el partido. Es por esto que surgen unas nuevas estadísticas más avanzadas que se calcular a partir del boxscore.

Los entrenadores suelen basarse en estas estadísticas, tanto simples como avanzadas, para estudiar a sus rivales. Por norma general dentro del equipo de entrenadores, *coaching staff*, suele haber un entrenador asistente dedicado únicamente al análisis de los equipos rivales. El problema reside en lo apretado que es el calendario de la NBA. Cada equipo disputa ochenta y dos partidos de temporada regular por lo que es bastante común que se disputen partidos con únicamente un día de separación. Esto complica mucho la labor de los encargados del análisis previo de los partidos.

Muchos equipos optan por centrarse en analizar su propio juego e intentar mejorar durante la temporada regular antes que centrarse en los equipos rivales. Cuando llega la post temporada, conocida comúnmente como *playoffs* [1] hay más tiempo para la preparación previa de los partidos y se hacen exámenes mucho más exhaustivos de los rivales.

Es por esto que la motivación detrás del desarrollo de este proyecto es brindar la opción de poder analizar los rivales durante la temporada regular. Con este proyecto se pretende realizar un sistema que permita agilizar el proceso de análisis previo de cada partido. Para ello se analizarán los datos de cada partido de la liga ofrecidos por la NBA, se hablarán sobre ellos con más detalle en el capítulo 6.

1.2. Objetivos

El objetivo principal de este proyecto es desarrollar un sistema software que sea capaz de facilitar y agilizar la labor de los entrenadores encargados de analizar a los rivales antes de cada partido. Para ello se construirán estadísticas simples y avanzadas de cada jugador y cada equipo en función de los datos de cada partido proporcionado por la propia NBA. Para cumplir con este objetivo se marcan los siguiente subojetivos más precisos:

- Desarrollo de una API basada en microservicios, de manera que cada microservicio implementa una funcionalidad del sistema.
- Diseño y creación de una base de datos que alberga la información necesaria para representar a equipos y jugadores, de manera que de ambos se guarde información general y las estadísticas simples. Las operaciones de gestión y lectura de la base de datos serán incluidas en forma de microservicio.
- Desarrollo de un microservicio que calcule las estadísticas avanzadas de equipos y jugadores en base a las estadísticas simples almacenadas en la base de datos.
- Desarrollo de un microservicio que analice los ficheros de cada partido y extraiga la información para añadirla a la base de datos.

[1] Torneo al mejor de siete partidos disputados por los ocho primeros equipos de cada conferencia.

- Desarrollo de un modelo de aprendizaje automático para la predicción de partidos en base a las estadísticas avanzadas de los equipos. Este modelo debe ser incluido en un microservicio.
- Desarrollo de modelos de clusterización para cada posición, de manera que se puedan agrupar los jugadores con un estilo de juego similar. También debe ser añadido a un microservicio.
- Desarrollo de una aplicación Android que consuma la API y muestre de manera clara y sencilla la información al usuario.

Para calcular las estadísticas simples se utilizarán los ficheros de eventos de cada partido que proporciona la NBA. Un evento es una acción que sucede en un partido de la NBA, por ejemplo un intento de canasta, un robo o una falta. Este fichero es proporcionado en formato CSV, que es un formato utilizado para representar datos en forma de tabla. Se analizará el fichero y se aplicarán técnicas para la extracción de información para poder calcular las estadísticas simples a partir de los eventos del partido. Una vez se hayan calculado la estadísticas simples de los jugadores y equipos que participan en el partido se realizarán los cálculos para obtener las estadísticas avanzadas. Los valores calculados serán añadidos a la información de cada jugador y cada equipo y almacenados en una base de datos.

La NBA también ofrece los datos de posicionamiento de los jugadores de cada partido en un fichero con formato JSON. Es un formato de texto sencillo que es utilizado muy frecuentemente para el intercambio de datos a través de páginas webs. Estos fichero incluyen la posición de los diez jugadores que se encuentran en la cancha de baloncesto durante el partido, las medidas son tomadas con un intervalo de tiempo de diez segundos. Estos datos son muy interesantes para analizar la interacción de un jugador individual en el partido. De estos ficheros el sistema extraerá información posicional sobre cada jugador para poder calcular el mapa de calor de un jugador. Estos mapas de calor se generarán para la defensa y para el atacante y proporciona información sobre en qué zonas del campo se siente cómodo un jugador y permite al entrenador la capacidad de diseñar movimientos para hacer que este jugador no esté en sus zonas favorables.

Se pretende implementar una función que dote al sistema de la capacidad de realizar un informe para un partido. En este informe se enfrentarán las estadísticas avanzadas de los equipos que disputan el partido para facilitar la comparación de aspectos claves que pueden decantar el enfrentamiento. También se mostrará en el informe una predicción del partido. Esta predicción se realizará mediante un modelo de aprendizaje automático en función de las estadísticas avanzadas de ambos equipos.

Otra técnica de aprendizaje automático que se utilizará en este proyecto es el agrupamiento de los jugadores según sus estadísticas. Esta técnica permite segmentar los jugadores sobre las posiciones relativas que ocupan basadas en el estilo de cada jugador, agrupando los jugadores de estilos similares. Esto se debe a que actualmente las cinco posiciones tradicionales no son tan descriptivas sobre los jugadores, por ejemplo, tanto Stephen Curry como Patrick Beverley son bases, pero Stephen Curry es un jugador puramente ofensivo y reconocido como uno de los mejores triplistas de la historia, mientras que Patrick Beverly ha conseguido basar su dominio en la defensa. Este agrupamiento se hará para cada una de las posiciones principales, esto permite dar unos pesos distintos a ciertas estadísticas más generalizada para una posición. Esto se debe a que si un base consigue dar cinco

asistencias por partido no tiene la misma importancia que si un pívot consigue dar las mismas asistencias por partido, para un base este dato es mediocre mientras que para el pívot es una cualidad bastante identificativa.

Uno de los grandes objetivos de este proyecto es poder mostrar adecuadamente toda la información calculada que se describe en los párrafos anteriores. La interfaz diseñada debe ser amigable y sencilla, teniendo en cuenta que el público objetivo, los entrenadores de la NBA, puede no estar muy familiarizado con las nuevas tecnologías. Es por esto que se eligió el formato tablet para poder mostrar la información de una manera precisa y concisa. También hay que tener en cuenta que se debe mostrar toda la información, pero no al mismo tiempo, para poder ofrecer un mejor entendimiento de la manera de jugar de cada equipo y jugador de la liga.

También se debe garantizar que las conexiones que se realicen sean seguras. Tanto la conexión que realiza el servidor con la base de datos como la conexión entre cliente y servidor. La conexión del servidor con el cliente se realizará mediante peticiones HTTP, que es un protocolo para el intercambio de información entre cliente y servidor web. Para realizar la conexión del servidor con la base datos se utilizarán los conectores propios que ofrecen los distintos motores de base de datos.

1.3. Estructura de la memoria

En esta sección se explica la división en capítulos y secciones de la memoria, explicando el contenido que alberga cada uno.

- **2. Estado del arte**: en este capítulo se contextualiza el proyecto dentro del marco histórico y tecnológico actual.
  - **2.1. Estudio de Mercado**: en esta sección se realiza un estudio sobre las aplicaciones software existentes relacionadas con entrenadores de baloncesto y estadísticas de la NBA.
- **3. Metodologías y Planificación**: en este capítulo se trata la gestión del proyecto, desde la metodología elegida para el desarrollo del mismo hasta la planificación realizada para llevar a cabo el proyecto.
  - **3.1. Metodología**: esta sección explica que metodología de desarrollo software se urilizará durante la realización de este proyecto. Se explican los motivos por los que se selecciona la metodología.
  - **3.2. Planificación**: esta sección se utilizará para realizar una planificación del proyecto en base a la metodología utilizada.
  - **3.3. Tecnologías usadas**: en esta sección se analizan los lenguajes de programación, entornos de trabajo, librerías, tecnología de base de datos y entornos de desarrollo que se utilizaran para el desarrollo del sistema, teniendo en cuenta la arquitectura.
- **4. Análisis de Especificaciones**: este capítulo recoge las especificaciones del sistema a desarrollar.
  - **4.1. Análisis de Requisitos**: en esta sección se realiza un estudio de los requisitos, tanto funcionales como no funcionales, que debe satisfacer el sistema.
  - **4.2 Casos de Uso**: en esta sección se realiza la especificación de la comunicación del usuario con el sistema, mediante el diseño de los casos de uso.
- **5. Diseño del Sistema**: en este capítulo se establece el modelo arquitectónico ideal para el sistema teniendo en cuenta el análisis de especificaciones previamente realizado.
  - **5.1. Arquitectura del sistema**: en esta sección se define la arquitectura elegida para el proyecto.
  - **5.2 Identificación de subsistemas**: en esta sección se definen cuáles son los distintos subsistemas existentes en este proyecto.
  - **5.3 Revisión de casos de uso**: sección en la que se identifica los subsistemas involucrados en cada caso de uso.
  - **5.4 Estructura de la base de datos**: en esta sección se define la base de datos necesaria para el sistema.
- **6. Análisis de los Datos**: en este capítulo se realiza un estudio de los datos de entrada para entender el formato en el que se reciben y cómo se calculan las estadísticas en base a estos datos.
  - **6.1. Ficheros de entrada**: en esta sección se analizan los ficheros que utiliza el sistema como entrada.

    - **Fichero Play-By-Play**: en esta sección se habla sobre el formato de uno de los tipos de ficheros de entrada. También se explica el significado de los datos que alberga.
    - **Fichero Tracking Data**: se explica el otro tipo de fichero de entrada en el que se encuentran los datos de posicionamiento, explicando el formato en el que es distribuido.
  - **6.2. Extracción de información**: en esta sección se define de qué manera se extraen la información necesaria para alimentar la base de datos.
  - **6.3. Selección de modelo de predicción de partidos**: a lo largo de esta sección se realiza un estudio con los resultados obtenidos con distintos modelos para la predicción de partidos. Este estudio se utiliza para seleccionar el modelo que mejor se ajuste a los datos para la predicción.
  - **6.4. Clusterización de jugadores por posición**: en esta sección se realiza el estudio de la clusterización de los jugadores de cada una de las cinco posiciones. En este proceso se identifican los clústers formados y se analizan para establecer el estilo de los jugadores de cada clúster.
- **7. Diseño de interfaces de Usuario**: este capítulo recoge el diseño de las interfaces de usuario del cliente. Se ofrecerá una explicación sobre las distintas gráficas y estadísticas que se muestran y una explicación de que ofrecen a un entrenador de la NBA.
- **8. Implementación**: capítulo que recoge la implementación realizada del sistema.
  - **8.3. Desarrollo de la API**: en esta sección se explica cómo ha sido la implementación en código de la API.
  - **8.2. Aplicación Android**: esta sección explica cómo ha sido el desarrollo del cliente Android
- **9. Conclusiones y Líneas Futuras**: en este último capítulo de la memoria se recogen los conocimientos adquiridos durante la realización del proyecto y se compararan los objetivos con el resultado obtenido.
  - **9.1. Dificultadas solventadas durante el desarrollo**: en esta sección se analizan los problemas que se han encontrado durante el desarrollo del proyecto.
  - **9.2. Líneas fututas**: se explica cómo puede evolucionar el proyecto en el futuro.
  - **9.3. Conclusión**: se realiza una valoración general sobre el proyecto.

CAPÍTULO 2

Estado del arte

Hoy en día es innegable que, debido al auge en importancia que han tenido los datos, el análisis de datos es uno de los campos más avanzados y conocidos en varios ámbitos que se interconectan como son la informática, las matemáticas y la estadística. Hemos llegado a este punto a causa del cambio de la perspectiva con la que se entienden los datos, antes se entendían como información mientras que actualmente se entienden y se tratan como materia prima. ¿Por qué se tratan como materia prima? Hay tres aspectos claves para responder esta pregunta:

- Es una fuente **inagotable**, ya que el crecimiento de los datos que se crean es exponencial.
- Son **manejables**, ya que en la actualidad hay un sin fin de herramientas que permiten el procesado y mantenimiento de una cantidad ingente de información.
- Es un sector que está **casi inexplorado**, aunque bien es cierto que actualmente se está empezando a aplicar a la mayoría de ámbitos posibles.

Naturalmente, también ha evolucionado de manera exponencial en el mundo del deporte, y de entre todos los deportes la NBA ha conseguido establecerse como una abanderada dentro del análisis de datos. Tanto es así que de manera anual celebran un *Hackathon* [1]. En este Hackathon participan estudiantes, estadísticos, desarrolladores e ingenieros, los equipos participantes crean herramientas para intentar solventar problemas importantes y desafiantes a los que se enfrenta la NBA[2]. Con actividades como ésta, la NBA pretende seguir siendo líder en el sector del análisis de datos fomentando estás técnicas para mejorar a todos los niveles.

Pero no solo la NBA como institución fomenta el análisis de datos, desde los equipos que participan en la liga es cada vez más común que, lo que anteriormente era un tema casi tabú, se reconozca que los éxitos conseguidos son gracias a analíticas realizadas sobre el baloncesto. Un claro ejemplo de esta situación es Daryl Morey gerente del equipo Houston Rockets quien posee formación en estadística y ha conformado el equipo en función de los análisis que él y su departamento de analistas de datos han realizado, llegando a ser serios aspirantes al título, sobretodo en la temporada 2017-2018. Los Houston Rockets han conseguido maximizar el tiro de tres puntos hasta tal punto que temporada tras

[1] Término utilizado para referirse a un encuentro de programadores cuyo objetivo es desarrollo colaborativo de software

temporada están entre los equipos que más tiros de tres intentados. Esto se debe a que según los análisis que han realizado, el tiro de tres puntos es el tiro que mayor beneficio sobre el riesgo que supone aporta.

2.1. Estudio de Mercado

Hay muchas aplicaciones y servicios para el análisis de partidos de baloncesto, en gran parte gracias a los esfuerzos de la NBA por globalizar la liga y como consecuencia el baloncesto, por lo que esta sección se procede a realizar un estudio sobre las alternativas más usadas y conocidas que existen actualmente.

Antes de empezar es necesario establecer una división entre los distintos tipos de software que pueden encontrarse:

- **Aplicaciones de gestión de equipos**: son aplicaciones que permiten gestionar aspectos relacionados con un equipo de baloncesto, como gestión de plantilla, gestión de eventos, etc.
- **Software de análisis**: permiten realizar análisis sobre distintos formatos y aspectos de un partido de baloncesto.

2.1.1. Aplicaciones de gestión de equipos

Este tipo de aplicaciones permiten llevar acabo una gestión de un equipo de baloncesto, ofreciendo la posibilidad de añadir los jugadores de la plantilla a la aplicación dando la posibilidad de realizar labores de gestión sobre los jugadores incluidos.

Otra funcionalidad que suelen incluir es la posibilidad de añadir los eventos que van sucediendo durante un partido, ofreciendo ciertas estadísticas de los jugadores después del partido.

Este tipo de aplicaciones no son competencia directa, ya que se centran más en labores de gestión y no tanto en análisis de los datos. Aún así es importante conocer las aplicaciones punteras incluidas en esta especificación para tener en cuenta ciertas labores de gestión que debe llevar a cabo el cuerpo técnico de un equipo. A continuación se explicarán las aplicaciones más usadas que encajen con la definición de aplicaciones de gestión de equipos.

SportEasy

SportEasy[3] es una aplicación que incluye varios deportes, entre ellos el baloncesto, con varios planes con distintos precios en función de las características que se deseen utilizar. Permite la gestión de evento como los entrenamientos, los partidos y las competiciones. También cuenta con un módulo para la comunicación con los jugadores y la gestión de asistencia. También permite la recogida de estadísticas de un partido y una sección con estadísticas generales. Esta aplicación es bastante usada en Norte América en equipos no profesionales y semi profesionales.

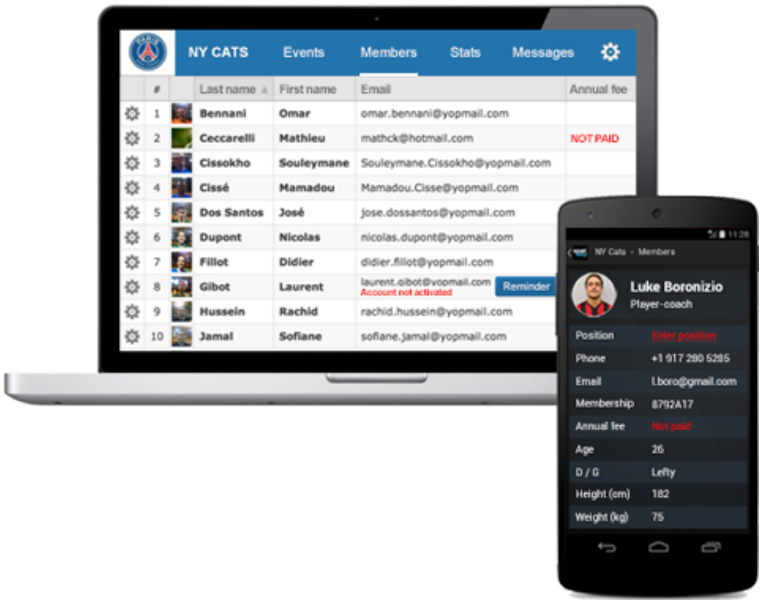


Figura 2.1: Interfaz de la aplicación SportEasy[3].

Easy Stats Basketball

Easy Stats Basketball[4] es una aplicación de gestión de equipos centrada en la recogida de estadísticas durante el trascurso de un partido de baloncesto para luego generar el boxscore del partido. También realiza análisis de tendencias y flujo del juego. Está muy bien ubicada en el mercado estadounidense y a menudo es utilizada por seguidores de la NBA para obtener ciertas estadísticas sobre algún partido.

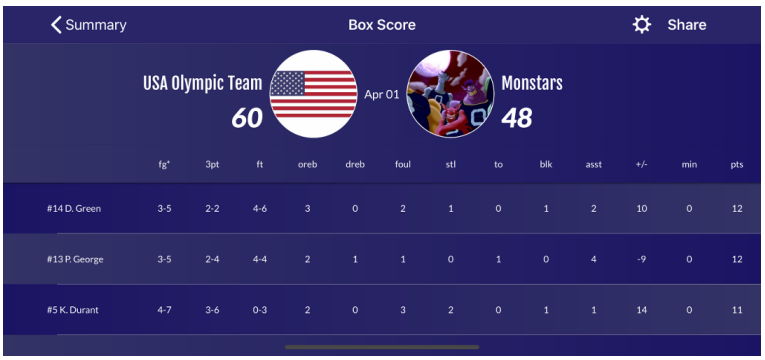


Figura 2.2: Boxscore generada por Easy Stats Basketball[4].

2.1.2. Software de análisis

Este tipo de software se centran en analizar las estadísticas de un partido de baloncesto para generar estadísticas más avanzadas sobre un partido o un jugador. La gran parte de este tipo de software son aplicaciones móviles centradas en la NBA, por lo que sí suponen una competencia más directa.

A continuación se nombran algunas aplicaciones y software relevantes en el análisis de partidos de baloncesto.

Basket Stats Assistance

Basket Stats Assistance es una aplicación móvil que permite añadir los eventos de los partidos de la NBA en vivo, calculando las estadísticas que añade posteriormente en un boxscore. Es una aplicación bastante limitada ya que no calcula ningún tipo de estadística avanzada.

Figura 2.3: Boxscore generada por Basket Stats Assistance

Breakthrough Stat

Breakthrough Stat[5] es una aplicación, diseñada para dispositivos con el sistema operativo IOS, que permite realizar un seguimiento en vivo de las estadísticas de un partido, que se deben ir añadiendo durante el transcurso del mismo. Ofrece un boxscore en vivo con los datos actualizados con los eventos que se van añadiendo, una vez finalizado el partido ofrece un reporte, que es accesible en cualquier momento.

Se centra principalmente en la recogida de datos del partido, pudiendo añadir la posición del jugador que realiza un tiro, mientras que los reportes que generan no incluyen estadísticas avanzadas.

Figura 2.4: Interfaz de la aplicación Breakthrough Stat[5].

Hudl assist for Basketball

Hudl assist for Basketball[6] es una aplicación que realiza un análisis sobre los partidos de baloncesto. Para que la aplicación realice este análisis el vídeo del partido ha de ser subido a la aplicación, y en un intervalo de veinticuatro horas se recibe el informe sobre el partido. Este informe contiene un boxscore y algunas estadísticas avanzadas sobre eficiencia de tiros y eficiencias avanzadas.

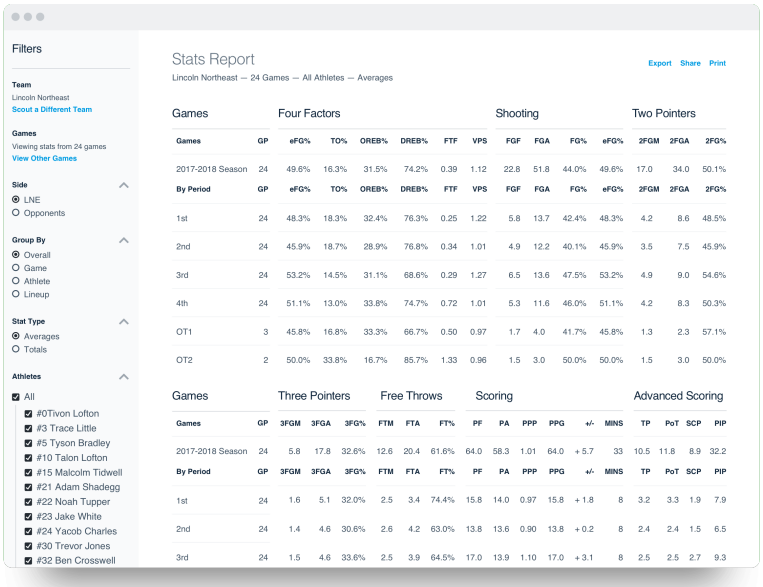


Figura 2.5: Reporte generado por Hudl assist for Basketball[6].

NBA Official App

La NBA tiene su propia aplicación móvil que permite el seguimiento en vídeo de los partidos y contiene un calendario para seguir los partidos durante todo el año. También ofrece un boxscore para cada partido que se juega. Esta aplicación está sobre todo orientada a los seguidores de la liga.

	CHA														ATL
JUGADOR	P	Min	PTS	REB	ASI	ROB	TAP	TAPC	TCC	TCI	TC%	3PC	3PI	3P%	TLC
Devonte' Graham	SG	46:05	27	4	10	4	0	0	9	26	34,6%	4	14	28,6%	5
Terry Rozier	PG	43:49	40	4	3	1	0	2	15	26	57,7%	8	13	61,5%	2
Miles Bridges	SF	26:12	4	4	4	0	1	0	2	10	20%	0	5	0%	0
Cody Zeller	C	22:49	10	7	2	0	0	1	3	5	60%	0	0	0%	4
P.J. Washington	PF	16:27	2	2	0	0	1	2	1	6	16,7%	0	2	0%	0
Caleb Martin		36:31	23	0	4	3	2	0	8	10	80%	5	6	83,3%	2
Cody Martin		35:26	11	4	5	0	0	0	3	8	37,5%	1	5	20%	4

Figura 2.6: Boxscore de la aplicación oficial de la NBA.

Sport Vu

Sport Vu[7] es socio oficial de la NBA y se encarga de proveer a la liga con la tecnología necesaria para la recolección de los datos de seguimiento de cada partido. Ofrecen reportes que incluyen estadísticas avanzadas sobre partidos en concreto bajo demanda, es decir, es necesario contactar con la empresa para poder obtener el informe. También realizan análisis en vivo de un partido directamente sobre la transmisión vídeo del mismo.

Es evidente que Sport Vu es un competidor directo con el sistema propuesto en este trabajo fin de grado, aunque tienen un enfoque más tecnológico y de cara al público que el que se pretende con el sistema definido en esta memoria.

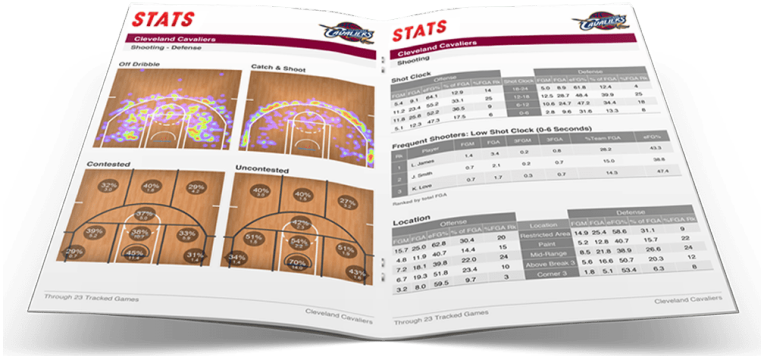


Figura 2.7: Informe generado por Sport Vu para un partido de NBA[7].

CAPÍTULO 3

Metodología y Planificación

En este capítulo se selecciona la metodología utilizada para el desarrollo del proyecto, tratando de expresar los motivos para la selección de dicha metodología. También se realizará la planificación que se seguirá para el desarrollo del proyecto. La planificación incluirá también una descripción de las tareas en las que se dividirá el desarrollo del proyecto.

3.1. Metodología

Dada la naturaleza de este proyecto se ha establecido que la metodología que mejor encaja es Scrum ya que permite el desarrollo colaborativo y en equipo mientras se van realizando pequeñas entregas para finalizar con la entrega del producto final. Scrum permite llevar a cabo una gestión regular del proyecto, dotándolo de flexibilidad y adaptación frente a los problemas y es una de las metodologías mas usadas hoy en día ya que una de sus principales características es la productividad y la calidad.

Scrum está basado en *sprints*, un sprint es el período de tiempo en el que se lleva a cabo el trabajo. La duración de los sprints debe ser constante, para este proyecto la duración base de un sprint se fija en 2 semanas. Al acabar un sprint se realizará una reunión de control con el tutor en la que se evaluará el trabajo realizado en el sprint y se definirá el trabajo a realizar en el siguiente sprint y se planificará.

Dentro de Scrum existen distintos *roles* que interactúan entre sí para llevar a cabo el desarrollo del proyecto. Los roles que intervendrán en el desarrollo de este proyecto son:

- **Scrum Master**. Es el encargado de conseguir que se alcance el objetivo de cada sprint. No organiza el equipo, simplemente trata de actuar a modo de guía para centrar los esfuerzos en las materias importantes del sprint que se está llevando a cabo. Este rol será llevado a cabo por el tutor del trabajo de fin de grado, Rafael Marcos Luque Baena.
- **Equipo de Desarrollo**. Son los encargados de realizar el desarrollo y los responsables de entregar el producto. En este caso sería yo, Jose Luis Ruiz Casado, el encargado de asumir este rol.

La selección de Scrum se debe a que dota al proyecto de capacidad de reacción ante requisitos cambiantes debido a factores como el cambio de necesidades o la evolución del mercado, además el marco de trabajo está pensado para adecuarse a nuevas exigencias que suelen darse en los proyectos complejos. También permite reducir el tiempo de espera hasta poder utilizar y testar las funcionalidades del proyecto, aunque éste esté sin completar. Otro motivo por el que se selecciona Scrum es por la productividad de la hace gala, debido a la eliminación de la burocracia. También permite cierta reducción de riesgos mediante la elaboración de las funcionalidades de mayor importancia en primer lugar. Conociendo la velocidad estimada de avance a lo largo de los distintos sprint que conforman el proyecto permite despejar riesgos de manera efectiva y anticipada.

3.2. Planificación

En esta sección se establece una división en fases del proyecto y la planificación para cada una de las fases. Las fases que se establecen para la división del proyecto se muestran a continuación en una tabla, en la que se muestra el nombre de la fase y las horas que se van a dedicar para completarla.

Fases del proyecto	
Fase	Horas
Obtención de Requisitos	16
Elaboración de Diagramas	10
Estudio de los datos sobre los que actúa el sistema	15
Diseño de las interfaces de usuario	25
Diseño de pruebas	30
Implementación:	Total: 170
Diseño y desarrollo del proceso de minería de datos	34
Diseño de modelo de aprendizaje automático	40
Desarrollo del servicio Web	34
Implantación del modelo de aprendizaje automático	28
Desarrollo de aplicación móvil	34
Pruebas de implantación	30
Total	296

Tabla 3.1: Tabla de fases del proyecto.

Para poder encajar las fases del proyecto en el marco de trabajo Scrum es necesario establecer cuántas horas se van a dedicar a la semana para cuadrar cada fase con uno o más sprints. Se estima que a la semana se le dedicarán 20 horas, de manera que estableciendo la duración de los sprints en dos semanas cada sprint conlleva 40 horas. A continuación se muestra una tabla con las fases del proyecto y los sprints en los que se incluye cada una.

Sprints del proyecto	
Fase	Sprint
Obtención de Requisitos	1
Elaboración de Diagramas	1
Estudio de los datos sobre los que actúa el sistema	2
Diseño de las interfaces de usuario	2
Diseño de pruebas	2
Diseño y desarrollo del proceso de minería de datos	3
Diseño de modelo de aprendizaje automático	4
Desarrollo del servicio Web	5
Implantación del modelo de aprendizaje automático	6
Desarrollo de aplicación móvil	7
Pruebas de implantación	8

Tabla 3.2: Tabla de fases y sprints correspondientes del proyecto.

Como se puede observar en la Tabla 3.2 el proyecto constará de 8 sprints en los cuales se irán completando las fases del proyecto. La memoria se irá completando en paralelo, añadiendo cada parte en función del sprint que se esté completando.

3.3. Tecnologías usadas

A lo largo de esta sección se explican las tecnologías que se utilizarán para la implementación de las distintas partes del proyecto. La tecnología seleccionada para cada parte debe adecuarse a la arquitectura, que es definida en la sección 5.1.

Entre las tecnologías que hay que seleccionar se encuentra un sistema que permita la gestión de la base de datos. Teniendo en cuenta la arquitectura propuesta (ver 5.1) MySQL [8] es el sistema gestor que más se adecua. MySQL es uno de los gestores más utilizados y reconocidos en la actualidad. Es de código abierto y por tanto hay una gran comunidad alrededor, esto permite que sea mucho más fácil solucionar los problemas. También se comporta de gran manera en entorno web, por lo que se ajusta especialmente bien a la arquitectura de microservicios web del sistema. Una de las principales características que hace que se decante el uso de este gestor es el motor de consultas de alto rendimiento que posee. Este motor permite realizar inserciones de datos en las tablas en poco tiempo y además cuenta con multitud de funciones web especializadas. MySQL posee una arquitectura única para el motor de almacenamiento que permite mejorar el rendimiento de la base de datos en función de la configuración que se realice.

Como lenguaje base de programación se selecciona Python [9]. Python es uno de los lenguajes de moda debido a su enfoque multiparadigma, es decir, combina propiedades de distintos paradigmas de programación. Principalmente es tratado como un lenguaje orientado a objetos pero incorpora características de la programación imperativa, funcional, procedural y reflexiva. Pero el principal motivo por el que ha sido seleccionado es la flexibilidad que ofrece debido a los entornos de trabajo y librerías que existen. Es por esto que Python puede brillar en el desarrollo web y en tareas de análisis de datos.

Para el desarrollo de la API REST se utilizará el entorno de trabajo Django [10]. El

desarrollo en Django es muy rápido debido a que incorpora múltiples herramientas para facilitar tareas tediosas del desarrollo. También incorpora muchos paquetes con funcionalidades ya desarrolladas haciendo posible una rápida implantación, ya que tan solo sería necesario adaptarla. Otro punto bastante importante para la selección de este entorno es la interfaz para acceso a la base de datos, que es muy completa y facilita la realización de consultas.

Una librería importante de Python que se va a utilizar en el desarrollo es Scikit-Learn [11]. Esta librería incluye multitud de funciones y algoritmos para el aprendizaje automático y permite la implantación de un modelo de aprendizaje automático de manera sencilla. Está construida sobre otras librerías de Python muy conocidas y utilizadas que son NumPy, Pandas y Matplotlib. NumPy [12] es una librería que mejora los vectores y arrays. Por su parte, Pandas [13] es una librería que permite el análisis y la manipulación de datos ofreciendo estructuras de datos y operaciones para modificarlas, mientras que Matplotlib [14] permite generar gráficos a partir de datos almacenados en listas o arrays propios de NumPy.

Como entorno de desarrollo para Python se utilizará PyCharm. Este entorno es el más utilizado hoy en día para Python ya que cuenta con multitud de herramientas para facilitar el desarrollo y posee con un marco de trabajo integrado para Django. También hace uso de herramientas científicas que facilitan la previsualización de datos y gráficas.

El cliente será desarrollado para dispositivos Android se utilizará el lenguaje de programación Kotlin [15]. Es un lenguaje de tipado estático que es ejecutado sobre una máquina virtual Java, actualmente cuenta con soporto oficial de Google [16]. además del apoyo de los desarrolladores del propio lenguaje, que son JetBrains [1]. Todo esto hace que el ecosistema actual se mantenga de forma estable y Kotlin sea considerado como el lenguaje propio de Android para el futuro.

La característica principal por la cual se utiliza Kotlin es que es un lenguaje muy seguro, en el sentido de que el compilador ayuda a la detección de errores de ejecución pero en tiempo de compilación, por lo que es obvio que nunca llega a producirse en tiempo de ejecución. Además, como el compilador asume varias tareas el código necesario para llevar a cabo una tarea es mínimo, reduciendo así los fallos potenciales derivados del código erróneo.

Para el desarrollo de Kotlin se utiliza el entorno AndroidStudio, que es un entorno diseñado especialmente para el desarrollo Android. Permite el diseño visual de interfaces y facilita el compilado de las aplicaciones. También incorpora un emulador de dispositivos Android integrado que permite ejecutar la aplicación y comprobar su correcto funcionamiento.

Se utilizará Github para la gestión del proyecto y el control de versiones del código del desarrollo del proyecto. La gestión de proyectos que ofrece se basa en repositorios online dónde se añade el código software para controlar las versiones. Se utilizarán dos repositorios para este proyecto, uno para el cliente y otro para los microservicios.

[1]Empresa encargada del desarrollo de herramientas como IntelliJ, AndroidStudio y Pycharm entre otras.

CAPÍTULO 4

Análisis de Especificaciones

A lo largo de este capítulo se describe el comportamiento que debe seguir el sistema que se va a desarrollar mediante la definición de requisitos, casos de uso y diagramas de flujo. No solo se centra en la enumeración y descripción de funcionalidades que se implementarán, se tratará de explicar el uso que se debe hacer y la interacción entre el usuario y el sistema, así como el comportamiento interno para llevar a cabo las distintas operaciones que conforman las funcionalidades disponibles. El objetivo de este capítulo es definir cómo actúa para permitir en capítulos posteriores poder realizar el diseño del sistema, y poder seleccionar las tecnologías más adecuadas para el desarrollo.

4.1. Análisis de Requisitos

En esta sección se realiza el estudio de las necesidades que pueden llegar a tener los entrenadores de la NBA para tratar de llegar a una correcta definición de los requisitos del sistema.

Hay que diferenciar dos tipos de requisitos:

- **Funcionales**: definen una función del sistema, entendiendo por función un conjunto de entradas, comportamientos y salidas.
- **No Funcionales**: imponen restricciones en el diseño o la implementación, son propiedades o cualidades propias del sistema.

Para cada requisito se ofrecerá un código para facilitar su identificación y una descripción para identificar que trata de especificar dicho requisito.

4.1.1. Requisitos Funcionales

Un requisito funcional puede ser un cálculo, un detalle técnicos, manipulación de dato u otro tipo de funcionalidades específicas que el sistema debe cumplir. Para cada requisito funcional se establece un código formado de la siguiente forma: $RFXX$, dónde la parte XX es un código numérico formado por dos números.

Calcular estadísticas simples sobre jugadores y equipos - RF01

Se calculan estadísticas sobre los jugadores y equipos generalizando la actuación de cada partido. Estas estadísticas se extraen directamente de los datos de cada partido y se calculan el promedio por partido. Las estadísticas simples que se calcularán son las siguientes:

- Puntos por partido.
- Asistencias por partido.
- Rebotes ofensivos por partido.
- Rebotes ofensivos por partido.
- Rebotes por partido.
- Tapones por partido.
- Robos por partido.
- Pérdidas por partido.
- Faltas por partido.
- Tiros de campo intentados por partido.
- Tiros de campo anotados por partido.
- Porcentaje de acierto en tiros de campo.
- Triples intentados por partido.
- Triples anotados por partido.
- Porcentaje de acierto en triples.
- Tiros libres intentados por partido.
- Tiros libres anotados por partido.
- Porcentaje de acierto en tiros libres.
- Pases intentados por partido.
- Pases acertados por partido.

Calcular estadísticas avanzadas sobre jugadores y equipos - RF02

También se calculan estadísticas avanzadas sobre jugadores y equipos. Por lo general el cálculo de estas estadísticas se realizan sobre las estadísticas simples. Las estadísticas avanzadas que se calcularán son las siguientes:

- Rating ofensivo. Esta estadística habla sobre la cantidad de puntos que un equipo o un jugador anota por cada intento. Cuánto más se acerque a 100 este valor mayor es la efectividad a la hora de anotar canasta.

- Rating defensivo. Es igual que el rating ofensivo pero a la hora de defender, indica la cantidad de puntos que un equipo o un jugador permite por cada intento. Al contrario que el rating ofensivo, cuanto más se aleje el valor de 100 más efectiva es la defensa.
- Net rating. Es la diferencia entre el rating ofensivo y el rating defensivo. Un net rating positivo indica que tanto el ataque como la defensa son efectivos, mientras que un net rating negativo es indicativo de que existe un déficit en el ataque o en la defensa.
- Floor porcentaje. Este valor indica el porcentaje de las posesiones ofensivas que acaban con al menos un punto. Está muy relacionado con el raiting ofensivo.
- Ritmo. Esta estadística indica el número de posesiones a las que juega un equipo o un jugador a lo largo de cuarenta y ocho minutos.
- Porcentaje real de tiro. Este porcentaje intenta medir la precisión de tiro de un jugador o equipo teniendo en cuenta los tiros de tres y dos puntos y los tiros libres realizados.
- Porcentaje de tiros de campo efectivo. Esta estadística es el porcentaje de tiro pero dando peso a los distintos tiros, siendo los tripes pesados con un punto más que los tiros de dos puntos.
- Ratio de tiros libres intentados. Habla sobre el número de tiros libres intentados por cada tiro de campo realizado.
- Ratio de triples intentados. Indica el porcentaje que suponen los tiros de tres puntos sobre el total de todos los tiros de campo realizados.
- Porcentaje total de rebotes ofensivos. Este porcentaje indica una estimación de los rebotes ofensivos atrapados sobre el total que ha habido disponibles.
- Porcentaje total de rebotes defensivos. Este porcentaje indica una estimación de los rebotes ofensivos atrapados sobre el total que ha habido disponibles.
- Porcentaje de tapones. Es una estimación del porcentaje de tiros de campo realizado por el rival que han sido bloqueados por el equipo o jugador.
- Porcentaje de pérdidas. Esta estadística calcula calcula las pérdidas cometidas por cada cien posesiones.
- Porcentaje de asistencias. Para los jugadores esta estadística indica la cantidad de asistencias que ha dado sobre el total de asistencias de su equipo. Para los equipos significa la cantidad de canastas que ha sido realizadas mediante asistencia.
- Ratio de asistencias pérdidas. Es el número de asistencias realizadas frente al número de pérdidas cometidas
- Porcentaje de robos. Es el porcentaje de robos que se realizan sobre el total de posesiones del rival.
- Porcentaje de uso. Esta estadística se calcula únicamente para los jugadores e indica el volumen de uso de un jugador en las jugadas ofensivas de su equipo, es decir el número de posesiones ofensivas que acaban en tiro, pérdida o falta a favor en el que participa el jugador.

En el capítulo 6 se especificará de manera más concreta qué información ofrece cada estadística a un entrenador y cómo se calcula.

Lectura de los equipos - RF03

Se debe mostrar toda la información almacenada sobre los equipos de manera que sea comprensible para el usuario y ayude a la compresión de los puntos fuertes y los puntos débiles de cada equipo.

Lectura de jugadores - RF04

Al igual que con los equipos, la información de los jugadores de cada equipo debe mostrarse adecuadamente para facilitar la comprensión táctica del juego de cada uno.

Análisis de los datos de los partidos - RF05

Es evidente que para poder calcular las estadísticas necesarias el sistema debe analizar los datos de cada partido. Se deben analizar las fuentes de datos, tanto los ficheros play-by-play como los datos de seguimiento, para obtener los datos que hacen posibles el cálculo de las estadísticas.

Conexión con la base de datos - RF06

El sistema debe conectarse con la base de datos para poder recuperar la información almacenada en la base de datos. La conexión con la base de datos debe ser segura y estable.

Informe de partidos - RF07

El sistema realizará un informe del partido que se disputará con el equipo que seleccione el usuario. Se mostrarán las estadísticas avanzadas de ambos equipos enfrentándolas para dar una visón al usuario de los aspectos en los que es mejor el rival. Además, el sistema realizará una predicción del partido mediante técnicas de aprendizaje profundo. Esta predicción se hará teniendo en cuenta las estadísticas avanzadas de ambos equipos y determinará cuál de los dos equipos saldrá victorioso del encuentro.

Selección de equipo - RF08

Es necesario que el usuario seleccione el equipo al que pertenece.

4.1.2. Requisitos no funcionales

Los requisitos no funcionales deben entenderse como aquellas características que hacen que el producto final sea más seguro, fiable, usable o atractivo. El código utilizado para los requisitos no funcionales es de la forma *RNFXX*, dónde *XX* es un número de dos cifras.

Rendimiento - RNF01

El rendimiento para este tipo de sistemas es indispensable y debe ser adecuado para los dispositivos en los que se vaya a ejecutar.

Disponibilidad - RNF02

Al ser un sistema conectado a una red se debe asegurar su disponibilidad a cualquier hora.

Fiabilidad - RNF03

Un sistema que busca el apoyo a la toma de decisiones debe ser fiable para que el apoyo tenga sentido. Las estadísticas calculadas deben ser precisas y evaluadas para su validación.

Usabilidad - RNF04

El sistema debe ser fácil de usar para cualquier entrenador o analista de la NBA. El sistema debe ser también atractivo y lo suficientemente sencillo para facilitar el uso.

Diseño Material Design - RNF05

El diseño del sistema debe basarse en la normativa *Material Design*[17], que está enfocada a la visualización del sistema Android.

4.2. Casos de Uso

Los casos de uso permiten describir una acción o actividad que se pueden realizar en el sistema. Los casos de uso se definen con diagramas, donde se definen las actividades que deberá hacer el usuario o el sistema para llevar a cabo algún proceso. Los personajes o entidades que actúan e interactúan en un diagrama se denominan actores.

En definitiva, un caso de uso es una secuencia de interacciones que se desarrollan entre el sistema y sus actores en respuesta al evento inicial lanzado por el actor principal. Los diagramas sirven para especificar la comunicación y el comportamiento del sistema mediante la interacción que este tiene con los usuarios o con algún componente del propio sistema, es decir la relación entre los usuarios y el sistema.

Los diagramas de casos de usos se utilizan para ilustras los requisitos funcionales del sistema, ya que muestran cómo reacciona el sistema los eventos que se producen dentro del propio sistema. Para cada diagrama se establece un código de la forma *CUXX*, donde *XX* es un número de dos cifras, y una descripción verbal de la acción que modela.

4.2.1. CU01 - Selección de equipo

Este caso de uso modela la selección de equipo que un usuario debe realizar antes de comenzar a utilizar la aplicación. Se mostrarán todos los equipos de la NBA divididos por conferencia y división. Una vez que el usuario seleccione un equipo esta selección se almacena en el sistema. Para poder mostrar los equipos es necesario realizar una consulta sobre la base de datos para obtener todos los equipos que se encuentran los sistemas.

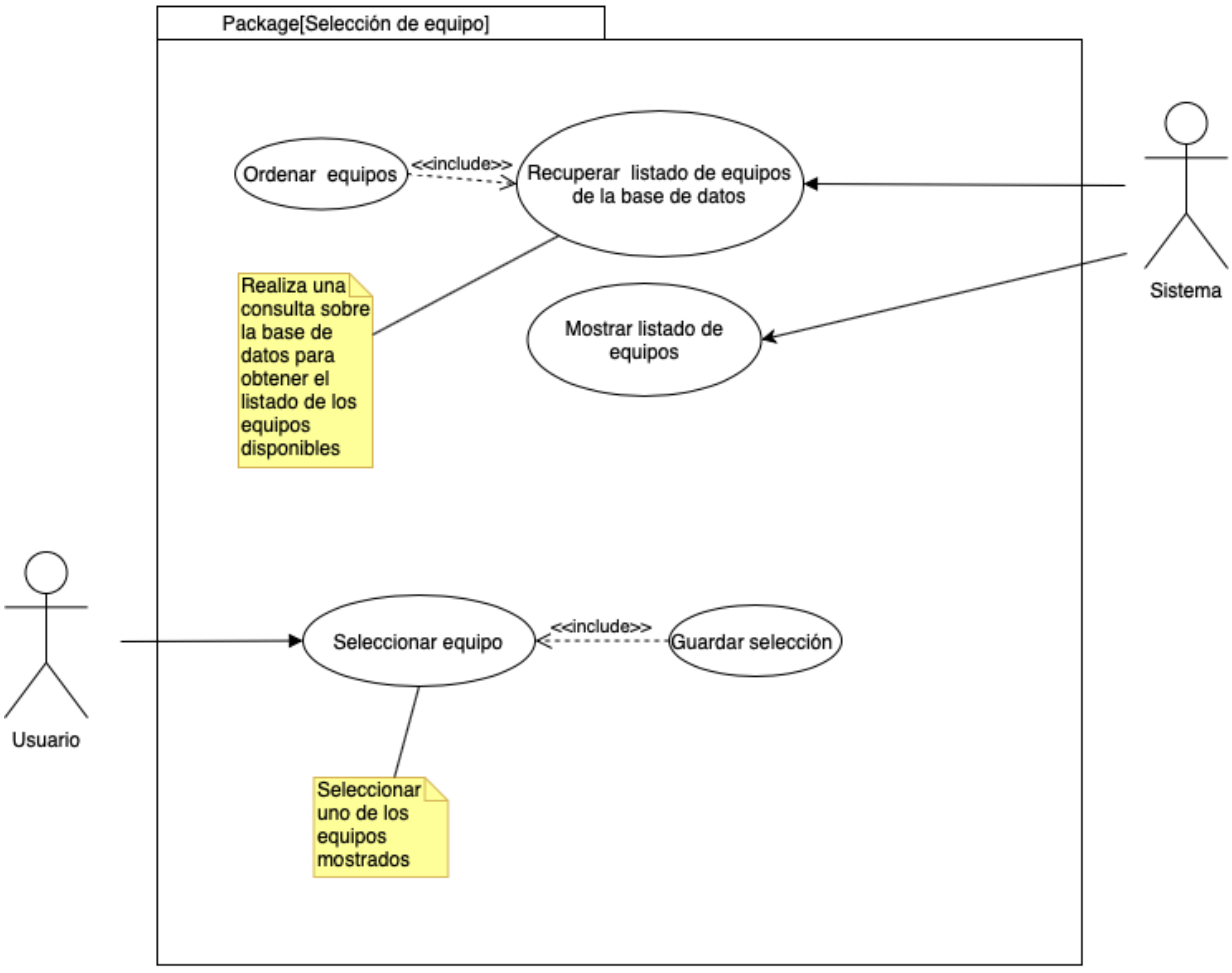


Figura 4.1: Diagrama de caso de uso, selección de equipo.

4.2.2. CU02 - Mostrar estadísticas de un equipo

Para modelar la manera en la que muestran las estadísticas y características de un equipo se especifica este caso de uso. Primero se muestran y ordenan todos los casos de uso, igual que el anterior caso de uso (4.2.1). El usuario selecciona un equipo y esta acción activa una consulta en la base de datos para poder recuperar los datos almacenados del equipo, entre los que se encuentran las estadísticas.

También se realiza una consulta para recuperar las estadísticas de los jugadores del equipo seleccionado que se mostraran junto a las estadísticas del propio equipo.

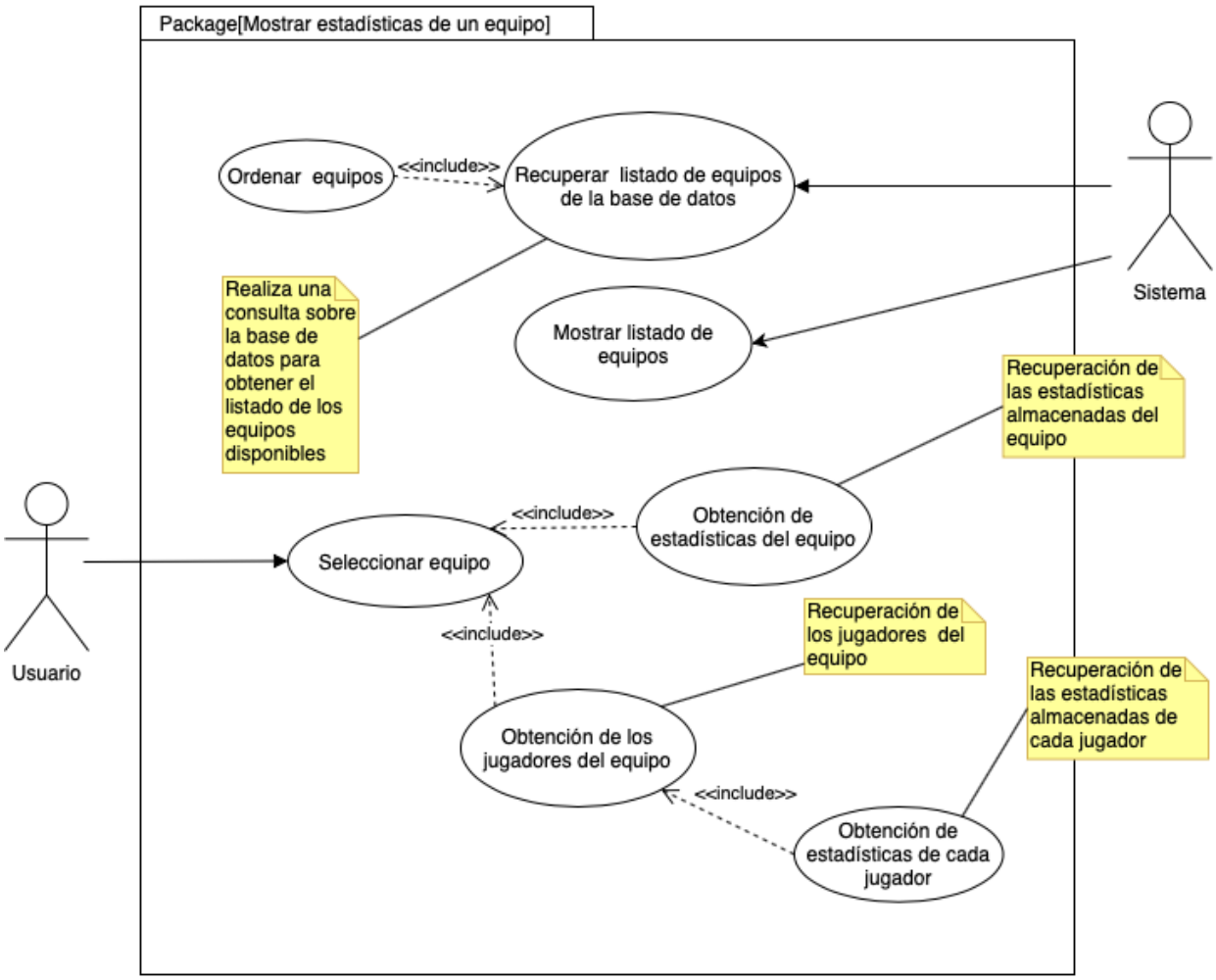


Figura 4.2: Diagrama de caso de uso, mostrar estadísticas de un equipo.

4.2.3. CU03 - Mostrar estadísticas de un jugador

El siguiente diagrama de caso de uso modela las acciones necesarias para visualizar las estadísticas de un jugador. El sistema recupera un listado con los jugadores de un equipo que se mostrarán al usuario. Una vez el usuario seleccione un jugador se mostrará toda la información del jugador seleccionado que se almacena en la base de datos. Dentro de esta información se muestra la posición obtenida para dicho jugador en el modelo de agrupamiento.

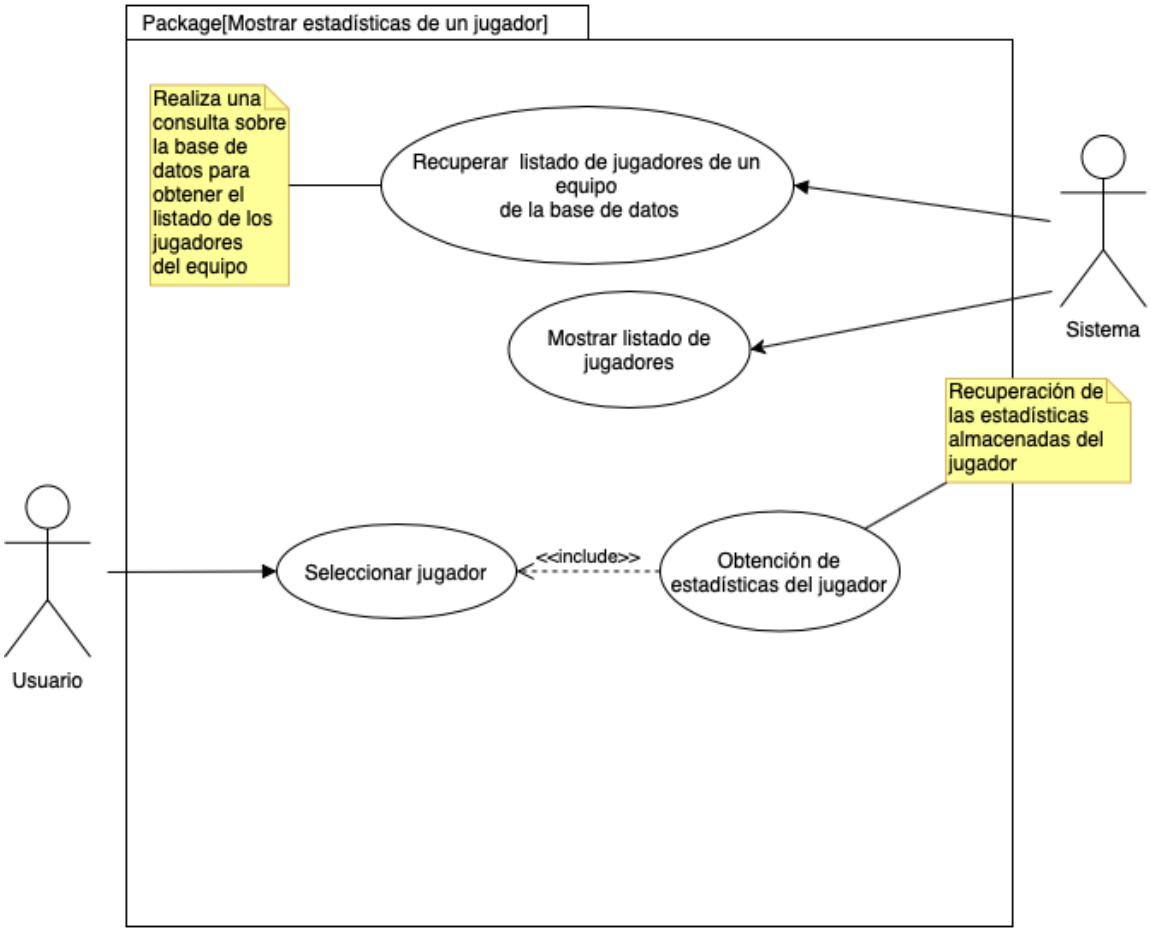


Figura 4.3: Diagrama de caso de uso, mostrar estadísticas de un jugador.

4.2.4. CU04 - Informe de un partido

Para moldear la realización de un informe de un partido se realiza este diagrama de caso de uso. El sistema muestra una pantalla dónde se muestra la información y estadísticas de un equipo (ver 4.2.2), dentro de esta pantalla aparecerá una pestaña dónde se realizará una comparación de las estadísticas avanzadas del equipo seleccionado y el equipo al que pertenece el entrenador, previamente seleccionado acorde al caso de uso *CU01* (ver 4.2.1). Cuando esta pestaña es activada el sistema envía la información necesaria al modelo de aprendizaje automático encargado de realizar la predicción. Una vez se predice el resultado se muestra junto con las estadísticas avanzadas de ambos equipos facilitando la comparación.

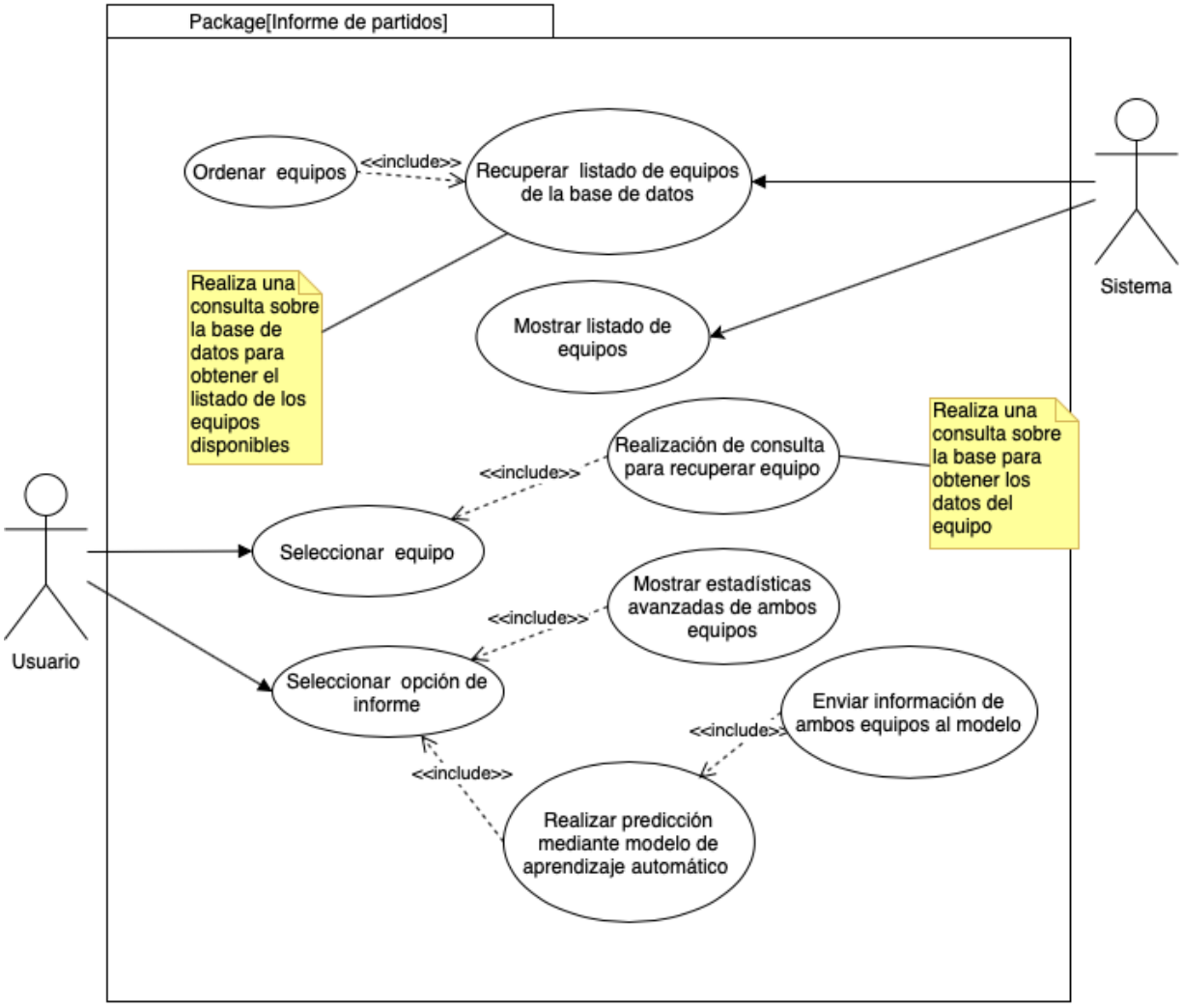


Figura 4.4: Diagrama de caso de uso, predicción de partidos.

4.2.5. CU05 - Análisis de los datos de cada partido

Para que el sistema tenga los datos necesarios para calcular las estadísticas es necesario analizar los ficheros disponibles de cada uno de los partidos. Para realizar este análisis primero se deben cargar los ficheros en el sistema. Una vez tenemos todos los ficheros cargados, para cada fichero se realiza en primer lugar un preprocesado de los datos para eliminar posible información irrelevante. Posteriormente se procede a formatear los datos de entrada para dejarlos en un formato que sea más manejable para el cálculo de las estadísticas simples y avanzadas de los equipos y jugadores.

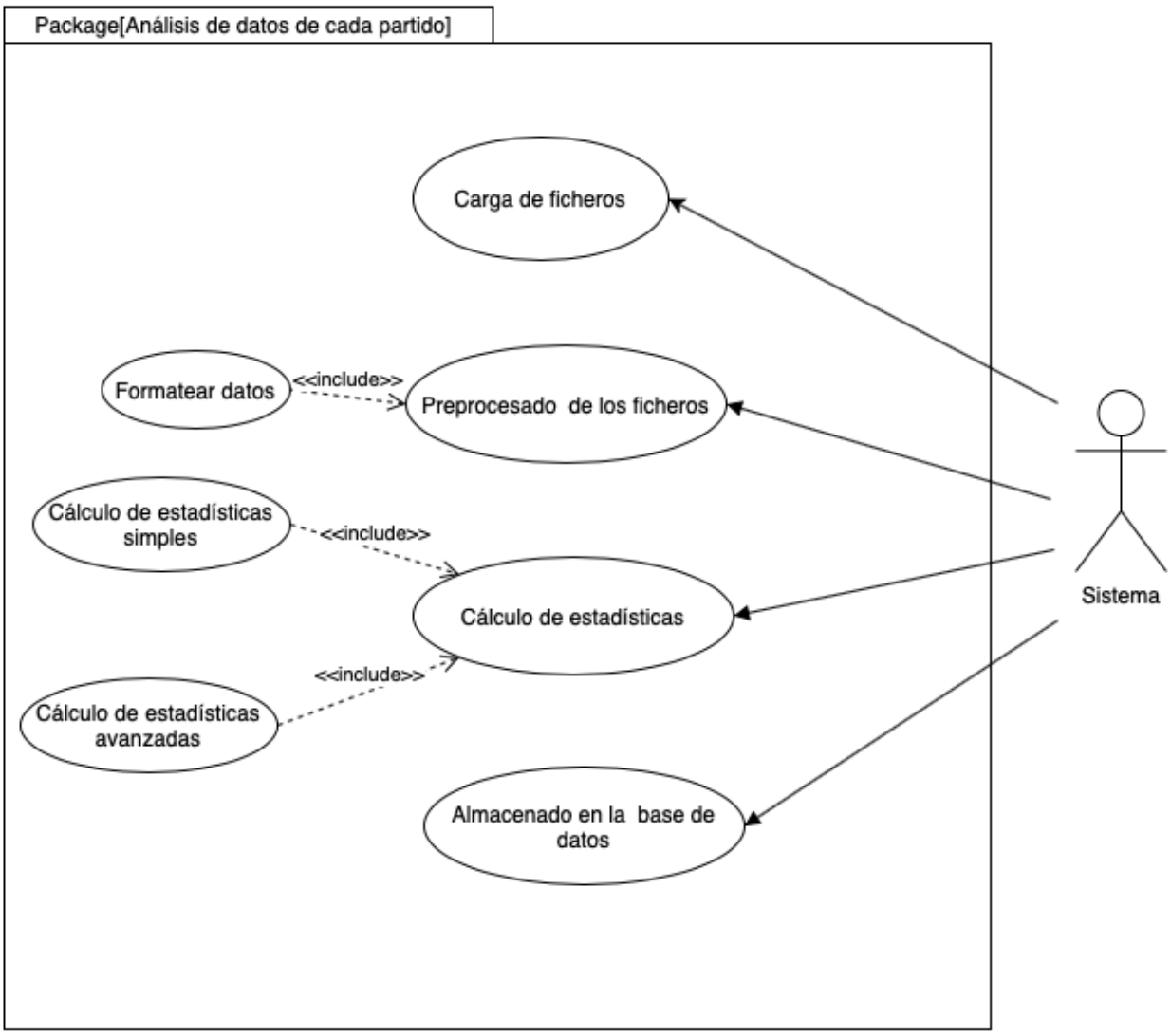


Figura 4.5: Diagrama de caso de uso, análisis de los datos de cada partido.

CAPÍTULO 5

Diseño del sistema

A lo largo de este sección se define el diseño y la estructura del sistema en función del análisis de especificaciones realizado en el capítulo anterior (4). Se entiende que el diseño del sistema el proceso de definición de la arquitectura, módulos, interfaces y datos del sistema para satisfacer los requisitos previamente identificados [18].

Este diseño pretende resolver los problemas planteados y construir una solución. Este diseño incluye la organización del sistema en subsistemas, que son los módulos que forman el sistema final. También se incluye una definición de la arquitectura que se utilizará. Esta arquitectura es la organización global del sistema. Existen distintos estilos de arquitectura, siendo cada uno más adecuado para cierto tipo de aplicaciones, es por eso que se debe elegir la arquitectura más eficaz para este proyecto.

La arquitectura permite dotar de contexto en el cual se tomarán decisiones de más bajo nivel en fases posteriores del diseño. Para llevar a cabo la definición del sistema se deben tomar decisiones de alto nivel que se aplicarán a todo el sistema, y el desglose del problema en subsistemas se realizará se tal manera que se asegure el cumplimento de los requisitos no funcionales (ver 4.1.2), y teniendo en cuenta que se debe asegurar el cumplimiento de los requisitos funcionales (ver 4.1.1).

5.1. Arquitectura del sistema

La arquitectura es un conjunto de patrones que dotan al sistema de un marco de referencia que permite guiar la construcción del software que forma parte del sistema, con la definición adecuada de la arquitectura se busca facilitar el cubrimiento de todos los requisitos del sistema. En esta sección se establecerá la estructura, el funcionamiento y la interacción entre cada parte del software.

Teniendo en cuenta las especificaciones que debe cumplir el sistema se utilizará una arquitectura basada en microservicios. Este tipo de arquitectura se estructura en base a pequeños servicios autónomos, donde cada servicio debe ser independiente al resto y cada uno debe implementar una funcionalidad distinta. En este modelo arquitectónico no existe una capa de datos independiente que controle la persistencia de los datos, es cada servicio el encargado de conservar y asegurar la persistencia de sus propios datos. Los microservicios ofrecen agilidad dado que cada microservicio se desarrolla de manera independiente

y favorece la corrección de errores y el control de versiones, ya que se puede actualizar o modificar un servicio sin que afecte al resto.

Otra de la ventajas que ofrece este tipo de arquitecturas es el aislamiento de errores, que significa que si alguno de los microservicios no está disponible la actividad de la aplicación no se ve interrumpida, siempre y cuando no sea necesaria la interacción entre los microservicios disponibles y los no disponibles.

La división del sistema en microservicios aumenta la escalabilidad, ya que al poder escalar de forma independiente cada uno de los distintos microservicios permite escalar de forma horizontal los subsistemas que más recursos requieran, sin tener que escalar el sistema completo. Como es lógico también ofrece aislamiento de los datos ya que entre microservicios sólo se comparte la información necesaria para el correcto funcionamiento del sistema global.

Es necesario definir una puerta de enlace para que sea el punto de entrada de los clientes, para esto en vez de llamar directamente a los microservicios se define una API [1]. De esta manera los clientes se comunicarán con la API y esta reenvía la petición al microservicio adecuado.

La API será una API REST [2]. REST no es más que una arquitectura para la creación de una API. Está pensada para la comunicación entre sistemas mediante la utilización directa del protocolo HTTP. Se utiliza para obtener datos o indicar la ejecución de operaciones sobre los datos en cualquier formato, sin necesidad de la abstracción adicional que ocurre en otros protocolos basados en patrones de intercambio de mensajes [19].

Al utilizar la arquitectura API REST para este sistema se pretende diseñar un protocolo cliente/servidor **sin estado** donde cada mensaje HTTP contiene toda la información necesaria para comprender la petición realizada por parte del cliente, permitiendo así que no haya que recordar ni almacenar ningún estado de las comunicaciones previo ni en el cliente ni el servidor. Para llevar a cabo las operaciones que se aplicarán sobre los datos almacenados se utilizan las operaciones definidas en HTTP, de las cuáles las más importantes son:

- **POST**: operación para solicitar la creación de nuevo recurso.
- **GET**: operación para solicitar la lectura de un recurso.
- **PUT**: operación para solicitar la actualización de un recurso.
- **DELETE**: operación para solicitar la eliminación de un recurso.

Mediante estas operaciones se realizan la peticiones para la modificación y lectura de los recursos del sistema y para la identificación de cada recurso se utiliza un sistema de identificación basado en URI [3], que hace que cada recurso sea direccionable únicamente a través de sus URI.

Para la representación de los recursos y el intercambio de datos entre el cliente y el servidor

[1] Término que proviene de Application Programming Interface, que indica a las aplicaciones cómo pueden mantener una comunicación entre sí.

[2] Término que proviene del inglés REpresentational State Transfer, es un estilo de arquitectura software para sistemas hipermedia distribuidos.

[3] Identificador de recursos uniforme, cadena de caracteres que identifica los recursos de una red de forma unívoca.

se utilizará el formato de texto JSON, ya que es un formato de texto sencillo pensado para el intercambio de texto.

{
'player_id': '1234',
'name': 'Stephen',
'team_id': '4321'
}

Figura 5.1: Ejemplo de representación JSON de un jugador

Hoy en día el formtato JSON está muy extendido ya que resulta bastante sencillo realizar un analizador sintáctico que transforme un objeto a JSON. Esto quiere decir que es muy sencillo encontrar información sobre JSON y librerías para cualquier lenguaje de programación que permitan trabajar con objetos en formato JSON.

Para la parte del cliente se desarrollará una aplicación para tablets con sistemas Android. Esta aplicación realizará peticiones a la API del servidor, que será la encargada de transmitir la petición al microservicio adecuado de manera que se devuelva a la aplicación la información adecuada.

5.2. Identificación de subsistemas

Como se comenta en la sección anterior (ver 5.1) el sistema se dividirá en subsistemas a los que, dado a la arquitectura que vamos a utilizar, se le llamarán microservicios. En esta sección se realiza una especificación de cada uno de los microservicios que conforman el sistema. Para cada microservicio se ofrecerá una descripción las funcionalidades que implementan y si se comunica o no con otros microservicios.

Los microservicios en los que se divide el sistema son:

- **Inteligencia artificial**.
- **Análisis de partidos**.
- **Cálculo de estadísticas**.
- **Gestor de base de datos**.

La representación de los microservicios dentro de la arquitectura del sistema quedaría de la siguiente forma:

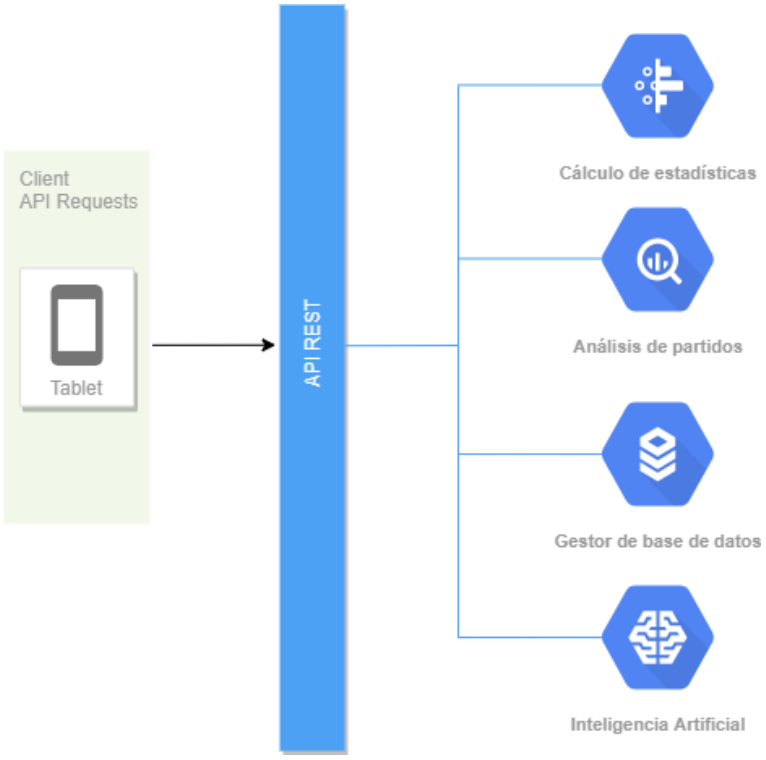


Figura 5.2: Arquitectura del sistema.

5.2.1. Análisis de partidos.

Este microservicio se encarga de analizar los ficheros con los datos de los partidos, estos ficheros son ofrecidos por la NBA y se hablará en detalle de ellos en el capítulo 6. Para realizar el análisis se procesan los ficheros extrayendo información sobre el partido. La información que se busca es la que es necesaria para el cálculo de las estadísticas. Una vez se obtiene esta información esta será enviada al microservicio encargado de calcular las estadísticas.

5.2.2. Cálculo de estadísticas.

Mediante la información procesada de los partidos este subsistema calcula las estadísticas para los jugadores y equipos implicados en el partido. Primero calcula las estadísticas simples y a partir de estas calcula las estadísticas avanzadas. Una vez las estadísticas simples son calculadas se guardan en la base de datos, por lo que debe conectar con el microservicio encargado de gestionar la base de datos. Sin embargo, las estadísticas avanzadas se calculan en el momento en el que el cliente solicite dicha información. Esto se debe a que son valores que pueden cambiar a menudo según se vayan añadiendo datos a la base de datos. En este microservicio también se incluye el modelo de agrupamiento que permite obtener el subtipo de posición de cada jugador.

5.2.3. Inteligencia artificial.

Este microservicio será el encargado de almacenar y gestionar los modelos de inteligencia artificial que se utilizan. Se almacena el modelo que realiza la predicción de partidos y los modelos de clusterización de cada posición. Cuando se realiza el informe para dos equipos

la información de ambos debe ser enviada a este microservicio que se encarga de realizar la predicción utilizando el modelo predictivo. Los modelos de clusterización se utilizan cuando se quiere añadir la información del clúster a un jugador existente en la base de datos.

5.2.4. Gestor de base de datos.

En este microservicio se implementan las operaciones necesarias para almacenar datos en la base de datos y realizar las consultas oportunas. Es el encargado de proporcionar la persistencia de datos a los microservicios que lo necesiten. Además, cuando el cliente haga peticiones para obtener datos de los equipos o de los jugadores será el encargado de realizar la consulta adecuada a la base de datos. Una vez hace la consulta debe devolver la información al cliente en un formato entendible por este.

5.3. Revisión de casos de usos

En esta sección se relacionan los casos de uso definidos en la sección 4.2 con los distintos subsistemas identificados en la sección anterior 5.2.

- **CU01 - Selección de equipo**.
  En este caso de uso solo interviene el microservicio gestor de base de datos. Este microservicio realiza una consulta a la base de datos para obtener el listado de los equipos disponibles y mandar la información al cliente. El cliente mostrará esta información al usuario que seleccionará el equipo.
- **CU02 - Mostrar estadísticas de un equipo**.
  Para este se utiliza el microservicio de gestión de la base de datos para obtener las estadísticas simples. El cliente realiza una petición a la API para recuperar las estadísticas de un equipo y la API reenvía la petición al microservicio gestor de la base de datos. Este microservicio realizará la consulta sobre el equipo a la base de datos y devolverá la información al cliente. Para mostrar las estadísticas avanzadas del equipo se utiliza el microservicio de cálculo de estadísticas, que recibe la información del equipo y a partir de esos datos las calcula.
- **CU03 - Mostrar estadísticas de un jugador**.
  Este caso de uso funciona de la misma manera que el anterior, con la única diferencia de que el microservicio gestor de la base de datos realiza la consulta de un jugador.
- **CU04 - Informe de un partido**.
  Para este caso de uso la petición del cliente será reenviada al microservicio de generación de informes. Este microservicio pedirá información sobre los equipos al microservicio gestor de la base de datos. Una vez obtenga la información la procesará y será devuelta al cliente en el formato adecuado.
- **CU05 - Análisis de los datos de cada partido**.
  En este caso de uso interviene tanto el microservicio de análisis de partidos, el microservicio gestor de la base de datos y el microservicio de cálculo de estadísticas. Primero el microservicio de análisis de partidos se encarga de procesar los datos del partido. Una vez ha sido procesado esa información se envía al microservicio de

cálculo de estadísticas que la utiliza para extraer las estadísticas de los equipos y jugadores. Una vez calculadas se guardan en la base de datos mediante el microservicio gestor.

5.4. Estructura de la base de datos

Para la base de datos se utilizará un modelo relacional que permite guardar puntos de datos que se relacionan ente si [20] . El modelo relacional permite separar las tablas de datos, las vistas y los índices, estructuras de datos lógicas, del almacenamiento físico. Esta separación permite la administración del almacenamiento físico de datos sin que el acceso a los datos como una estructura lógica se vea afectado.

La elección del modelo relacional se debe en parte a las ventajas que este ofrece. Aplicando las reglas de normalización de forma consecuente el modelo relacional facilita que el almacenamiento de datos sea libre de redundancias. La normalización también permite obtener una alta consistencia de datos ya que permite almacenar datos sin contradicciones. Los motores de bases de datos relacionales ofrecen funcionalidades que controlan automáticamente las condiciones de integridad, bloqueando automáticamente las transacciones que pongan en peligro la integridad. Una transacción no es más que un conjunto de operaciones que se deben ejecutar una después de la otra. Las operaciones de la transacción se ejecutan de manera indivisible.

El diagrama de entidad relación de la base de datos es el siguiente:

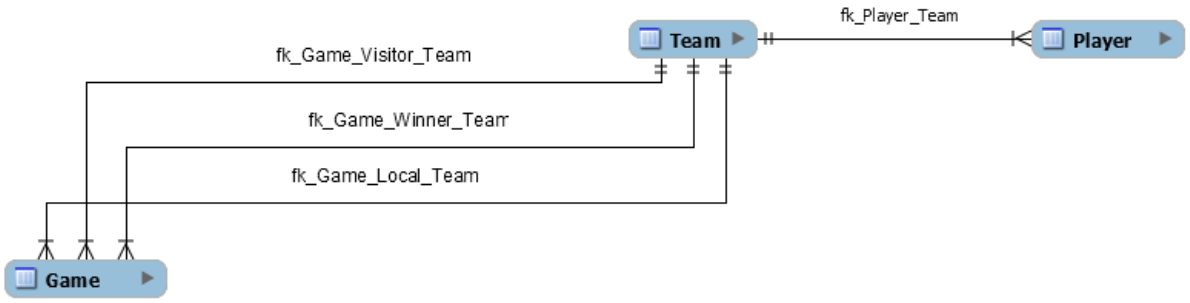


Figura 5.3: Diagrama entidad relación de la base de datos del sistema

CAPÍTULO 6

Análisis de los Datos

Para poder realizar un análisis correcto es necesario estudiar el formato de entrada de los datos. Esto es necesario para entender qué información es relevante y cómo encontrarla dentro del fichero. En este capítulo se realiza el estudio de los ficheros de entrada del sistema. También se explica el proceso para analizarlos y calcular las distintas estadísticas para equipos y jugadores. Para cada estadística avanzada se proporcionará una explicación de qué aporta al entrenador dicha estadística.

De la misma manera, se explica el proceso de análisis realizado para la construcción del modelo de aprendizaje automático para la predicción de partidos. El proceso para la clusterización de los jugadors también queda recogido en este capítulo.

6.1. Ficheros de entrada

La NBA proporciona dos tipos de ficheros con los datos de cada partido de la temporada. Cada fichero proporciona información distinta sobre el partido, uno ofrece información sobre los eventos ocurridos en el encuentro y el otro sobre las coordenadas de los jugadores en cada momento. A partir de estos datos se construyen las estadísticas que apoyan al entrenador en la toma de decisiones referentes a próximos encuentros.

El formato de los ficheros es distinto, lo que implica que el análisis que se realice debe ajustarse al formato en el que se encuentra la información. El fichero que incluye los eventos del partido es conocido como fichero **play-by-play** y está formato CSV, en el que se incluye la información separada por comas y es representable en forma de tabla. El fichero en el que se incluyen las coordenadas es conocido como **tracking data** y la información está en formato JSON, que es un formato adecuado para la transmisión vía web.

6.1.1. Fichero Play-By-Play

Como se comenta anteriormente, este tipo de ficheros almacena los eventos ocurridos durante un partido. Estos eventos son tiros, anotados o fallados, asistencias, rebotes, pérdidas, faltas, violaciones, tiempos muertos, saltos entre dos, expulsiones y comienzos y

finales de cuartos. Todos estos eventos son recogidos para el equipo local y para el equipo visitante.

GAME_ID	EVENTNUM	EVENTMSGTYPE	EVENTMSGACTIONTYPE	PERIOD	WCTIMESTRING	PCTIMESTRING	HOMEDESCRIPTION
21500001	0	12	0	1	8:12 PM	12:00	
21500001	1	10	0	1	8:12 PM	12:00	Jump Ball Horford vs. Drummond: Tip to Ilyasova
21500001	2	2	42	1	8:13 PM	11:41	Horford BLOCK (1 BLK)
21500001	3	4	0	1	8:13 PM	11:39	Bazemore REBOUND (Off:0 Def:1)
21500001	4	5	45	1	8:13 PM	11:37	Bazemore Out of Bounds - Bad Pass Turnover Turnover (P1.T1)

NEUTRALDESCRIPTION	VISITORDESCRIPTION	SCORE	SCOREMARGIN	PERSON1TYPE	PLAYER1_ID	PLAYER1_NAME
				0.0	0	
				4.0	201143	Al Horford
	MISS Drummond 2' Driving Layup			5.0	203083	Andre Drummond
				4.0	203145	Kent Bazemore
				4.0	203145	Kent Bazemore

PLAYER1_TEAM_ID	PLAYER1_TEAM_CITY	PLAYER1_TEAM_NICKNAME	PLAYER1_TEAM_ABBREVIATION	PERSON2TYPE	PLAYER2_ID	PLAYER2_NAME	PLAYER2_TEAM_ID	PLAYER2_TEAM_CITY
				0	0			
1610612737.0	Atlanta	Hawks	ATL	5	203083	Andre Drummond	1610612765.0	Detroit
1610612765.0	Detroit	Pistons	DET	0	0			
1610612737.0	Atlanta	Hawks	ATL	0	0			
1610612737.0	Atlanta	Hawks	ATL	0	0			

PLAYER2_TEAM_NICKNAME	PLAYER2_TEAM_ABBREVIATION	PERSON3TYPE	PLAYER3_ID	PLAYER3_NAME	PLAYER3_TEAM_ID	PLAYER3_TEAM_CITY	PLAYER3_TEAM_NICKNAME	PLAYER3_TEAM_ABBREVIATION
		0	0					
Pistons	DET	5	101141	Ersan Ilyasova	1610612765.0	Detroit	Pistons	DET
		4	201143	Al Horford	1610612737.0	Atlanta	Hawks	ATL
		0	0					
		0	0					

Figura 6.1: Ejemplo de fichero play-by-play

Estos eventos se recogen en el fichero con formato CSV. Este formato representa los datos como una tabla, donde cada fila está separada del resto por un salto de línea y cada columna se separa por comas. El fichero tiene un total de treinta tres columnas, aunque no todas nos van a resultar de utilidad. La primera columna es **GAME_ID** y es el identificador del partido, por lo que contiene el mismo valor para todas las filas del mismo fichero. La segunda columna es **EVENTNUM** que es el número de evento, es decir, en esta columna se lleva un contador de los eventos que se incluyen en el fichero.

La tercera columna es la más importante y se trata de **EVENTMSGTYPE**. Esta columna representa el tipo de evento del que se trata, es decir, es el identificador del tipo de evento. El valor es un número del 1 al 13, o 18 si ha habido algún error al capturar el evento. El significado de cada valor es el siguiente:

1. Este valor indica que el evento es un tiro de campo anotado. No hay valor para diferenciar entre tiro normal y triple por lo que esta información deberá ser extraída de otras columnas.
2. Indica un tiro de campo fallado. Al igual que ocurre con el valor anterior no hace distinción entre tiros de tres o tiros de dos.
3. Los eventos con este EVENTMSGTYPE son tiros libres realizados. Para saber si ha sido anotado o fallado hay que utilizar otras columnas.
4. Indica que el evento se trata de un rebote. Es el mismo valor para rebotes defensivos y rebotes ofensivos.
5. Este valor indica una pérdida de la posición.
6. Cuando la columna tiene este valor significa que el evento se trata de una falta. En este evento se incluyen tanto faltas personales, faltas en ataque y demás tipos de

faltas.

7. Este valor significa que el evento es una violación. Esta violación puede ser que se ha acabado el tiempo de la posesión sin que el equipo atacante haya tirado. También puede ser una violación de *goaltending*, que es cuando un defensor realiza un tapón mientras la pelota está descendiendo.
8. Este valor indica que ha sucedido una sustitución. En otro campo se indican el jugador sustituido y el jugador sustituto.
9. Indica un tiempo muerto, tanto completo como corto.
10. Este valor indica un salto entre dos. Estos saltos entre dos incluyen el salto inicial y los que suceden a causa de una pelota disputada por dos jugadores.
11. Significa que ha sucedido una expulsión.
12. Este valor se utiliza para indicar el comienzo de un cuarto.
13. Indica el final de un cuarto.

En cuarta posición se encuentra una columna con el nombre **EVENTMSACTIONTYPE** que aporta información sobre el evento ocurrido. Sigue el mismo modelo que la columna EVENTMSGTYPE, los valores que puede recibir son numéricos, con un rango mayor. La información que aporta es irrelevante para este proyecto, debido a que la información que aporta es relativa a cómo se ha realizado el evento. No aporta información que sea necesaria para calcular estadísticas.

La siguiente columna tiene el nombre de **PERIOD** y especifica el cuarto en el que sucede el evento. El valor que puede recibir es un número del 1 al 4. Después se encuentra la columna **WCTIMESTRING** que indica la hora a la que sucede el evento. Otra columna que se encuentra a continuación es **PCTIMESTRING**, que es el tiempo restante del cuarto.

A continuación se encuentra una columna importante que es **HOMEDESCRIPTION**. En esta columna se encuentra una descripción del evento ocurrido si el jugador que protagoniza el evento es del equipo local. También se encuentra una columna similar pero para el equipo visitante que es **VISITORDESCRIPTION**. De ambas columnas se puede extraer información muy importante para el sistema. Por ejemplo, cuando el evento es un tiro de campo anotado en este campo encontraríamos algo similar a "Player1 12' Step Back Jump Shot (2 PTS) (Player2 1 AST)". Aquí observamos que dentro de este campo se ofrece una pequeña descripción del tipo de tiro que ha realizado, permitiendo diferenciar entre tiros de dos y tiros de tres, que incluirían antes la cadena de texto "3 PTS". También incluye si la canasta es precedida o no de una asistencia. El texto que se incluye en esta columna tiene un formato concreto para cada tipo de evento, facilitando así el análisis.

En el fichero se incluyen seis columnas para identificar a un jugador. Además, se identifican un máximo de tres jugadores que pueden intervenir en un evento. Las columnas son: **PLAYER_ID**, **PLAYER_NAME**, **PLAYER_TEAM_ID**, **PLAYER_TEAM_CITY**, **PLAYER_TEAM_NICKNAME** y **PLAYER_TEAM_ABBREVATION.** En las columnas que hagan referencia a PLAYER1 tendremos la información del jugador que realiza el evento. En PLAYER2 se encuentra información sobre el otro participante en el evento, si

el evento es de un tiro de campo anotado indica el jugador que ha realizado la asistencia, si se trata de un tiro de campo fallado se trata del jugador que ha realizado el tapón.

Otras dos columnas importantes son **SCORE** y **SCOREMARGIN**, donde la primera contiene el marcador del partido cuando sucede el evento y la segunda contiene el margen de puntos. Hay más columnas pero son totalmente irrelevantes para el análisis del partido.

6.1.2. Fichero Tracking Data

En este fichero se almacenan las coordenadas de cada jugador en un momento determinado. El formato de este fichero es JSON, la estructura de este formato se basa en clave valor, como los diccionarios en los lenguajes de programación. Este formato es sencillo y facilita la labor de definir un analizador sintáctico para el fichero.

El fichero consta de tres valores principales. La primera clave es **gameid** que contiene un número que es el identificador del partido. Otra clave es **gamedate** que tiene como valor la fecha en la que se realiza el partido. Por último nos encontramos la clave **events** cuyo valor es una lista de diccionarios. En esta lista se encuentran todos los eventos del partido recogidos en el fichero play-by-play, cada diccionario de la lista tiene una clave **eventid** que tiene el identificador del evento. Además están las claves **home** y **visitor** cuyo valor es un diccionario que incluye información sobre el equipo local y el equipo visitante respectivamente.

Estos diccionarios tienen una clave **teamid** cuyo valor es el número identificativo del equipo. La clave **abbrevation** también se puede encontrar en estos diccionarios y tiene como valor la abreviación del equipo. Contiene también una clave **name** con el nombre del equipo. Y por último se puede encontrar la clave **players** con un valor que es una lista de diccionarios, donde en cada diccionario se incluye información de un jugador.

Cada representación de un jugador tiene como clave **firstname** cuyo valor es una cadena de texto con el nombre del jugador, también se encuentra la clave **lastname** con el valor del primer apellido del jugador como cadena de texto. También está la clave **jersey** con el valor numérico del dorsal del jugador. También se incluye el número identificador del jugador con la clave **playerid**. Por último se encuentra la clave **position** donde el valor hace referencia a las posiciones que puede ocupar el jugador. Las posiciones son reducidas a tres tipos:

- Un valor de "G" indica que el jugador juega en la posición de base o escolta.
- El valor "F" significa que el jugador ocupa las zonas de alero y ala-pívot.
- Cuando tiene el valor "C" quiere decir que el jugador juega como pívot.
- También puede incluir una combinación de los valores anteriores.

Dentro de los diccionarios de la lista events la última clave que se encuentra es **moments**, que es una lista de listas, a cada lista se le llamará momento. Esta lista representan todos los momentos sucedidos en el evento recogido, es decir, todos los movimientos sucedidos durante el evento. Cada elemento de la lista tiene seis valores con significado especial cada uno.

1. El primer valor es número del 1 al 4 que indica el cuarto en el que sucede el momento.

```
{'abbreviation': 'LAC',
 'name': 'Los Angeles Clippers',
 'players': [{'firstname': 'Glen',
   'jersey': '0',
   'lastname': 'Davis',
   'playerid': 201175,
   'position': 'F-C'},
  {'firstname': 'Chris',
   'jersey': '3',
   'lastname': 'Paul',
   'playerid': 101108,
   'position': 'G'},
  {'firstname': 'JJ',
   'jersey': '4',
   'lastname': 'Redick',
   'playerid': 200755,
   'position': 'G'},
```


Figura 6.2: Ejemplo de equipo visitante o local extraído de un fichero de tracking data

2. En el segundo valor hay un número que indica la hora en la franja horaria del pacífico central en milisegundos y formato unix-time.

3. El tercer valor que se encuentra en el momento es el tiempo restante del cuarto en segundos. Teniendo en cuenta que un cuarto tiene una duración de doce minutos el valor máximo que puede aparecer aquí es de 720.

4. El valor que se encuentra en la cuarta posición indica el tiempo que queda en el reloj de posesión. El valor máximo que puede tener es de 24, que son los segundos de posición permitidos en el reglamento NBA.

5. En la posición número cinco se encuentra casi siempre el valor None, que pude hacer referencia a un valor nulo. No se ha encontrado ningún tipo de información sobre qué indica este valor.

6. En última posición se encuentra una lista con once ítems. Cada ítem es una lista y contiene cinco valores, donde se encuentran las coordenadas de cada jugador y de la pelota.

    a) La primera de las listas contiene la información de las coordenadas de la pelota. Los dos primeros valores son -1, ya que hace referencia al identificador del equipo y del jugador. Los valores que se encuentran en la posición tercera y cuarta son la posición de la pelota en coordenadas x y. El último valor es el radio de la pelota, que cambia dependiendo de la elevación de la pelota. Alcanza su valor máximo en el vértice del arco del tiro, y comienza a disminuir conforme se acerca a la canasta.

b) Los siguientes diez ítems contienen la información de los jugadores de ambos equipos. Tienen el mismo formato que los valores de la pelota. El primer valor de la lista es el número identificativo del equipo y el segundo el identificador del jugador. A continuación nos encontramos las coordenadas x e y de la posición del jugador y por último el valor del radio de la pelota, que para los jugadores tomará un valor de 0.0.

```
[3,
 1431486313010,
 715.32,
 19.0,
 None,
 [[-1, -1, 43.51745, 10.76997, 1.11823],
  [1610612745, 1891, 43.21625, 12.9461, 0.0],
  [1610612745, 2772, 90.84496, 7.79534, 0.0],
  [1610612745, 2730, 77.19964, 34.36718, 0.0],
  [1610612745, 2746, 46.24382, 21.14748, 0.0],
  [1610612745, 201935, 81.0992, 48.10742, 0.0],
  [1610612746, 2440, 88.12605, 11.23036, 0.0],
  [1610612746, 200755, 84.41011, 43.47075, 0.0],
  [1610612746, 101108, 46.18569, 16.49072, 0.0],
  [1610612746, 201599, 78.64683, 31.87798, 0.0],
  [1610612746, 201933, 65.89714, 25.57281, 0.0]]]
```

Figura 6.3: Ejemplo de momento extraído de un fichero de tracking data

6.2. Extracción de Información

Una vez que se han identificado los ficheros de entrada y se han analizado se procede a definir cómo se va a extraer la información de estos ficheros. Para ello, a lo largo de esta sección se define de qué parte se extrae la información necesaria para alimentar la base de datos. No se pretende dar una solución mediante el diseño de una función programada, para ello ver el capítulo 8, si no, explicar en qué parte de cada fichero se encuentra la información relevante.

Como se explica en la sección anterior, para cada partido tenemos dos ficheros con datos distintos. Para cada partido es necesario extraer las estadísticas base de cada jugador, que son almacenadas en la base de datos. A partir de estas estadísticas base se calculan las del equipo. Las estadísticas básicas son:

- Puntos anotados.
- Tiros de campo fallados.
- Tiros de campo anotados.
- Tiros de tres anotados.
- Tiros de tres fallados.
- Asistencias realizadas.
- Rebotes defensivos.

- Rebotes ofensivos.
- Robos.
- Tapones.
- Tiros libres anotados.
- Tiros libres fallados.
- Pérdidas.
- Faltas personales.

Todos estos datos son obtenidos del fichero play-by-play, mediante la búsqueda de los distintos valores de la columna EVENTMSGTYPE. Hay valores que dan directamente una de las estadísticas básicas, pero para obtener otras hay que hay que analizar valores de otras columnas.

De estos ficheros también se extrae la información necesaria para calcular las estadísticas avanzadas, que son descritas en el análisis de requisitos (ver 4.1.1). Para este tipo de estadísticas se ofrecerá una descripción y una explicación de qué aportan a nivel de análisis.

6.2.1. Estadísticas Simples

Para obtener todas estas estadísticas primero se van realizando filtros sobre el fichero para obtener solo un tipo específico de evento. Una vez que se filtra la información que se va extrayendo se almacena en un objeto intermedio que identifica a un jugador, para obtener las de un equipo se hace un sumatorio las de los jugadores del propio equipo.

Para calcular los puntos anotados, los tiros de campo de anotados y las asistencias se utiliza el valor 1 de la columna EVENTMSGTYPE. Este tipo de eventos son tiros de campo anotados, y en la columna PLAYER1_ID se encuentra el identificador del jugador que ha anotado el tiro de campo. De la columna PLAYER2_ID se puede extraer el identificador del jugador que ha propiciado la canasta mediante una asistencia, si la canasta no proviene de asistencia esta columna está vacía. En el fichero se lleva un contador de los puntos que lleva anotado cada jugador, este contador se puede observar en las columnas HOMEDESCRIPTION o VISITORDESCRIPTION, dependiendo del equipo al que pertenezca el jugador que ha anotado la canasta. En esta columna de encuentran valores de texto similiares a “Marc Morris 13’ Step Back Jump Shot (2 PTS) (Drummond 1 AST) ”. Como se puede observar justo después de donde se especifica los puntos que lleva anotados el jugador que acaba de encestar, contando los que acaba de anotar. De este tipo de evento también se extrae si el tiro ha sido triple o no. Para ello se deben analizar también las casillas HOMEDESCRIPTION y VISITORDESCRIPTION, que al tratarse de un tiro triple tendrán un valor con la estructura de la siguiente cadena de texto: “Schroder 3PT Jump Shot (14 PTS) (Millsap 3 AST)”. Dentro de este valor se encuentra la subcadena “3PT” que indica que el tiro anotado ha sido de tres puntos. Estos datos se van recopilando realizando una iteración de manera que se analicen todos los eventos de este tipo de uno en uno. Cuando se esté analizando un evento de este tipo la información se almacenada en objetos que representan a los jugadores y en los que se van a añadiendo las asistencias, los puntos anotados, los tiros de campo encestados y los triples anotados.

Para un valor de 2 en la columna de EVENTMSGTYPE tenemos los eventos que representan un tiro de campo fallado. De estos eventos se puede extraer información sobre los tiros de campo fallados, los triples fallados y los tapones. De la misma forma que se extraen los tiros de campo anotados se extraen los fallados. En las columnas que aportan las descripciones de los eventos se puede identificar la subcadena "3PTS" que indica que el tiro fallado ha sido de tres. En la columna PLAYER1_ID se encuentra el identificador del jugador que ha fallado el tiro.

Para conocer si el tiro fallado se debe a un tapón, en la descripción del evento se detallará, junto a un contador de los tapones que lleve el jugador. Además, en la columna PLAYER3_ID se puede encontrar el identificador del jugador que realiza el tapón.

Para extraer la información referente a los tiros libres se utiliza el valor 3 de la columna EVENTMSGTYPE. Estos eventos son todos tiros libres, y para distinguir los tiros libres que se fallan se busca en la descripción del evento la cadena de texto "MISS", que indica que el tiro libre ha sido fallado. Para identificar el jugador que ha anotado o fallado el tiro libre se utiliza el valor de la columna PLAYER1_ID.

Para contabilizar los rebotes se utiliza el valor 4 de la columna EVENTMSGTYPE. Estos eventos contabilizan los rebotes, pero no establecen diferencia entre los defensivos y los ofensivos. Para averiguar de qué tipo de rebote se trata se debe analizar las descripciones del evento. Dentro de la descripción aparece un contador con los rebotes ofensivos y defensivos del jugador, tiene la siguiente forma: "(Off: 0 Def: 1)". De este contador se puede extraer el número de rebotes ofensivos y defensivos que lleva un jugador.

Las pérdidas y los robos se extraen del valor 5 de EVENTMSGTYPE en el que se encuentran los eventos de pérdidas. Para conocer qué jugador ha perdido la posesión se debe extraer el valor de PLAYER1_ID, que contiene el identificador del jugador. Si la pérdida ha sido propiciada por un robo en PLAYER2_ID se encontrará el identificador del jugador que ha realizado el robo.

En el baloncesto existe una penalización conocida como bonus, y que se le otorga a un equipo cuando ha recibido cinco faltas. Cuando se activa este bonus cualquier falta recibida será penalizada con tiros libres a favor. Las faltas que cuentan para este bonus son las personales, excepto las que se produzcan en ataque. Este tipo de faltas son recogidas con el valor 6 de la columna EVENTMSGTYPE.

En este tipo de eventos se ofrece en la el tipo de falta se ha cometido mediante las siguientes siglas:

- "S. FOUL": hace referencia a una falta de tiro, que se produce al realizar una falta a un jugador que está realizando la acción de tirar
- "P. FOUL": significa que se ha cometido una falta personal, que engloba todos los tipos de falta que cuentan para el bonus. Cuando aparezca este valor significa que ha habido dificultad para identificar el tipo exacto de falta cometida.
- "L. B. FOUL": indica que la falta cometida es de tipo pérdida de pelota, que se producen en disputas de balones divididos.

Con esto ya tendríamos todas las estadísticas simples de cada jugador que ha participado en el partido. Para calcular la de los equipos simplemente se suman las de sus jugadores. Una vez que tenemos tanto las estadísticas de los jugadores como las de los equipos

la información se añade a la base de datos y ya sería posible calcular las estadísticas avanzadas.

6.2.2. Estadísticas Avanzadas

Este tipo de estadísticas permiten apreciar a jugadores cuyos números no son espectaculares pero que desempeñan un papel esencial en el funcionamiento del equipo. Además, han puesto sobre la mesa la posibilidad de proyectos deportivos basados en el análisis de los datos, ejemplos de este tipo de proyectos son los Houston Rockets de Daryl Morey o los Philadelphia 76ers de Sam Hinkie.

Las estadísticas avanzadas ofrecen una forma diferente de analizar el baloncestos permitiendo conocer como de eficiente en ataque o defensa es un equipo o jugador. Este tipo de datos deben ser puestos en contexto para que tengan sentido, es por esto que normalmente se comparan con otras estadísticas avanzadas.

Rating Ofensivo y Floor porcentaje.

El raiting ofensivo es una métrica de la eficiencia ofensiva de los equipos y jugadores, que fueron descritas por Dean Oliver en 2004 en su libro "Basketball on Paper" . Según el propio Dean: "El rating ofensivo individual es el número de puntos producidos por un jugador cada cien posesiones individuales. En otras palabras, '¿Cuántos puntos puede generar un jugador cuando lo intenta?'." [21]. Para el floor porcentaje Dean da la siguiente definición: "El el porcentaje de posesiones de un equipo en las cuales el equipo anota al menos un punto. Básicamente, el floor porcentaje responde a la pregunta: '¿Qué porcentaje de los tiempos de un equipo en la pista anota?'.".

De estas definiciones se observa que ambas estadísticas están estrechamente relacionadas y son necesarias para entender la eficiencia ofensiva. Combinando ambas estadísticas obtenemos la información sobre la cantidad de puntos que es capaz de anotar un equipo o un jugador y cuantas posesiones ofensivas es capaz de transformar en al menos un punto.

El cálculo de estas estadísticas para los equipos es relativamente simple, ya que se basa en aplicar la definición de cada estadística. Siendo la fórmula del raiting ofensivo la siguiente:

$$TMOffRtg = 100 * \frac{PTS}{TeamPossessions} \tag{6.1}$$

Dónde PTS son los puntos anotados por el equipo y $Possessions$ son las posesiones ofensivas que ha realizado. Actualmente no hay ningún registro estadístico en los ficheros proporcionados por la NBA que se asocie con las posesiones, es por esto que utiliza la siguiente fórmula para aproximar las posesiones de un equipo (Para comprender el significado de las variables usadas visitar el anexo A):

$$TeamPossessions = FGA - \frac{OREB}{OREB + CDREB} * (FGA - FGM) * 1{,}07 + TOV + 0{,}4 * FTA \tag{6.2}$$

Esta aproximación se realiza teniendo en cuenta que la posesión solo puede acabar de tres formas:

1. Con un tiro de campo fallado cuyo rebote no es capturado por el equipo atacante.
2. Con pérdida (TOV).
3. Con la realización de una falta personal que conlleve tiros libres para el equipo atacante (FTA).

Para calcular el floor porcentaje de un equipo basta con calcular las posesiones del equipo y en cuales de ellas anota al menos un punto:

$$TMFloor\,\% = \frac{TeamScoringPossessions}{TeamPossessions} \tag{6.3}$$

En esta fórmula se introduce un nuevo concepto que es *TeamScoringPossessions*, son las posesiones ofensivas en las que un equipo es capaz de anotar al menos un punto. Para un equipo se utiliza la siguiente aproximación:

$$TeamScoringPossessions = FGM + (1 - (1 - FT\,\%)^2) * FTA * 0{,}4 \tag{6.4}$$

Las fórmulas que se utilizan para calcular estas estadísticas para cada jugador son más complejas que las fórmulas aproximadas que se utilizan para calcular el raiting ofensivo y el floor porcentaje de un equipo. Para poder calcularlas para cada jugador individual las fórmulas se basan en los siguientes conceptos:

- **Posesiones individuales con puntuación**: representa la contribución que realiza un jugador cuando su equipo anota al menos un punto.
- **Posesiones individuales**: representa la contribución de un jugador cuando su equipo termina una posesión.
- **Puntos producidos individualmente**: representa la contribución de un jugador cuando su equipo genera puntos en el ataque.

El concepto que es más difícil de calcular son las posesiones individuales con puntación. Para calcularlo se divide en distintas partes, que son las menaras que tiene un jugador de contribuir a la anotación en una posesión de su equipo. En inglés se conoce como *ScoringPossessions* y su fórmula es (cabe destacar que los campos que aparecen con el prefijo *TM* se refieren a estadísticas del equipo al que pertenece el jugador):

$$\begin{aligned} ScoringPossessions =& (FGPart + ASTPart + FTPart) \\ & * (1 - \frac{TMOR}{TeamScoringPossessions} * TMORWeight * TMPlay\,\%) \\ & + ORPart \end{aligned} \tag{6.5}$$

TeamScoringPossessions se calcula con la fórmula 6.4, ya que son las posesiones en las que anota el equipo del jugador. Encontramos además dos conceptos que debemos calcular. Para calcular el porcentaje de jugadas *TMPlay %* usamos la fórmula:

$$TMPlay\,\% = \frac{TeamScoringPossessions}{TMPlay} \tag{6.6}$$

El porcentaje de jugadas del equipo (*TMPlay %*) hace referencia a las jugadas en las que se anotan al menos un punto. El concepto de jugada es parecido a una posesión, pero

se ve interrumpida por un fallo en el tiro aunque el rebote sea capturado por el equipo atacante. La fórmula que aproxima las jugadas de un equipo es:

$$TMPlay = TMFGA + TMFTA * 0{,}4 + TMTOV \tag{6.7}$$

El operador *TMORWeight* de la fórmula de *ScoringPossessions* (6.5) es una fórmula para calcular la importancia de los rebotes ofensivos en base a lo difícil que son de capturar para el equipo. Se calcula de la siguiente manera:

$$TMORWeight = \frac{(1 - TMOR\%) * TMPlay\%}{(1 - TMOR\%) * TMPlay\% + TMOR\% * (1 - TMPlay\%)} \tag{6.8}$$

Para calcular la parte de tiros de campos (*FGAPart*) se contempla la contribución del jugador en forma de tiros de campos, de manera que se intenta aproximar en función de los puntos anotados y los tiros de campo intentados. La fórmula utilizada es:

$$FGPart = FGM * (1 - \frac{1}{2} * \frac{PTS - FTM}{2 * FGA} * q_{AST}) \tag{6.9}$$

Es necesario calcular el porcentaje de los tiros que son asistidos mediante el operador q_{AST}. Esto se debe a que por convenio se piensa que es más fácil anotar cuando has recibido una asistencia y se debe repartir el "crédito" entre el asistente y el anotador. La fórmula es:

$$q_{AST} = \frac{5 * MIN}{TMMIN} * q_5 + (1 - \frac{MIN}{TMMIN}) * q_{12} \tag{6.10}$$

$$q_5 = 1{,}14 * \frac{TMAST - AST}{TMFGM} \tag{6.11}$$

$$q_{12} = \frac{\frac{TMAST}{TMMIN} * MIN * 5 - AST}{\frac{TFGM}{TMMIN} * MIN * 5 - FGM} \tag{6.12}$$

La ecuación de q se basa en dos aproximaciones con pesos basadas en los minutos que un jugador está en la pista. Si un jugador está durante muchos minutos jugando q_5 es más adecuada a la realidad ya que se basa en la probabilidad de un jugador de anotar un tiro precedido por una asistencia. Para jugadores que juegan de manera infrecuente es mejor la aproximación de q_{12} que se basa en la asunción de una distribución igualitaria de asistencias por minuto.

La parte que contabiliza las asistencias *ASTPart* es calculada mediante la fórmula:

$$ASTPart = \frac{1}{2} * \frac{(TMPTS - TMFTM) - (PTS - FTM)}{2 * (TMFGA - FGA)} * AST \tag{6.13}$$

Para calcular la parte correspondiente a los tiros libres *FTPart* se utiliza una fórmula que podemos encontrar dentro de la fórmula de *TeamScoringPossessions* 6.4:

$$FTPart = (1 - (1 - FT\%)^2) * 0{,}4 * FTA \tag{6.14}$$

La parte entre paréntesis de la fórmula 6.14 indica la fracción de tiros libres en los cuales el jugador anota al menos un tiro libre. Por último nos falta la calcular la parte de los

rebotes ofensivos *ORPart* que utiliza las fórmulas de *TMORWeight* y *TMPaly %*, fórmulas 6.8 y 6.6 respectivamente.

$$ORPart = OR * TMORWeight * TMPlay\,\% \tag{6.15}$$

Con estas fórmulas ya tendríamos calculadas las posesiones individuales con puntuación. Ahora calculamos las posesiones individuales con la siguiente fórmula:

$$Possessions = ScoringPossessions + MissedFGPart + MissedFTPart \tag{6.16}$$

Como se observa, para calcular las posesiones individuales se tienen en cuenta las posesiones en las que se anotan al menos un punto, que se calculan con las fórmulas anteriores (6.5). También hay que añadir en las que no se anotan, que se aproximan con *MissedFGPart* y *MissedFTPart*. Para estas partes se utilizan las fórmulas:

$$MissedFGPart = (FGA - FGM) * (1 - 1{,}07 * TMOR\,\%) \tag{6.17}$$

$$MissedFTPart = (1 - FT\,\%)^2 * 0{,}4 * FTA \tag{6.18}$$

Por último calculamos los puntos producidos por el jugador con la siguiente fórmula:

$$\begin{aligned} PointsProduced =& (FGPart + ASTPart + FTPart) * \\ & * (1 - \frac{TMOR}{TeamScoringPossessions} * TMORWeight * TMPlay\,\%) + \\ & + ORPart \end{aligned} \tag{6.19}$$

Esta fórmula es igual que la de la posesiones individuales con puntación (6.5), lo único que cambia es cómo se calcula cada parte. La parte de que hace referencia a los tiros libres (*FTPart*) se calcula de la misma manera por lo que usamos la fórmula 6.14. *TMORWeight* se calcula con la fórmula 6.8 y *TMPlay %* con la fórmula 6.6. El cálculo de las demás partes es más sencillo que el visto para las posesiones individuales con puntación. La fórmulas para la parte de tiros de campo (*FGPart*) es:

$$FGPart = 2 * (FGM + \frac{1}{2} * FG3M) * (1 - \frac{1}{2} * \frac{PTS - FTM}{2 * FGA} * q_{AST}) \tag{6.20}$$

En este caso q_{AST} se calcula con la fórmula 6.10 que también se utiliza en el cálculo de las posesiones individuales con puntuación. Para la parte de puntos producidos mediante asistencias (*ASTPart*) utilizamos una fórmula en la que se tienen en cuenta los triples:

$$\begin{aligned} ASTPart =& \frac{TMFGM - FGM + 0{,}5 * (TMFG3M - FG3M)}{TMFGM - FGM} * \frac{1}{2} \\ & * \frac{(TMPTS - TMFTM) - (PTS - FTM)}{2 * (TMFGA - FGA)} * AST \end{aligned} \tag{6.21}$$

Por último falta calcular la parte de rebotes ofensivos capturados por el jugador y que ayudan a que el equipo anote una canasta. La fórmula para aproximar este valor es:

$$\begin{aligned} ORPart =& OR * TMORWeight * TMPlay\,\% \\ & * \frac{TMPTS}{TMFGM + (1 - (1 - TMFT\,\%)^2) * 0{,}4 * TMFTA} \end{aligned} \tag{6.22}$$

En esta fórmula aparecen de nuevo los términos *TMORWeight* y *TMPlay %* que se calculan respectivamente con las fórmulas 6.8 y 6.6. Una vez hemos calculado el valor de los tres conceptos tenemos que las fórmulas resultantes para el cálculo del raiting ofensivo *OffRtg* y el floor porcentaje *Floor %* para un jugador son:

$$OffRtg = 100 * \frac{PointsProduced}{Possessions} \tag{6.23}$$

$$Floor\,\% = \frac{ScoringPossessions}{Possessions} \tag{6.24}$$

Raiting Defensivo

Este término también fue definido por Dean Oliver en su libro [21] y da la siguiente definición para el rating defensivo individual: "Fundamentalmente, el rating defensivo individual intenta determinar cuántos puntos permite el jugador por cada cien posesiones". Para definir el rating defensivo de un equipo se puede usar la misma definición: Cuántos puntos permite por cada cien posesiones. La fórmula para el rating defensivo de un equipo es:

$$TeamDefRtg = 100 * \frac{CPTS}{TeamPossessions} \tag{6.25}$$

Para la fórmula del raiting defensivo 6.25 se necesitan los puntos concedidos del equipo (*CPTS*) y se divide entre el total de posesiones del equipo. La fórmula individual es más compleja ya que intervienen más factores, ya que se debe aproximar los puntos que permite un jugador individual. Se tiene en cuenta el raiting defensivo del equipo ya que se asume que todos los jugadores impactan en cierto grado en las posesiones defensivas. Se calcula entonces con la siguiente fórmula:

$$\begin{aligned} DefRtg =& TeamDefRtg \\ &+ 0{,}2 * (100 * CPtsPerScPoss * (1 - Stop\,\%) - TeamDefRtg) \end{aligned} \tag{6.26}$$

El término *CPtsPerScPoss* son los puntos por cada posesión que le anotan al equipo al que pertenece el jugador. Para aproximar este valor se utilizan estadísticas simples que deben ser almacenadas cuando se analizan los partidos. La fórmula que se utiliza es (El prefijo *C* delante de una estadística hace referencia a que son registros de los oponentes durante los partidos):

$$CPtsPerScPoss = \frac{CPTS}{CFGM + (1 - \frac{CFTM}{CFTA})^2 * CFTA * 0{,}4} \tag{6.27}$$

El otro término que apetece es *Stop %* que indica el porcentaje de paradas. Se entiende por parada cuando un jugador detiene la posesión ofensiva del equipo rival, de manera que no se ha anotado ningún punto. La fórmula que se utiliza se basa en los minutos jugados por el jugador y las posesiones totales del equipo

$$Stop\,\% = \frac{Stops * TMMIN}{TMPOSS * MIN} \tag{6.28}$$

El término *Stops* es la aproximación de las paradas que realiza cada jugador. Debido a que no se registran estadísticas específicas sobre las paradas de cada jugador se realiza una aproximación dividiendo la fórmula en dos partes:

$$Stops = Stops_1 + Stops_2 \tag{6.29}$$

La primera parte ($Stops_1$) estima las paradas mediante tapones y robos, estadísticas que se registran en los ficheros play-by-play. Permite identificar una fracción de las paradas en las que el jugador es el responsable en base a estas estadísticas.

$$Stops_1 = STL + BLK * FMWeight * (1 - 1{,}07 * COR\%) + DREB * (1 - FMWt) \tag{6.30}$$

La segunda parte ($Stops_2$) estima cuántas pérdidas y fallos provoca un jugador y no han sido registradas mediante robos o tapones. Esta parte de la fórmula se basa en estadísticas de equipo y asume que todos los jugadores son igual de buenos forzando pérdidas y fallos por minuto.

$$\begin{aligned} Stops_2 =& (\frac{CFGA - CFGM - TMBLK}{TMMIN} * FMWeight * (1 - 1{,}07 * COR\%) \\ &+ \frac{CTO - TMSTL}{TMIN}) * MIN + \frac{PF}{TMPF} * 0{,}4 * CFA * (1 - CFT\%)^2 \end{aligned} \tag{6.31}$$

En ambas partes aparece el término *FMWeight* que representa la dificultad de forzar tiros fallados contra la dificultad de atrapar rebotes defensivos:

$$FMWeight = \frac{CFG\% * (1 - COR\%)}{CFG\% * (1 - COR\%) + (1 - CFG\%) * COR\%} \tag{6.32}$$

Net Raiting

El Net Raiting se calcula mediante la diferencia del raiting ofensivo y el raiting defensivo. Mediante esta estadística se mide el diferencial de puntos de un equipo por cada cien posesiones. Para un jugador indica el diferencial de puntos del equipor por cada cien posesiones cuando el jugador está en la pista. La fórmula es muy sencilla:

$$NetRtg = OffRtg - DefRtg \tag{6.33}$$

Ritmo

El Ritmo se suele medir para equipos de manera que indica el número de posesiones en un partido o el número de posesiones por partido que realiza un equipo. Si se obtiene un valor de 100 en un partido específico indica que en ese partido ha habido cien posesiones. Un valor de 100 por partido indica que el equipo promedia cien posesiones por partido.

La fórmula que se utiliza utiliza conceptos previamente definidos, las posesiones. Tanto para calcular las posesiones del equipo como para las posesiones del equipo oponente se utiliza la fórmula de *TeamPossessions* 6.4. La fórmula del ritmo es:

$$Pace = 48 * \frac{TeamPossesions + OponentTeamPossessions}{2 * \frac{TMMIN}{5}} \tag{6.34}$$

El valor de 48 son los minutos totales de un partido de baloncesto, en los que se juegan cuatro cuartos de doce minutos cada uno.

Porcentaje Real de tiro

El Porcentaje Real de Tiro es una métrica que pretende identificar la capacidad de un jugador para anotar. Surgió como alternativa al clásico porcentaje de tiros de campo ya que en este no se tienen en cuenta los tiros libres. Este nuevo porcentaje trata de tener en cuenta la dificultad de anotar un triple y un tiro de campo de dos puntos frente a un tiro libre. Esta estadística es un indicador aproximado y tiene problemas en ciertas situaciones extremas, por ejemplo, un jugador que anote muchos triples puede tener un porcentaje real de tiro por encima de cien. La fórmula que se utiliza es la siguiente:

$$TS\,\% = \frac{PTS}{2 * (FGA + 0{,}44 * FTA)} \tag{6.35}$$

Porcentaje de tiros de campo efectivo

Este porcentaje de tiro trata de poner en valor los triples anotados por un jugador, ya que un triple da tres puntos. Para calcular este valor se ponderan los triples por encima del resto de tiros, ya que son los que más beneficio generan. Esta estadística surgió ya que para anotar una cantidad de, por ejemplo, 12 puntos se puede hacer anotando 4 triples o 6 tiros de dos puntos. Si el porcentaje de tiros tradicional en ambos casos es del 50 % en el primer caso habremos lanzado sólo 8 triples y en el segundo habremos lanzado 12 tiros de dos puntos. Como se observa con este ejemplo, los tiros de tres aportan mayor beneficio a la anotación, por eso reciben una mejor ponderación. La fórmula que se utiliza es:

$$eFG\,\% = \frac{2FGM + (1{,}5 * 3FGM)}{FGA} \tag{6.36}$$

Hay formas más simplificadas de esta estadística, pero esta es la más usada por los equipos de la NBA y por la propia NBA.

Ratio de Tiros Libres intentados

Esta estadística indica el volumen de tiros libres que se realizan sobre el total de tiros de campo realizados. Esto permite evaluar con qué frecuencia un equipo o jugador recibe falta y lanza tiros libres. La fórmula que se utiliza es bastante simple:

$$FTARate = \frac{FTA}{FGA} \tag{6.37}$$

Ratio de Triples intentados

Al igual que en el caso del ratio de tiros libres intentados, esta estadística indica el volumen de triples intentados sobre el total de tiros de campo realizados. Con esta estadística evaluamos la frecuencia con la que se lanzan triples por cada tiro de campo intentado. La fórmula es:

$$3FGARate = \frac{3FGA}{FGA} \tag{6.38}$$

Porcentaje total de Rebotes Ofensivos

Esta estadística indica el porcentaje de rebotes ofensivos capturados sobre el total de los disponibles. El total de los rebotes ofensivos se calcula sumando los rebotes ofensivos del equipo y los rebotes defensivos del equipo rival.

$$TMOR\,\% = 100 * \frac{TMOR}{TMOR + CDREB} \tag{6.39}$$

Para un jugador es necesario realizar una aproximación un poco más sofisticada para aproximar los rebotes ofensivos disponibles teniendo en cuenta los minutos que ha estado en pista.

$$OR\,\% = 100 * \frac{OR * \frac{TMIN}{5}}{MIN * (TMOR + CDREB)} \tag{6.40}$$

Porcentaje total de Rebotes Defensivos

Esta es la contra parte de la estadística anterior, indica el porcentaje de rebotes defensivos capturados sobre el total disponible. En este caso el total disponible se calcula sumando los rebotes defensivos del equipo y los rebotes ofensivos concedidos al equipo rival.

$$TMDR\,\% = 100 * \frac{TMDR}{TMDR + COREB} \tag{6.41}$$

A igual que en el caso anterior para calcular esta estadística de manera individual se debe de hacer la aproximación sobre los rebotes disponibles durante los minutos que el jugador ha disputado.

$$DR\,\% = 100 * \frac{DR * \frac{TMIN}{5}}{MIN * (TMDR + COREB)} \tag{6.42}$$

Porcentaje de Tapones

Esta estadística indica el volumen total de tapones que se han realizado sobre el total de tiros del oponente. Para calcular este valor para un equipo simplemente se dividen los tapones del equipo entre el total de tiros realizados por los oponentes.

$$TMBLK\,\% = 100 * \frac{TMBLK}{CFGA - 3FGA} \tag{6.43}$$

Si queremos calcular esta estadística para un jugador debemos tener en cuenta el factor de los minutos jugados por este jugador, quedando como resultado la siguiente fórmula:

$$BLK\,\% = 100 * \frac{BLK * \frac{TMMIN}{5}}{MIN * (CFGA - C3FGA)} \tag{6.44}$$

Porcentaje de Pérdidas

Con esta estadística se trate de aproximar la cantidad de pérdidas que se cometen cada cien posesiones. Para calcular este valor se tienen en cuenta los tiros de campo y tiros libres intentados, la fórmula es la siguiente:

$$TOV\,\% = 100 * \frac{TOV}{FGA + 0{,}44 * FTA + TOV} \tag{6.45}$$

Porcentaje de Asistencias

Para un equipo indica la cantidad de asistencias realizadas sobre el total de tiros anotados. Para un jugador indica también el total de asistencias repartidas sobre el total de los tiros de campo anotados por el resto del equipo, teniendo en cuenta los minutos en los que el jugador ha estado sobre la pista.

$$TMAST\,\% = 100 * \frac{TMAST}{TMFGM} \tag{6.46}$$

$$AST\,\% = 100 * \frac{AST}{(\frac{MIN*5}{TMMIN} * TMFGM) - FGM} \tag{6.47}$$

Ratio de Asistencias Pérdidas

Este ratio surge para identificar la cantidad de asistencias que se reparten por cada balón que se ha perdido. Es una estadística muy simple de calcular, se calcula e la misma manera para un jugador que para un equipo, pero ofrece información muy preciada sobre la habilidad de manejo del balón y distribución del juego. Si un jugador tiene la habilidad de asistir mucho a sus compañeros, habiendo cometido pocas pérdidas, el ratio será alto. Por el contrario, si un jugador comete más pérdidas y no es capaz de encontrar a sus compañeros para asistirles en una canasta el ratio será menor.

$$AST/TOV = \frac{AST}{TOV} \tag{6.48}$$

Porcentaje de Robos

Con esta estadística se mide el porcentaje de las posesiones del rival que han sido acabadas con un robo. Para calcular este valor para un jugador debemos tener en cuenta el tiempo que éste ha estado en la cancha.

$$TMSTL\,\% = 100 * \frac{TMSTL}{OpponentTeamPossessions} \tag{6.49}$$

$$STL\,\% = 100 * \frac{STL * \frac{TMMIN}{5}}{MIN * OpponentTeamPossessions} \tag{6.50}$$

Para calcular las posesiones del equipo rival *OpponentTeamPossessions* usamos la fórmula 6.4.

Porcentaje de Uso

Esta estadísticas es una aproximación del porcentaje de jugadas finalizadas por un jugador. La forma de finalizar una jugada es mediante un tiro de campo, un tiro libre o una pérdida. Normalmente este valor se asocia a la influencia ofensiva de un jugador dentro de su equipo. Cuanto mayor sea el porcentaje mayor es su incidencia. La fórmula que se utiliza para calcularlo es:

$$USG\,\% = \frac{(FGA + 0{,}44 * FTA + TOV) * \frac{TMMIN}{5}}{MIN * (TMFGA + 0{,}44 * TMFTA + TMTOV)} \tag{6.51}$$

6.3. Selección del modelo de predicción de partidos

Para predecir el resultado de un partido se entrenará un modelo de aprendizaje automático, de manera que en función de los datos de entrada clasificará el partido en victoria o derrota del equipo local. Los datos de entrada serán las estadísticas avanzadas de ambos equipos. Para definir los modelos y procesos necesarios para el aprendizaje automático se utilizará el paquete *sklearn*.

Para encontrar un modelo que se adecue al problema planteado se evaluarán distintos modelos para la clasificación. Los modelos evaluados serán:

- ***Logistic Regression***. La regresión logística es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica [22]. Este modelo es muy útil para modelar la probabilidad de un evento, en este caso la victoria del equipo local, en función de otros estadísticas, las estadísticas avanzadas de los equipos que participan en el partido. Se probarán distintos algoritmos de optimización (*solver*) para tratar de encontrar el que mejor se ajuste al conjunto de datos.
- ***Support Vector Machines***. Las máquinas de soporte vectorial son modelos que construyen un hiperplano o conjunto de hiperplanos en un espacio de dimensionaliad muy alta que puede ser utilizado en problemas de clasificación. Una buena separación entre las clases permitirá una clasificación correcta [23].
- ***Random Forest***. Los Bosques Aleatorios son modelos que se forman como combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de éstos. Se construye una larga colección de árboles no correlacionados y luego los promedia [24].
- ***Neural Network***. Las Redes Neuronales son modelos computacionales basados en un conjunto de unidades, llamadas neuronas, conectadas entre sí para transmitirse señales. Las conexiones entre neuronas son conocidas como enlaces. En los enlaces el valor de salida de la neurona es multiplicado por un valor de peso. En la salida de las neuronas puede haber una función limitadora o umbral, que modifica el valor del resultado. Esta función se conoce como función de activación. Cuando la información de entrada atraviesa la red se producen unos valores de salida. Se utilizarán distintos algoritmos de optimización de pesos (*solver*) [25].

	Local_TmOffRtg	Local_TmFloor%	Local_TmDefRtg	Local_Pace	Local_TS%	Local_eFG%	Local_FTARate	Local_3FGARate	Local_TmOR%	Local_TmD
0	106.326895	0.480894	100.864813	93.274918	0.579954	0.514913	0.241137	0.316263	20.654628	74.140753
1	106.326895	0.480894	100.864813	93.274918	0.579954	0.514913	0.241137	0.316263	20.654628	74.140753
2	106.326895	0.480894	100.864813	93.274918	0.579954	0.514913	0.241137	0.316263	20.654628	74.140753
3	106.326895	0.480894	100.864813	93.274918	0.579954	0.514913	0.241137	0.316263	20.654628	74.140753
4	106.326895	0.480894	100.864813	93.274918	0.579954	0.514913	0.241137	0.316263	20.654628	74.140753

Figura 6.4: Conjunto de datos para entrenar el modelo de predicción.

Datos de entrada.

Antes de comenzar a definir los modelos es importante analizar el formato de los datos de entrada. Para predecir el resultado de un partido se utilizan las estadísticas avanzadas del equipo local y el equipo visitante. Para generar el conjunto de datos para entrenar los modelos se calculan las estadísticas avanzadas de los equipos que participan en cada partido. Si el equipo local ha ganado el partido se clasifica con el valor de salida 1. El conjunto de datos generado consta de 26 columnas, que son las estadísticas avanzadas de ambos equipos, y 632 filas, que son los partidos disputados de los que cuyos datos tenemos disponibles.

Como se observa en la figura 6.4, los datos de entrada no están en la misma escala. Esta diferencia de escala sobre las variables de entrada puede aumentar la dificultad para generalizar de los modelos.

Para evitar este problema se realizará un escalado, de tal manera que se eliminará la media y se escala a la varianza unitaria. Para una entrada x su valor escalado se calcula de la siguiente manera:

$$z = \frac{x - u}{s} \tag{6.52}$$

Una vez que tenemos los datos escalados podemos proceder a construir los distintos modelos de predicción.

Evaluación de modelos predictivos.

Para poder evaluar los modelos es necesario estimar la precisión de los modelos se utiliza una técnica conocida como validación cruzada (*Cross Validation*). Esta técnica permite asegurar que los resultados obtenidos son independientes a la partición del conjunto de datos original en datos de entrenamiento y datos de prueba. Consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones [26].

Más concretamente, se utilizará la técnica conocida como ***KFold***. Los datos son divididos en K subconjuntos. Uno de los subconjuntos es utilizado como datos de prueba y el resto (*K-1*) como datos de entrenamiento. Este proceso se repite durante K iteraciones, con cada uno de los posibles subconjuntos. La puntuación de cada modelo durante cada iteración será almacenada.

Una vez que se han realizado todas las iteraciones se calcula la puntuación media de cada modelo y la desviación típica. Estas medidas ayudan a entender qué modelo es más robusto. Los datos obtenidos son:

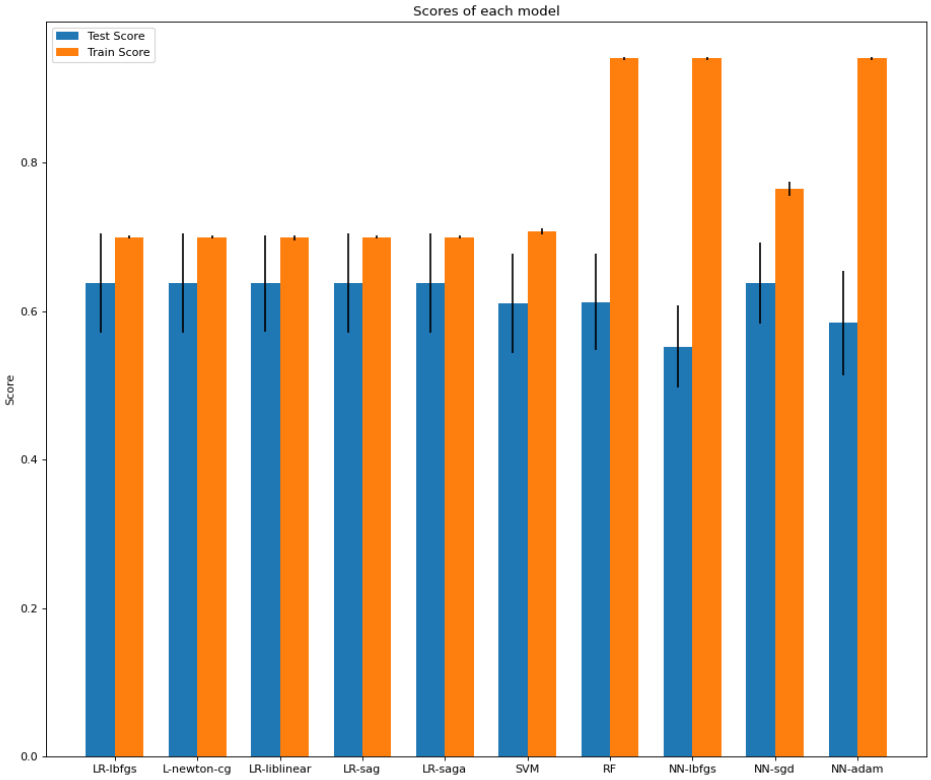


Figura 6.5: Puntuación media y desviación típica de los modelos evaluados.

Los modelos que aparecen en la figura 6.5 son los siguientes:

- LR-lbfgs: Regresión Lógica con algoritmo de optimización *lbfgs*.
- LR-newton-cg: Regresión Lógica con algoritmo de optimización *newton-cg*.
- LR-liblinear: Regresión Lógica con algoritmo de optimización *liblinear*.
- LR-sag: Regresión Lógica con algoritmo de optimización *sag*.
- LR-saga: Regresión Lógica con algoritmo de optimización *saga*.
- SVM: Máquinas de Soporte Vectorial.
- RF: Bosques Aleatorios.
- NN-lbfgs: Red Neuronal con algoritmo de optimización de pesos *lbfgs*.
- NN-sgd: Red Neuronal con algoritmo de optimización de pesos *sgs*.
- NN-adam: Red Neuronal con algoritmo de optimización de pesos *adam*.

Como se observa en la figura 6.5, los modelos basados en redes neuronales y bosques aleatorios tienen mucha mayor precisión en el conjunto de entrenamiento que la que tienen en el conjunto de pruebas. Este suceso es conocido como sobreajuste (*overfitting*),

y se produce cuando el modelo no es capaz de generalizar la información y se producen resultados pobres sobre los datos que el modelo no ha conocido durante el entrenamiento.

Para seguir evaluando los modelos se generan las matrices de confusión. Una matriz de confusión es una herramienta que permite la visualización de la actuación de un modelo de aprendizaje automático. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real [27].

Las matrices de confusión permiten detectar cuatro factores importantes de un modelo de predicción.

- Positivo Real (*true positive*). Cuando el clasificador produce como salida la clase positiva (clase 1) para una entrada etiquetada como clase positiva.
- Falso positivo (*false positive*). Cuando el clasificador produce como salida la clase positiva (clase 1) para una entrada etiquetada como clase negativa (clase 0).
- Negativo real (*true negative*). Cuando el clasificador produce como salida la clase negativa para una entrada etiquetada como clase negativa.
- Falso negativo (*false negative*). Cuando el clasificador produce como salida la clase negativa para una entrada etiquetada como clase positiva.

Mediante esta gráfica se pretende identificar qué modelo consigue más positivos reales y negativos reales. La matriz de confusión son de la forma:

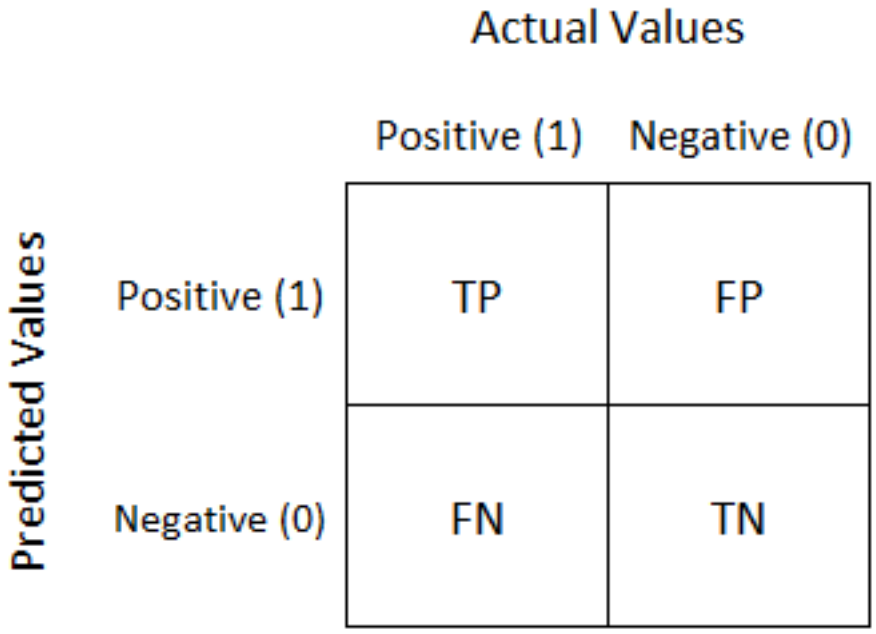


Figura 6.6: Estructura de una matriz de confusión

Para el modelo de regresión lógica se generan una matriz de confusión para cada algoritmo de optimización, las matrices obtenidas son las siguientes:

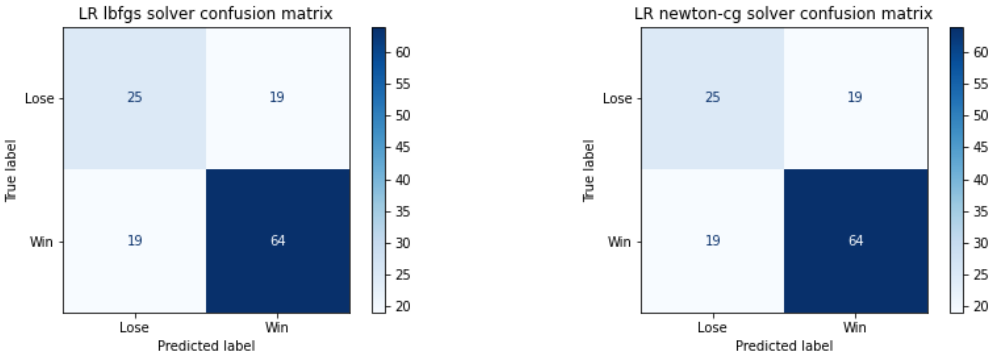


Figura 6.7: Matriz de confusión Regresión Lógica lbfgs solver (izquierda) y Matriz de confusión Regresión Lógica liblinear solver (derecha).

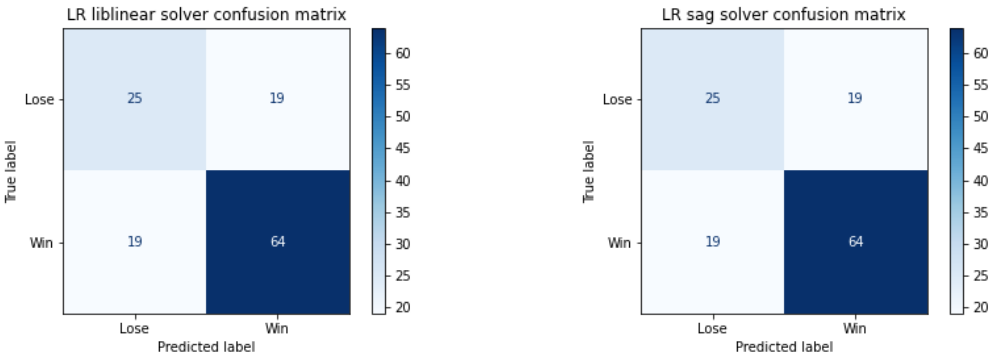


Figura 6.8: Matriz de confusión Regresión Lógica liblinear solver (izquierda) matriz de confusión Regresión Lógica sag solver (derecha).

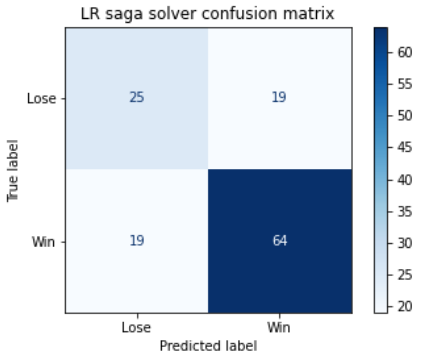


Figura 6.9: Matriz de confusión Regresión Lógica saga solver (dercha).

Como se observan en las matrices de confusión, el modelo de regresión lógica se comporta de igual manera para los distintos algoritmos de optimización.

Para las máquinas de soporte vectorial obtenemos la matriz de confusión 6.10. Este modelo es capaz de detectar más positivos reales pero aumenta ligeramente el número de falsos positivos respecto a la regresión lógica. En la matriz de confusión de bosques aleatorios se observa una mejora notable respecto a los modelos anteriores, ya que aumenta en gran medida el número de negativos reales y disminuyen los falsos positivos.

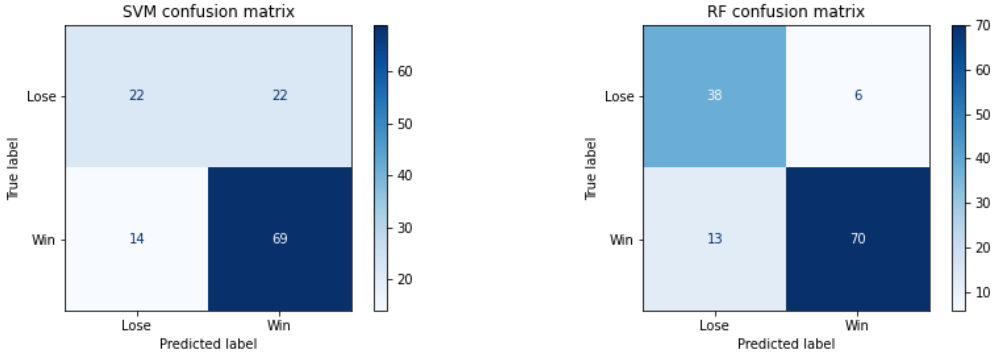


Figura 6.10: Matriz de confusión Máquina de Soporte Vectorial (izquierda) y Matriz de confusión Bosques Aleatorios (derecha).

Para la red neuronal también se genera una matriz de confusión para los distintos algoritmos de optimización de pesos. Las matrices obtenidas son:

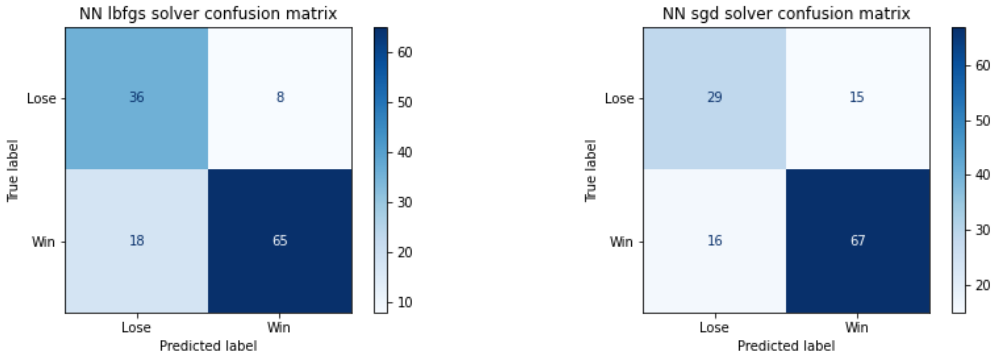


Figura 6.11: Matriz de confusión Red Neuronal lbfgs solver (izquierda) y matriz de confusión Red Neuronal sgd solver (derecha).

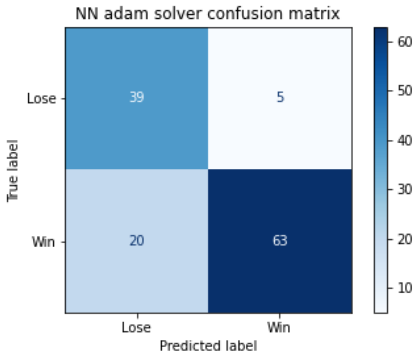


Figura 6.12: Matriz de confusión Red Neuronal adam solver.

Las redes neuronales si que muestran pequeñas diferencias según el algoritmo seleccionado. En general son resultados similares entre sí. Estos modelos producen menos falsos positivos que la regresión lógica y las máquinas de soporte vectorial, pero producen más falsos negativos.

Para seleccionar el modelo más adecuado hay que tener en cuenta la precisión media y las matrices de confusiones. Tendiendo en cuenta la precisión media de los modelos la regresión lógica y las máquinas de soporte vectorial se ajustan mejor a los datos ya que la precisión en el conjunto de entrenamiento y el de pruebas es similar. El modelo basado en bosques aleatorios sufre de sobreajsute en los datos de entrenamiento, aún así la precisión en el conjunto de pruebas es bastante aceptable ya que está al nivel de la regresión lógica y las máquinas de soporte vectorial.

Observando las matrices de confusión, el modelo que despunta es el de bosques aleatorios. Este modelo consigue el mayor numero de positivos y negativos reales entre todos los modelos evaluados.

Para construir un modelo más robusto se creará un modelo conocido como clasificador por votación (*Voting Classifier*). Este modelo combinación de distintos clasificadores y tiene dos posibles funcionamientos:

1. ***Hard Voting***. En este modo el modelo por votación produce como salida la clasificación que más se ha dado en los modelos que lo conforman. Es decir, es como si cada modelo votase a una clase, y el resultado es la clase más votada. Para este modo son necesarios al menos tres modelos.

2. ***Soft Voting***. Cuando el modelo por votación funciona en este modelo el resultado producido es la probabilidad media de la clase positiva que han producido los modelos que lo conforman.

Para la implementación del modelo por votación usaremos los modelos de regresión lógica con algoritmo de optimización *lbgfs*, el modelo de máquinas de soporte vectorial y el modelo de bosques aleatorios. Para decidir qué método (*hard voting* o *soft voting*) utilizar para la predicción de modelos generamos la matriz de confusión para cada método. Las matrices obtenidas son las que se presentan en la figura 6.13. Como se observa en las figuras, ambos modelos tienen una matriz bastante similar. La diferencia es mínima, el *hard voting* es capaz de realizar más predicciones positivas reales (dos más en concreto) a pesar de realizar más falsos positivos. En contra posición, el *soft voting* es capaz de

realizar más negativos verdaderos y más falsos negativos.

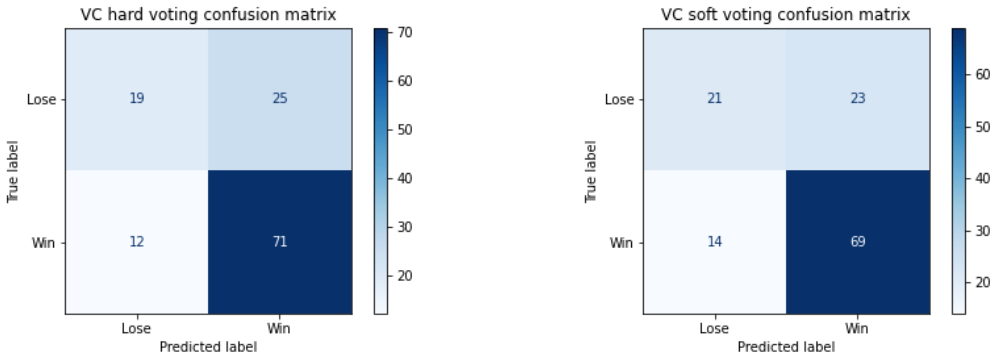


Figura 6.13: Matriz de confusión Clasificador por Votación, *hard voting* (izquierda), y matriz de confusión Clasificador por Votación, *soft voting* (derecha).

Observando todas las medidas realizadas se puede concluir que el modelo que más se adecua al problema es el modelo basado en bosques aleatorios. Este modelo a pesar de mostrar sobreajuste logra una precesión media en el conjunto de pruebas en torno al 70 %. También se observa en la matriz de confusión que es el modelo que más valores verdaderos, tanto positivos como negativos, consigue entre todos los modelos evaluados. Para demostrar el buen funcionamiento del modelo se ha realizado una gráfica con la precisión para los partidos jugados por cada equipo, gráfica 6.14. Las puntuaciones por equipo superan todas el 70 %, y para la mayoría de equipos supera incluso el 80 %. El modelo de bosques aleatorios será el elegido para realizar las predicciones y enviar el resultado a través de la API.

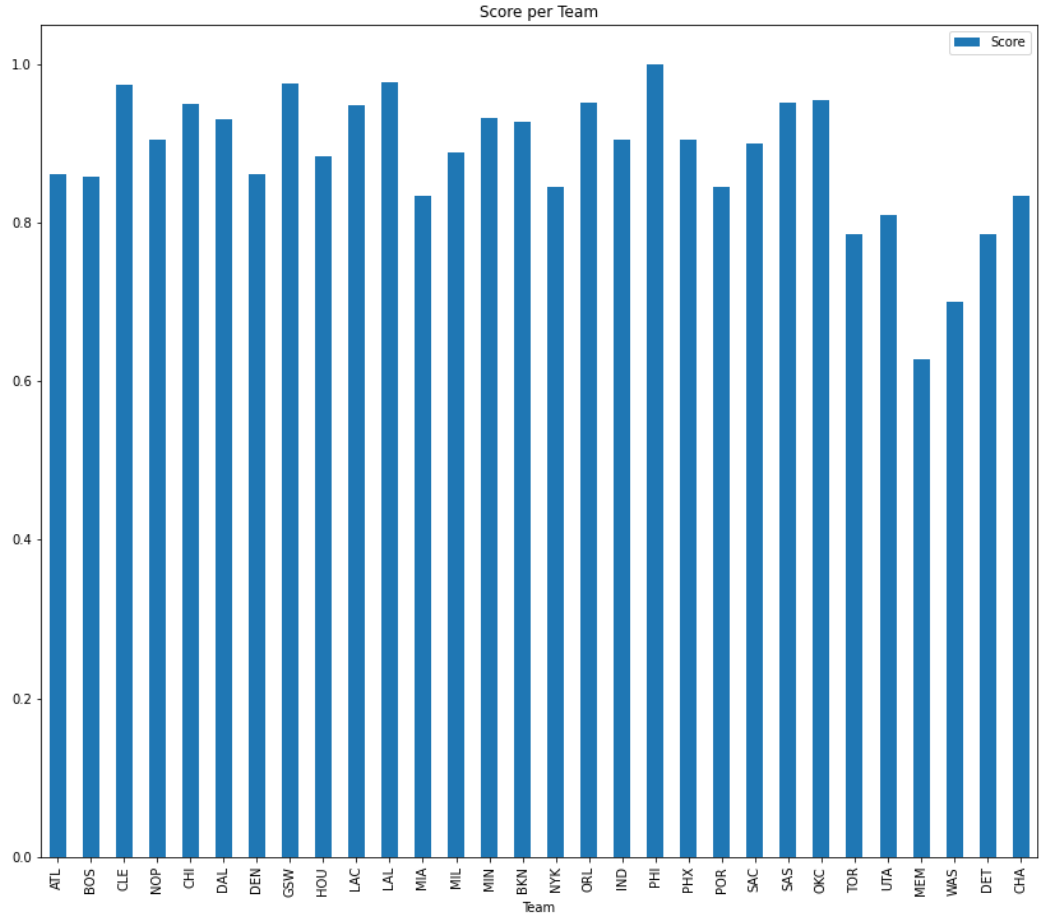


Figura 6.14: Precisión del modelo de Bosques Aleatorios para los partidos de cada equipo.

6.4. Clusterización de los jugadores por posición.

La clusterización o **Anális de Grupos** es la tarea de de agrupar un conjunto de objetos de tal manera que los miembros del mismo grupo sean similares [28]. Se pretende aplicar esta técnica a los jugadores de la NBA para descubrir similitudes en el estilo de juego. Este análisis se realizará por cada una de las cinco posiciones clásicas del baloncesto. Para agrupar los distintos jugadores de cada posición se utilizarán las estadísticas simples promediadas por partidos y las estadísticas avanzadas.

Para realizar la clusterización se probarán tres algoritmos distintos y se estudiará cuál de ellos actúa mejor en el conjunto de datos. Los algoritmos que se evaluarán son:

- **K-Means**: Este algoritmo construye una partición de los datos de entrada en k conjuntos. Inicialmente el algoritmo selecciona los k **centroides** iniciales, los centroides serán los centros de cada grupo. Después asigna a cada jugador al grupo con la media más cercana (**Paso de asignación**) y por último calcula los centroides de cada grupo, calculando las medias de cada grupo (**Paso de actualización**) [29].
- **DBSCAN**: El nombre de este algoritmo proviene de la siglas en inglés de agrupamiento espacial basado en densidad de aplicaciones con ruido (*Density-based spatial clustering of applications with noise*). Es un agrupamiento basado en densidad porque encuentra un número de grupos comenzando por estimar la distribución de la densidad de los nodos correspondientes. El algoritmo visita un punto arbitrario que no haya sido visitado, si se encuentra en una zona densa se inicia un clúster sobre el mismo, de lo contrario será etiquetado como ruido. Si se visita un punto que ya se encuentra en la parte densa de un clúster, se añada a él [30].
- **OPTICS**: Proviene del inglés *Ordering points to identify the clustering structure*. Este algoritmo puede ser considerado como una generalización de DBSCAN. Los datos son agrupados de acuerdo a las estructuras de los clústers, de manera que los puntos especialmente cercanos acaban siendo vecinos en la ordenación [31].

Los datos que se utilizarán como entrada serán las estadísticas de los jugadores. Para realizar distintas pruebas con los algoritmos de clusterización variando las estadísticas de cada jugador que son proporcionadas a los algoritmos. Las estadísticas de los jugadores son separadas en simples y avanzadas, ambos tipos de estadísticas serán usadas conjuntamente y por separado para entrenar los modelos.

Con los algoritmos DBSCAN Y OPTICS no se han obtenido resultados concluyentes. En ambos algoritmos existen dos parámetros claves en la agrupación que realizan dichos algoritmos:

- ***eps***: es la distancia máxima que existe entre dos puntos para que se consideren que están en la misma vecindad.
- ***metric***: es la métrica que se utilizará cuando se calculan las distancias.

Se han probado las métricas disponibles para estos algoritmos, estos valores pueden encontrarse en la página de sklearn [32]. También se han probado valores mayores que el valor por defecto del parámetro *eps*, que es 0.5, para que se formen clúster con mayor facilidad. En el mejor de los casos los algoritmos eran capaces de formar tres grupos, pero

la mayoría de jugadores eran considerado ruido. Es por esto que se desestimó el uso de estos algoritmos.

K-Means

Para el algoritmo de K-Means es necesario determinar previamente el número de clústers, k, óptimo para el conjunto de datos. Para averiguar qué valor es más adecuado se utiliza el método conocido como ***Elbow Curve***, este método se basa en mostrar en una gráfica el valor de inertia del modelo en función del número de clústers. La inertia es una métrica que mide la distancia entre los puntos que forman cada clúster. Como es lógico hay que tratar de minimizar el valor de la inertia pero manteniendo un número lógico de clúster. En la gráfica se buscará en qué valor de k el descenso del valor de la inertia empieza a reducirse.

Al ejecutar el algoritmo con las estadísticas simples promedio de cada jugador como entrada los clúster que se forman parecen indicar el estatus de cada jugador. Los clúster se forman por peso de los jugadores en su equipo, los minutos jugados por partido parecen tener mucho peso en la formación de los grupos. Por ejemplo, al realizar la agrupación con estas estadísticas sobre los bases obtenemos que uno de los clúster contiene a los siguientes jugadores:

Cluster	first_name	last_name
0	Isaiah	Thomas
0	Stephen	Curry
0	Chris	Paul
0	Eric	Bledsoe
0	Damian	Lillard
0	Russell	Westbrook
0	Kyle	Lowry
0	John	Wall
0	Kemba	Walker

Figura 6.15: Clúster formado usando estadísticas simples.

Todos los jugadores que se incluyen en el clúster son súper estrellas de la NBA en la posición de base. No todos estos jugadores tienen el mismo estilo, Stephen Curry basa su anotación en los lanzamientos de tres puntos y Russel WestBrook es un jugador más completo y que en ataque realiza penetraciones hacia canasta para poder anotar lo más cerca del aro posible. Los demás clústers se forman de manera similar, es por esto que se decidió a utilizar únicamente las estadísticas avanzadas.

Con la información del clúster al que pertenece cada jugador se pretende construir un sistema que proporcione soporte a los entrenadores para preparar la defensa de su equipo. Para ello se analizan los ficheros con las coordenadas de los eventos de cada partido. De

cada partido se extraen los eventos relacionados con los tiros fallados y acertados. Para cada tiro realizado se busca al defensor más cercano del equipo rival. Una vez que se identifica al defensor se guarda para cada tiro la información del jugador que realiza el tiro y el defensor, y si el tiro ha sido anotado o no.

Esta será la base para la creación del sistema, que será capaz de identificar a qué clúster defiende mejor cada integrante del equipo. Para cada defensor se identificará a qué tipo de jugador, según al clúster que pertenece, defiende mejor. Para un entrenador esta información es muy útil, ya que permite hacerse una idea de qué debilidades tiene su equipo. Identificar las debilidades del equipo permite enfocar esfuerzos a la hora de preparar enfrentamientos.

Para averiguar el número de clúster adecuado para los bases se realizan varios modelos variando el valor de k, desde 1 hasta 20, y mostrando en una gráfica el valor de inertia de cada uno de ellos (*Elbow Curve*). El resultado obtenido es:

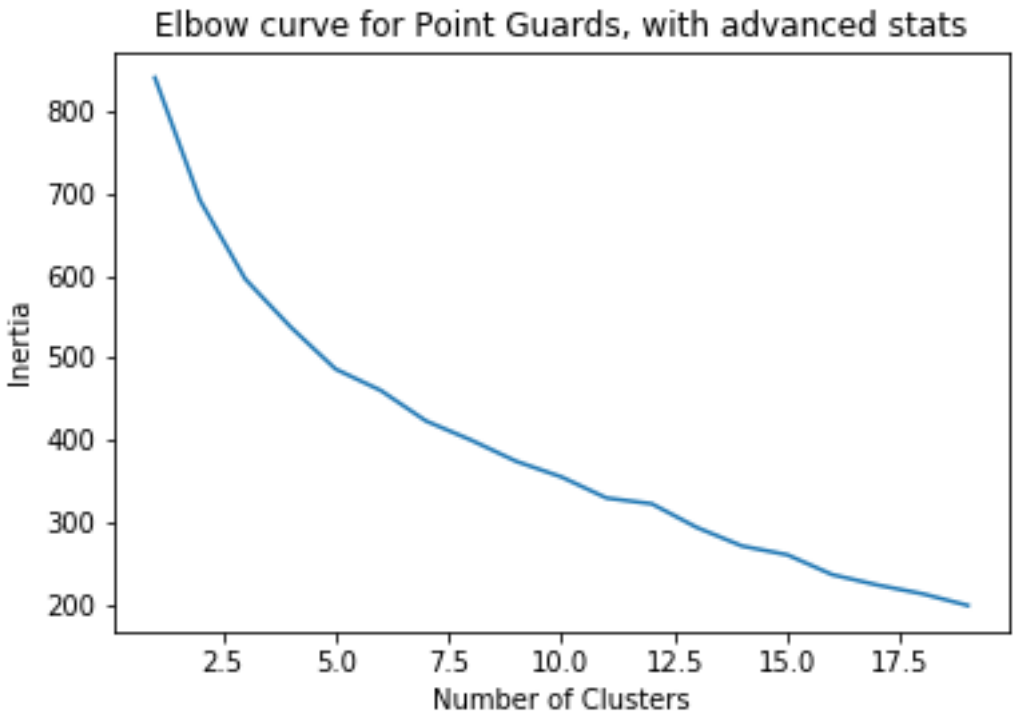


Figura 6.16: Elbow Curve para K-Means usando las estadísticas avanzadas de los bases.

Observando la curva se estima que el número adecuado de clúster es 5. A partir de este valor el descenso en el valor de la inertia se suaviza. Al construir el modelo de K-Means con k=5 la distribución obtenida es la siguiente:

Cluster	Players
0	6
1	20
2	5
3	11
4	14

Figura 6.17: Distribución de los bases miembros de cada clúster.

Las estadísticas medias de los clústers formados para los bases son:

Clúter	0	1	2	3	4
Etiqueta	Scorer Playmaker	Secondary Scorer	Playmaker	Accurate Scorer	Defender
OffRtg	118.5	107.67	120.12	124.35	126.02
Floor %	0.35	0.33	0.38	0.39	0.36
DefRtg	113.59	108.32	103.31	105.34	106.66
NetRtg	4.94	-0.66	16.81	19.01	19.36
TS %	0.55	0.52	0.53	0.61	0.57
eFG %	0.48	0.45	0.46	0.51	0.52
FTARate	0.28	0.22	0.30	0.33	0.16
3PTRate	0.08	0.12	0.07	0.12	0.17
OR %	2.49	1.71	4.03	2.03	2.24
DR %	13.53	10.21	15.21	11.27	9.01
BLK %	1.57	0.57	0.49	0.67	0.58
TOV %	15.65	17.21	20.71	13.19	13.42
AST %	38.36	26.69	43.44	34.01	19.07
STL %	2.33	1.79	3.20	2.35	1.78
USG %	28.28	21.55	22.41	27.13	17.21
Jugador	K. Walker	D. Rose	R. Westbrook	S. Curry	P. Beverly

Tabla 6.1: Tabla de estadísticas medias de los clúster formados para los bases.

El jugador más representativo del clúster 0 es Kemba Walker, que es una súper estrella de la NBA. Observando la media de las estadísticas avanzadas de este clúster observamos que estos jugadores tienen una gran influencia en sus equipos, ya que finalizan un porcentaje elevado de posesiones ofensivas, como se observa en la estadística de uso (*USG %*). Observando las estadísticas ofensivas del raiting ofensivo (*OffRtg*) y el porcentaje floor (*Floor %*) podemos determinar que no son jugadores especialmente eficaces en el ataque, aunque anotan una cantidad decente de puntos. El porcentaje real (*TS %*) es un porcentaje bueno, aunque no destacable, y del porcentaje de eficiencia de tiros de campos (*eFG %*) podemos extraer que son jugadores que realizan en su mayoría tiros de dos puntos, sin realizar muchos triples.

En el clúster 1 se encuentran la mayor parte de los bases, y observamos que, según indica el uso, son jugadores con importancia en sus equipos. Observando las medias de cada estadísticas nos encontramos con que son los bases con menor raiting ofensivo medio, y un floor porcentaje bajo, esto indica que son jugadores ineficientes en ataque. El raiting defensivo (*DefRtg*) es aceptable, por lo que estos bases tienen unas capacidades defensivas

aceptables. Si miramos los porcentajes de tiro real y efectivo, nos damos cuenta que no son extremadamente altos pero son decentes. El jugador más representativo de este cluster es Derrick Rose.

El jugador que más sorprende en el clúster 2 es Russell Westbrook, que es uno de los mejores bases. Si vemos las estadísticas medias observamos que el raiting ofensivo y el floor porcentaje elevado, pero observando las estadísticas de los integrantes del clúster se observa que estas estadísticas están infladas debido a la influencia de Russel Westbrook. La estadística que parece ser más determinante en este clúster es el porcentaje de asistencias ($AST\,\%$), que es el más elevado de todos los clusters, por lo que estos jugadores destacan por la increíble capacidad de organizar y finalizar los ataques de sus equipos con una asistencia. Además esta estadística nos dice que realizan casi la mitad de las asistencias de sus equipos. También observamos que el raiting defensivo es bajo, por lo que estos jugadores son buenos defensores.

En el clúster 3 encontramos los bases que mejor rating ofensivo y floor porcentaje tienen, por lo que se encuentran agrupados a los bases con mejor capacidad de anotación. El uso en este clúster es bastante elevado, por lo que son jugadores con bastante importancia en sus equipos. La gran capacidad anotadora se refleja en los valores extremadamente elevados tanto del porcentaje real de tiro como el del porcentaje de tiro efectivo. El jugador que mejor representa a los jugadores de este clúster es Stephen Curry.

Al observar las estadísticas del clúster 4 nos damos cuenta de que las estadísticas ofensivas son similiares a las del clúster 3, la principal diferencia está en el uso, siendo el de este clúster mucho menor, diez puntos por debajo. Esto indica que son anotadores fiables, pero con un volumen mucho menor de canastas. Además el rating defensivo es bajo, y observando los jugadores que se encuentran en este clúster podemos identificar a estos jugadores como defensores con capacidad de anotación. El rating defensivo nos es mucho más bajo que el resto pero esto puede ser resultado de que estos jugadores defienden normalmente a los bases rivales con gran capacidad de anotación. El jugador destacado de este clúster es Patrick Beverly.

Para los escoltas hay que estudiar qué número de clúster es óptimo. Para ello seguimos usando el método de *Elbow Curve*. En este caso, el valor que parece ser óptimo para el número de clústers es 6. El modelo de K-Means para los escoltas también utilizará únicamente las estadísticas avanzadas para construir los 6 grupos de jugadores.

En el clúster 0 se han agrupado escoltas con un ratio de triple alto, por lo que relizan bastantes durante un partido. Podemos observar que, además, los porcentaje de tiro son elevados y el raiting ofensivo es alto también. El raiting defensivo también tiene un valor decente, por lo que podemos clasificar este clúster como defensores tiradores. El jugador más representativo de este clúster es Klay Thompson.

Las estadísticas del clúster 1 son similares a las del clúster 0, pero son por lo geneal peores. El uso de los escoltas de este clúster es elevado, por lo que estos jugadores tienen importancia en el ataque de sus equipos. El jugador más destacado del clúster 1 es CJ McCollum.

En el clúster 2 se encuentran escoltas que tienen una eficiencia pobre en ataque, a pesar de tener un uso alto. El raiting ofensivo es malo aunque los porcentajes de tiro no son del todo malos. El jugador más característico de este clúster es Kobe Bryant.

Clúter	0	1	2	3	4	5
Etiqueta	3&D	2nd Shooter	Ineff. Scorer	Eff. Scorer	Eff. Shooter	Defender
OffRtg	124.91	113.13	99.41	112.96	112.60	114.32
Floor %	0.37	0.36	0.30	0.40	0.37	0.40
DefRtg	107.43	109.36	111.74	107.03	103.92	102.90
NetRtg	17.48	3.77	-12.33	5.93	8.68	11.43
TS %	0.59	0.56	0.48	0.60	0.56	0.50
eFG %	0.56	0.50	0.44	0.47	0.49	0.45
FTARate	0.14	0.21	0.13	0.47	0.24	0.25
3PTRate	0.22	0.14	0.17	0.08	0.10	0.07
OR %	1.39	1.90	1.86	2.88	2.69	6.57
DR %	9.73	8.99	11.05	12.18	12.52	12.63
BLK %	0.80	0.50	1.01	1.03	1.02	1.46
TOV %	9.10	10.99	11.90	11.73	14.55	11.76
AST %	8.97	12.43	12.70	23.40	20.84	9.93
STL %	1.33	1.72	1.60	1.80	2.05	2.96
USG %	18.42	20.96	20.04	29.08	23.01	15.34
Jugador	K. Thompson	CJ. McCollum	K. Bryant	D. Wade	B. Beal	T. Allen

Tabla 6.2: Tabla de estadísticas medias de los clúster formados para los escoltas.

En el clúster 3 están los escoltas con un gran uso en sus equipos, es decir, posiblemente sean los jugadores con más importancia en sus respectivos equipos. El raiting ofensivo no es extremadamente alto, aunque el floor porcentaje y los porcentajes de tiro son elevados. Por lo general estos escoltas basan sus ataques en tiros de dos puntos. También observamos que el porcentaje de asistencias es elevado para tratarse de escoltas. El jugador destacable de este clúster es Dwayne Wade.

Los jugadores del clúster 4 son similares a los del clúster 3, pero con un porcentaje real de tiros mayor. El uso es elevado, sin ser extremadamente elevado, por lo que podría indicar que hay otro jugador importante en sus equipos. Quién mejor representa a este clúster es Bradly Beal.

En el clúster 5 han sido agrupados los escoltas que destacan en defensa, como se observa en su rating defensivo. Observando su raiting ofensivo y el porcentaje real de tiro se entiende que son jugadores capaces de anotar con consistencia. A pesar de esto su uso nos dice que no son jugadores que finalicen muchas jugadas en sus equipos, ya que su rol parece ser el de defensores. El jugador más destacable de este clúster es Tony Allen.

Para los aleros el valor que parece ser óptimo para el número de clúster es 5, que es donde la curva parece suavizarse. Al construirse el modelo de K-Means se utiliza 5 como valor para el número de clúster.

En el clúster 0 se encuentran aleros con un raiting ofensivo bajo y un porcentaje floor alto, esto indica que son jugadores que anotan en la mayoría de sus posesiones, por lo general, cerca de la canasta. El porcentaje real de tiros es elevado, mientras que el porcentaje de tiros efectivos es más bajo, lo que hacer referencia a que no realizan muchos tiros de tres puntos. Son aleros importantes en sus equipos, ya que tienen un uso alto. El jugador más destacado de este clúster es Andrew Wiggings.

Los aleros que se agrupan en el clúster 1 son jugadores que ofensivamente son muy eficientes, como se ve en el raiting ofensivo y el porcentaje floor, que son elevados. El porcentaje real de tiro es excepcionalmente alto y el porcentaje de tiro eficiente también es alto.

Clúter	0	1	2	3	4
Etiqueta	Athletic	2 Side Dominator	Ineff. Scorer	Eff. Scorer	3PT Shooter
OffRtg	105.90	119.70	101. I 70	117.49	119.40
Floor %	0.43	0.41	0.32	0.39	0.34
DefRtg	109.57	100.44	107.39	107.40	107.21
NetRtg	-3.67	19.26	-5.69	10.08	12.18
TS %	0.59	0.61	0.51	0.55	0.56
eFG %	0.49	0.53	0.46	0.50	0.53
FTARate	0.44	0.32	0.19	0.21	0.15
3PTRate	0.05	0.12	0.14	0.11	0.21
OR %	5.08	3.27	2.92	4.80	2.02
DR %	15.28	17.77	14.98	14.85	11.15
BLK %	1.44	1.60	0.75	1.20	1.19
TOV %	11.96	12.82	14.34	11.18	9.92
AST %	9.23	17.61	9.34	8.73	7.56
STL %	1.10	2.30	1.56	1.88	1.84
USG %	24.25	23.52	17.66	17.39	16.34
Jugador	A. Wiggins	L. James	L. Stephenson	R. Gay	W. Matthews

Tabla 6.3: Tabla de estadísticas medias de los clúster formados para los Aleros.

Además el raiting defensivo es bajo, lo que indica que estos aleros son jugadores con gran habilidad en ambos lados de la cancha. El jugador que destaca en este clúster es LeBron James, considerado por muchos como el mejor jugador del planeta.

Los aleros del clúster 2 no destacan en el ataque, ya que las estadísticas relacionadas con este aspecto son bajas. Por lo que se observa en el raiting defensivo, no son jugadores que destaquen en defensa. Podemos determinar que estos jugadores son atacantes ineficientes, y el jugador que mejor resume estas estadísticas es Lance Stephenson.

En el clúster 3 encontramos aleros con un alto raiting ofensivo y un porcentaje floor elevados, y poseen buenos porcentajes de tiro. No son excepcionales en el ataque, aunque son anotadores fiables. En el clúster 3 el jugador que destaca por encima del resto es Rudy Gay.

Para el clúster 4 se han agrupado aleros con un alto raiting ofensivo, pero bajo porcentaje floor. Esto pude deberse a que cumplen el rol de tiradores en sus respectivos equipos, y como se observa en el uso, son jugadores de rol, sin más importancia. El jugador destacado de este clúster es Wesley Matthews.

Al realizare el estudio sobre los ala pívots podemos determinar que el punto en el que se suaviza la curva es para el valor de 6 clústers. La curva se observa que no es muy suave, y tiene muchos picos, a pesar de esto, 6 clústers es un número coherente con la curva y con los posibles tipos de jugadores.

En el clúster 0 encontramos ala pívots que rinden bien en ambos lados de la cancha, como podemos ver en los ratings ofensivo y defensivo. También tienen unos porcentajes de tiro real y de tiros efectivos elevado. Estos ala pívots tienen poca importancia, como se ve en el uso medio. El jugador más representativo es Kenneth Faried.

Los ala pívots agrupados en el clúster 1 son grandes anotadores, a pesar de que el rating ofensivo no es muy alto el porcentaje floor si que es elevado. Son jugadores con un uso elevado, por lo que una parte considerable de la ofensiva de sus equipos recae en ellos. El

Clúter	0	1	2	3	4	5
Etiqueta	2 Side	1er Scorer	Open 4	2 Side Dominator	Role Player	Eff. Shooter
OffRtg	129.77	110.75	123.77	122.06	111.42	107.02
Floor %	0.51	0.41	0.38	0.45	0.41	0.39
DefRtg	102.94	100.35	105.12	98.70	106.02	109.36
NetRtg	26.94	10.40	18.66	23.36	5.40	-2.34
TS %	0.63	0.54	0.59	0.59	0.51	0.55
eFG %	0.55	0.48	0.54	0.52	0.46	0.47
FTARate	0.41	0.25	0.19	0.29	0.23	0.29
3PTRate	0.02	0.07	0.16	0.03	0.5	0.09
OR %	11.99	8.48	4.59	7.66	9.38	6.64
DR %	21.26	22.52	17.19	22.76	17.35	21.08
BLK %	3.70	2.78	1.74	2.79	1.40	1.99
TOV %	13.03	11.89	11.46	13.85	15.80	10.90
AST %	8.23	10.77	8.52	19.28	6.76	9.27
STL %	1.16	1.73	1.25	1.90	1.33	1.14
USG %	15.21	22.13	17.63	21.61	12.95	24.03
Jugador	K. Faried	L. Aldridge	M. Teletovic	P. Gasol	T. Booker	D. Nowitzki

Tabla 6.4: Tabla de estadísticas medias de los clúster formados para los ala pívots.

jugador más destacable de este clúster es LaMarcus Aldridge.

En el clúser 2 encontramos ala pívots con estadísticas ofensivas similares a los del clúster 0, aunque son un poco peores en defensa. La gran diferencia se nota en el ratio de triples, ya que son jugadores con un ratio elevado y que basan parte de su anotación en triples. El jugador que podemos destacar de este clúster es Mirza Teletovic.

En el clúster 3 encontramos a ala pívots muy dominantes en ambos lados de la cancha. En la parte defensiva destaca su rating defensivo excepcionalmente bajo y un elevado porcentaje de rebotes defensivos. En en lado ofensivo poseen muy buenos porcentajes de tiro y un rating ofensivo y porcentaje floor muy elevados. Debido a esto son ala pívots con un uso alto en sus equipos. Como jugador destacado de este clúster podemos nombrar a Pau Gasol.

En el clúster 4 se encuentran ala pívots con un uso muy bajo. Estos jugadores tienen un raiting ofensivo decente y buenos porcentajes de tiro. Están dentro de las rotaciones de sus equipos pero no tienen apenas importancia dentro de ella. Un jugador representativo del clúster puede ser Trevor Booker.

El clúster 5 es similar al anterior, pero los ala pívots que lo forman tienen mayor uso en sus equipos, y en consecuencia, mayor importancia en la rotación. Los porcentajes de tiro no son malos, aunque el raiting ofensivo es bajo. De este clúster podemos destacar a Dirk Nowitzki.

Por último, hay que realizar la clusterización para los pívots. Al realizar el método *Elbow Curve* obtenemos que en este caso también encontramos el valor óptimo para el número de clústers es 5, en este punto encontramos un equilibro entre la suavización de la curva y un número razonable de grupos de jugadores. Las estadísticas medias de cada clúster se ven en la tabla de abajo.

En el clúster 0 se han agrupado a pívots con poco uso, es decir, que no finalizan muchos ataques. A pesar de esto las estadísticas ofensivas de estos pívots no son malas, y son

Clúter	0	1	2	3	4
Etiqueta	Role Defender	2 Way	2 Side Dominator	Open 5	Defender
OffRtg	109.27	119.58	131.52	115.15	110.21
Floor %	0.44	0.45	0.55	0.36	0.42
DefRtg	105.22	103.75	99.96	109.106	108.56
NetRtg	4.05	15.82	31.56	5.98	1.65
TS %	0.58	0.59	0.66	0.55	0.56
eFG %	0.48	0.52	0.60	0.49	0.48
FTARate	0.49	0.25	0.56	0.20	0.32
3PTRate	0.00	0.01	0.00	0.15	0.01
OR %	11.10	10.26	13.08	4.87	7.51
DR %	24.87	21.07	25.86	19.98	22.84
BLK %	3.18	3.59	4.72	1.90	3.49
TOV %	19.63	13.03	13.88	12.48	10.26
AST %	10.78	9.86	5.99	14.08	12.52
STL %	1.59	1.28	1.31	1.19	1.37
USG %	16.29	19.89	16.72	20.13	29.31
Jugador	J. Noah	T. Duncan	R. Gobert	S. Hawes	M. Gasol

Tabla 6.5: Tabla de estadísticas medias de los clúster formados para los Aleros.

capaces de anotar con buenos porcentajes. Si observamos el rating defensivo podemos determinar que son jugadores de rol en el equipo cuya principal tarea es defender. De este clúster podemos destacar a Joakim Noah.

Los pívots que pertenecen al clúster 1 son importantes en sus equipos, como se ve en el uso medio del clúster. En la parte ofensiva encontramos que tienen un buen rating ofensivo con un floor porcentaje bueno, esto indica que son capaces de anotar de manera constante a pesar de no contar con un elevado número de tiros. El porcentaje real de tiros es bastante elevado, lo que refuerza la idea de que son anotadores de confianza. El rating defensivo es bajo, por lo que también son buenos defensores aunque no destaquen por ello. Con esta información podemos concluir que son jugadores con importancia en sus equipos y que son fiables tanto en ataque como en defensa. De este clúster podemos destacar a Tim Duncan.

En el clúster 2 han sido agrupados los pívots más dominantes de la NBA. Como se observa su rating ofensivo es muy elevado aunque puede ser consecuencia de que, como se observa en el porcentaje real de tiros y el porcentaje de tiros de campo efectivos, anotan la mayoría de sus tiros. Si observamos también su porcentaje floor podemos identificar a estos pívots como finalizadores en ataque, es decir, anotan la mayoría de puntos debajo del aro. En el lado defensivo también son capaces de dominar, como se en su rating defensivo que es uno de los más bajos, y en el porcentaje de tapones que realizan. El uso de estos jugadores no es elevado, por lo que no son la primera opción en ataque. De este clúster podemos destacar a Rudy Gobert.

El clúster 3 es quizás el más especial, ya que se ha formado con pívots que son capaces de de abrir la cancha en ataque y anotar desde la línea de tres puntos. Esto se observa en el ratio de triples intentados, donde vemos que de media el 15 % de sus tiros son triples. A pesar de tirar de tres, el porcentaje real de tiro y el porcentaje de eficiencia de tiros

de campo es relativamente alto. El uso de estos pívots es un poco más elevado y esto seguramente sea debido a que finalizan más ataques gracias a realizar un triple desde una buena posición. De este clúster podemos destacar a Spencer Hawes.

En el clúster 4 nos encontramos a pívots con mucho peso dentro del equipo. Su uso es el más elevado de entre los pívots, lo que indica que finalizan muchas de las jugadas de sus equipos. Ni el rating ofensivo ni el porcentaje floor son extremadamente altos, pero esto se puede deber a tener mayor volumen de tiro lo que hace que en consecuencia que fallen más tiros que el resto. El rating defensivo parece no haber tenido mucha importancia en este clúster, ya que al ver los valores de cada jugador encontramos datos muy variados. El porcentaje real de tiro es bueno y nos indican que son jugadores, que a pesar de tener un volumen alto de tiros, son capaces de anotar de manera consistente. El jugador destacable de este clúster es Marc Gasol.

Con los clúster por posición se pretende crear información defensiva sobre los jugadores. Para ello se analizan los ficheros de posición y para cada tiro realizado en cada partido se extrae el tirador y el defensor y si el tiro ha sido anotado o no. También se añade la información sobre el clúster y posición del defensor y del tirador. Con esta información se puede obtener a qué cluster defiende mejor cada jugador y cada clúster en general, brindando apoyo útil al entrenador.

CAPÍTULO 7

Diseño de Interfaces de Usuario

Las interfaces de usuario son el medio por el cual un usuario, el cuerpo técnico de un equipo de baloncesto en este caso, se comunica con el sistema de información. Es importante seleccionar de manera adecuada qué información se muestra, ya que esta debe ser adecuada para describir la forma de jugar de un equipo.

A lo largo de este capítulo se mostrarán los diseños de las distintas pantallas con las que tendrá que interactuar el usuario. Para las gráficas que se muestran se ofrecerá un explicación de la información que aportan al entrenador. También se pretende explicar como debería de ser el flujo de la aplicación.

Lo primero que tiene que hacer un usuario al abrir la aplicación es seleccionar su equipo. Para ello se muestran todos los equipos dividido por conferencia, de manera que cada conferencia está en una pestaña.

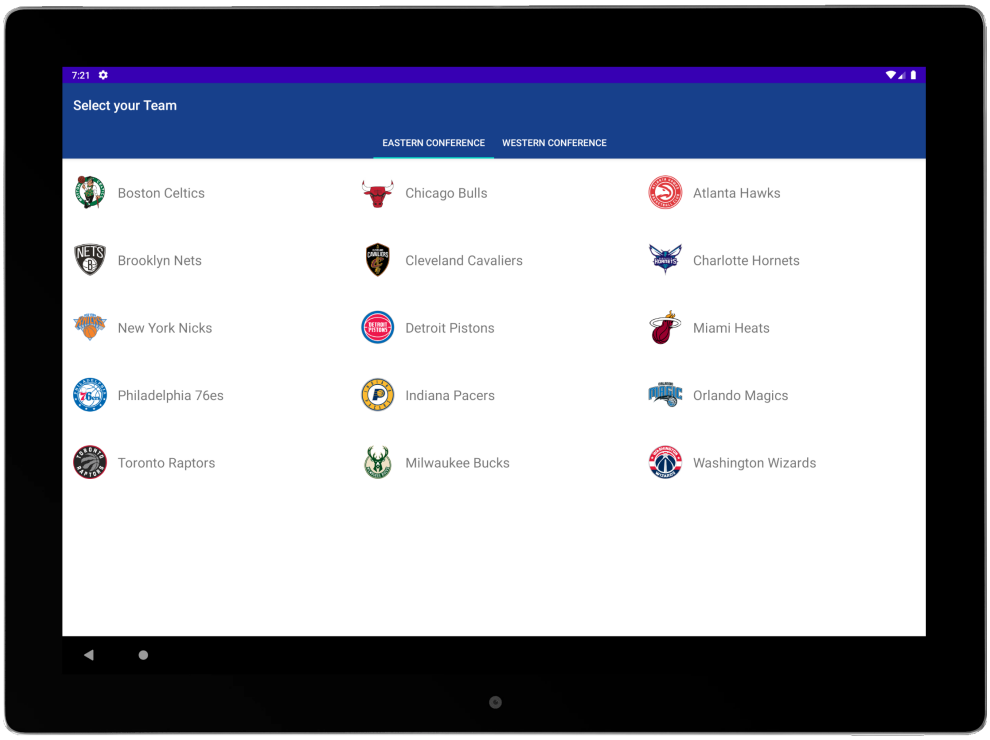


Figura 7.1: Interfaz para la selección de equipos de la conferencia este.

Como se observa en la figura 7.1 hay una pestaña que muestra los equipos de la conferencia este y otra los de la conferencia oeste, si se toca la pestaña "Western Conference" se cargan el resto de equipos.

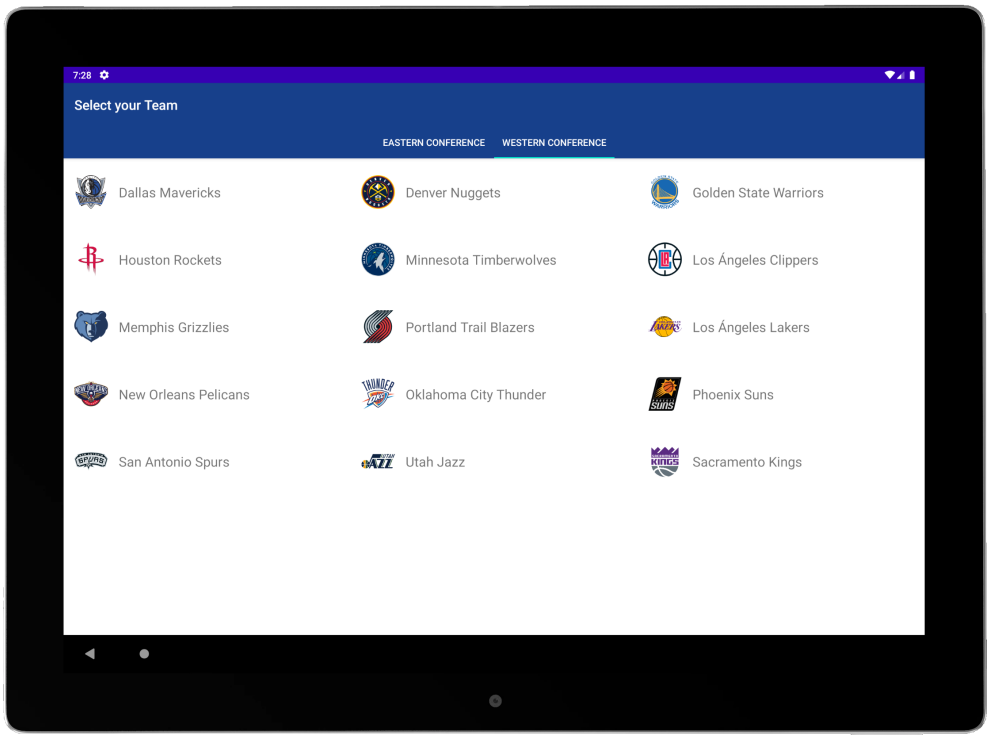


Figura 7.2: Interfaz para la selección de equipos de la conferencia este.

Dentro de cada vista los equipos se organizan de manera que en cada columna están los equipos de la misma división, ya que en la NBA hay tres divisiones por conferencia y cinco equipos por división.

Una vez se seleccione un equipo se cargará la página de vista previa del equipo. Para acceder a la información de los demás equipos no seleccionados anteriormente se debe desplegar el menú lateral y acceder a la opción "TEAMS", que mostrará las mismas pantallas que se ven en las figuras 7.1 y 7.2.

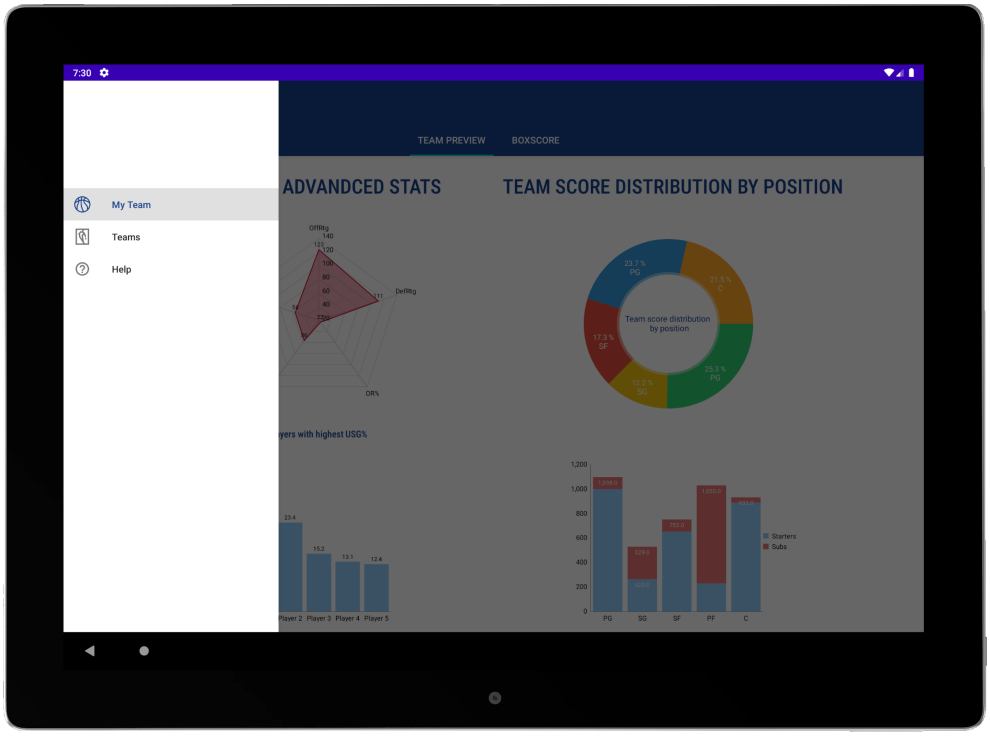


Figura 7.3: Interfaz con menú desplegado.

La información que se ofrece en la vista previa de los equipos se ofrece a modo de resumen de las estadísticas más representativas de los equipos y que permiten identificar de manera clara el estilo de juego que tienen.

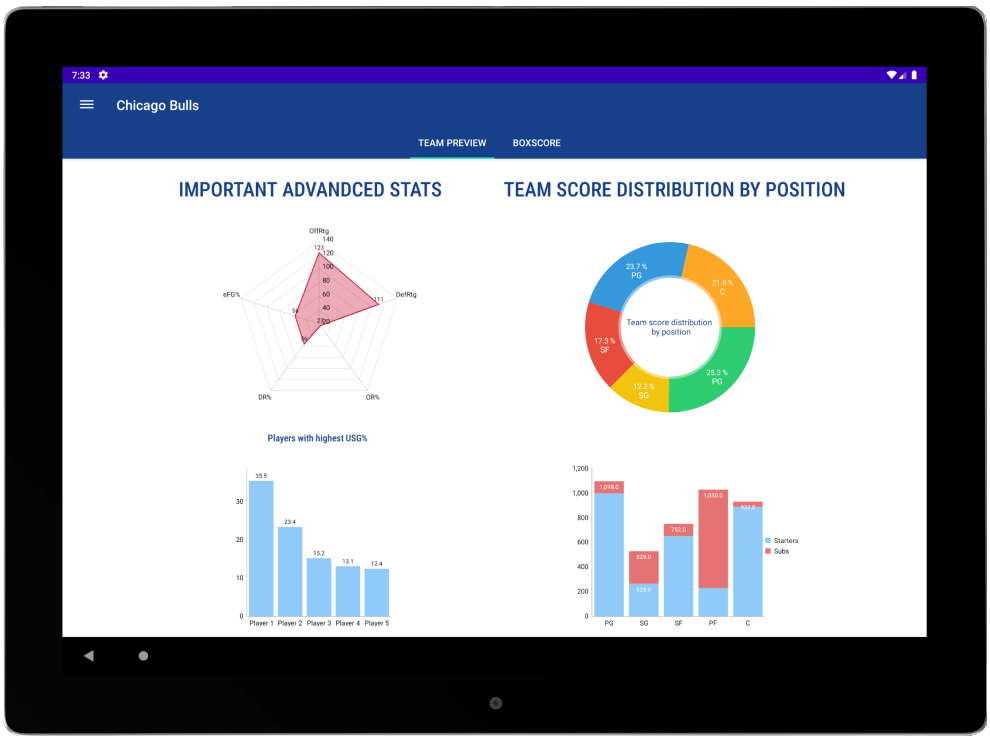


Figura 7.4: Interfaz para la selección de equipos de la conferencia este.

En la parte superior izquierda encontramos un gráfico de radar en el que se muestran

estadísticas avanzadas que permiten evaluar la actuación de un equipo durante los partidos. Las estadísticas que se incluyen son el rating ofensivo y defensivo, el porcentaje de rebotes defensivos y ofensivos y el porcentaje de tiros de campo eficientes. Con estos datos podemos hacernos una idea preliminar bastante acertada sobre las capacidades ofensivas y defensivas del equipo.

Justo debajo se encuentra un gráfico de barras que indica el porcentaje de uso del quinteto titular. Esta estadística indica el porcentaje de jugadas ofensivas que acaba el jugador. Mediante este gráfico se pretende ofrecer una visión de qué jugador tiene más peso dentro del ataque del equipo. Esto permite centrar la defensa sobre el jugador que más volumen ofensivo acapara, centrando el estudio del rival en dicho jugador.

En la parte izquierda se incluyen dos gráficas que hablan sobre la distribución de la anotación del equipo en función de las posiciones de los jugadores. Esto permite conocer la efectividad de cada una de las posiciones y averiguar a qué posición da más importancia cada equipo en la parte ofensiva.

En la parte superior derecha se muestra un gráfico circular en el que se incluye por cada posición el porcentaje de los puntos anotados por el equipo. En esta gráfica se observa qué posición es más anotadora en cada equipo y sobre la que se tiene que tener mayor presencia defensiva.

Justo debajo del gráfico circular hay un gráfico de barras agrupadas que muestra la anotación por por posición de los titulares y de los suplentes. Normalmente los equipos suelen utilizar entre tres y cuatro suplentes, por lo que normalmente los titulares suelen anotar más puntos. A pesar de esto, pueden encontrarse excepciones en las que el equipo suplente sea más efectivo, como el equipo de Los Ángeles Clippers de la temporada 2018/2019. En este gráfico se pretende mostrar qué jugadores anotan más por posición dentro del equipo, y si esto se debe a un jugador titular excepcionalmente anotador o a suplentes con buena habilidad anotadora.

Si se pulsa sobre la pestaña “INFORM” se mostrará un informe en el que se enfrentan las estadísticas avanzadas del equipo seleccionado y del equipo del usuario. También se muestra el resultado de la predicción realizada mediante aprendizaje automático.

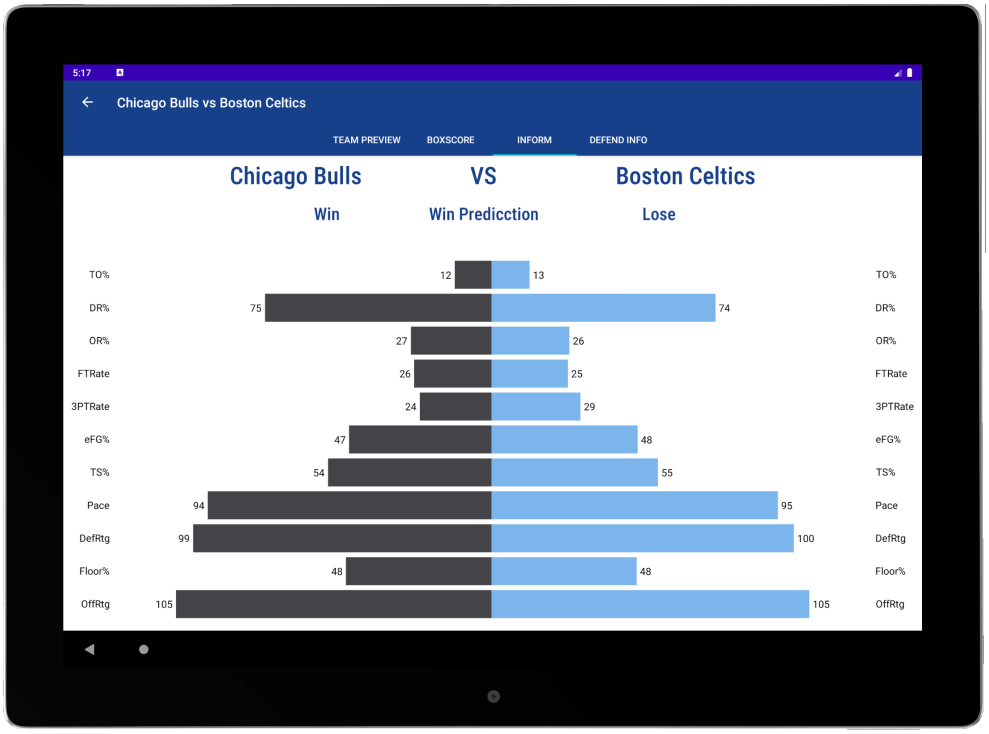


Figura 7.5: Interfaz que muestra el informe para el partido contra el equipo seleccionado.

En esta pantalla se podrá deslizar la pantalla para ver el resto de estadísticas avanzadas. Se muestran de esta manera para contraponer los puntos fuertes y débiles de ambos equipos. De esta manera si, por ejemplo, el equipo rival tiene un porcentaje de rebotes ofensivos mayor que el porcentaje de rebotes defensivos de nuestro equipo podamos identificar la necesidad de cargar la zona de rebotes defensivos.

En la pestaña “DEFEND INFO” que aparece cuando se pulsa sobre uno de los equipos rivales se muestra a la izquierda es el quinteto titular de dicho equipo y a la derecha el quinteto del equipo del usuario que mayor eficacia tiene defendiendo al quinteto contrario.

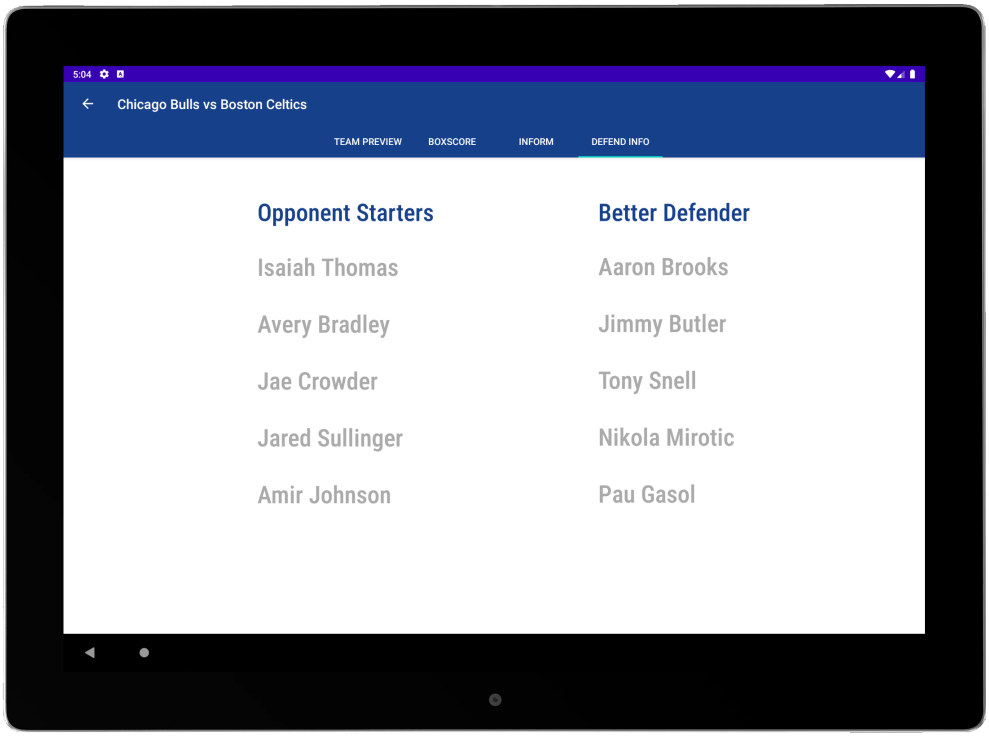


Figura 7.6: Interfaz que muestra el mejor quinteto para defender al quinteto rival.

Otra pestaña para mostrar la información del equipo es "BOXSCORE". En esta pestaña se muestra en forma de tabla todas las estadísticas sobre el equipo y sus jugadores. Se puede realizar scroll tanto vertical como horizontal para poder visualizar toda la información que se muestra en la tabla.

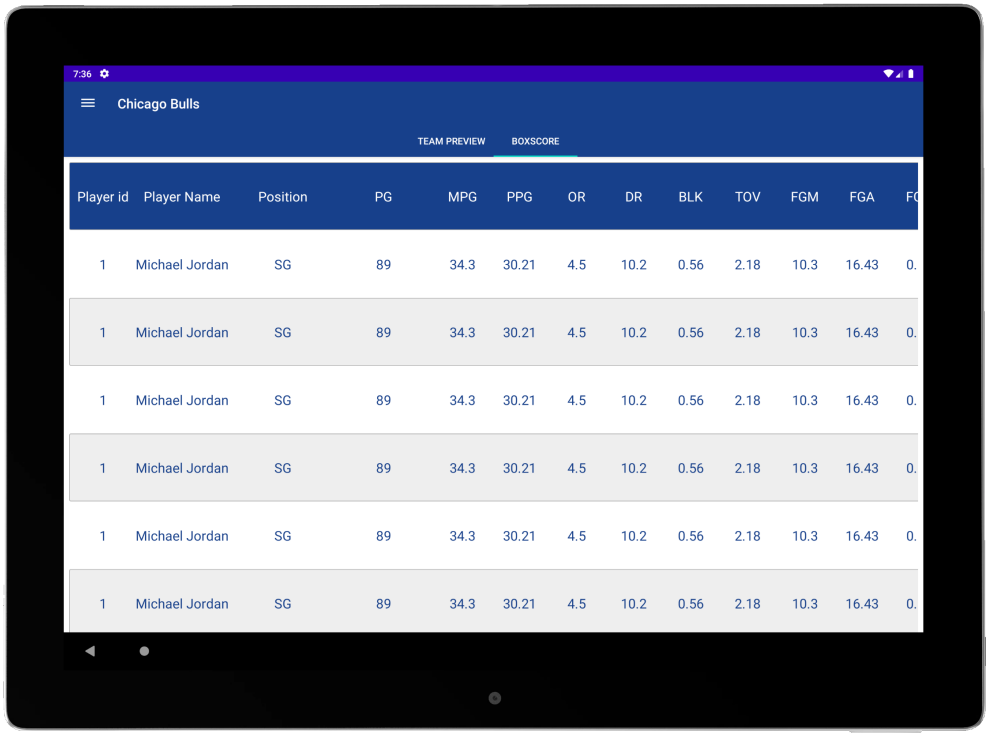


Figura 7.7: Interfaz que muestra una tabla con las estadísticas del equipo y de sus jugadores.

Se podrá realizar un desplazamiento horizontal para ver el resto de estadísticas, de manera que el nombre del jugador se quedará siempre visible. Si se pulsa sobre alguna de las filas de los jugadores se abrirá la vista en la que se muestran los detalles relevantes de cada jugador.

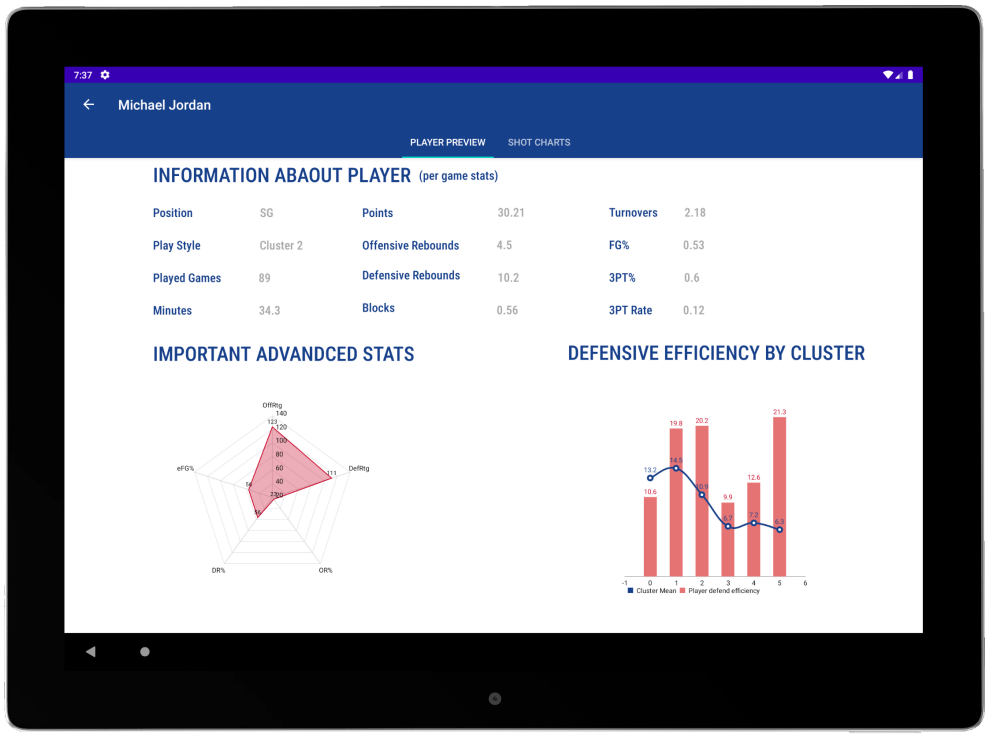


Figura 7.8: Vista resumen de los detalles de un jugador.

La manera en la que se muestra la información de los jugadores está organizada de la misma manera que las vistas para los equipos. Se separa en dos pestañas, de modo que en una de ellas se muestra información relevante a modo de resumen y la otra se muestran gráficas de tiro para analizar los puntos fuertes del jugador.

En la pestaña "PLAYER PREVIEW" se muestra la información a modo de resumen. En la mitad superior se muestran datos relevantes y estadísticas por encuentro del jugador. También se muestran algunas estadísticas avanzadas que permiten entender cómo es la actuación del jugador en los partidos.

En la parte inferior izquierda se encuentra un gráfico de radar como el que se utiliza para los equipos y que produce el mismo efecto de calificación sobre el jugador. En la parte inferior derecha se incluye un gráfico combinando un gráfico de barras y un gráfico lineal.El gráfico de barras muestra el total de defensas exitosas que el jugador ha realizado sobre los diferentes clústers que hay para su posición, mientras que el gráfico lineal muestra la media de defensas exitosas de su clúster sobre los demás clústeres. Con esta información podremos saber cómo de bueno es un jugador en defensa para los distintos clústers que hay dentro de su posición, permitiendo decidir al entrenador a qué tipo de jugador tiene que defender.

Si pulsamos sobre la pestaña "SHOT CHART" se abrirá una vista con dos gráficas sobre los tiros del jugador.

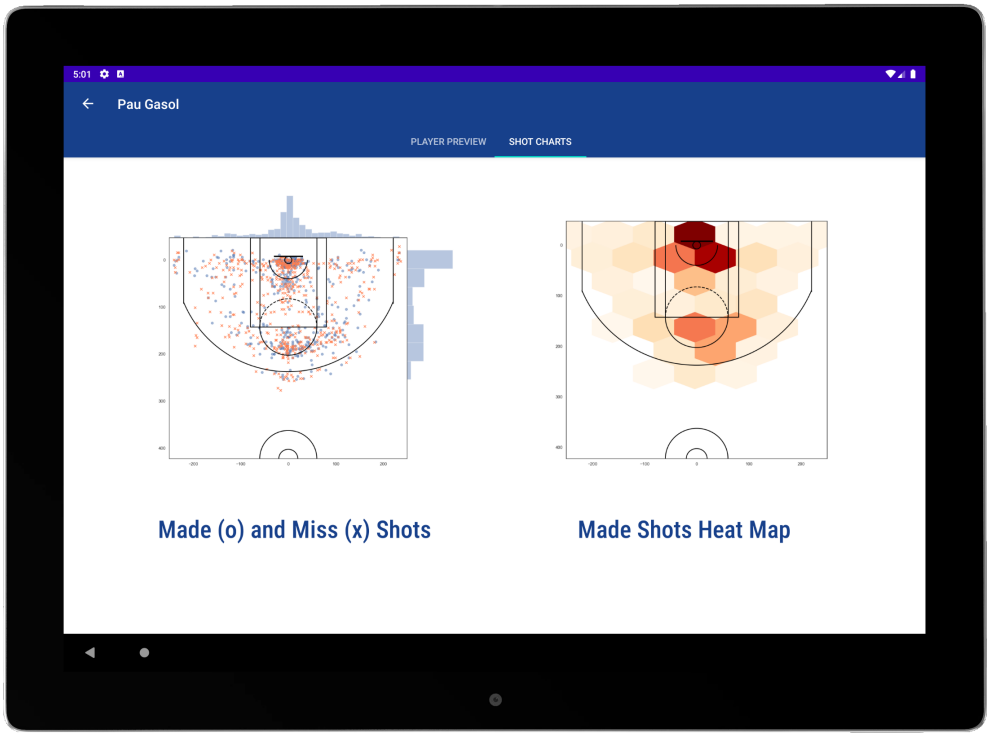


Figura 7.9: Vista que incluye las gráficas de tiro.

La gráfica que aparece a la izquierda es un gráfico de puntos, dónde cada punto representa un tiro realizado por el jugador. En el eje x se muestra además un gráfico de barras que muestra en que valor de la coordenada x se realizan más tiros, lo mismo ocurre con el eje y. Este gráfico nos brinda la información necesaria para saber en qué zonas de la cancha el jugador ejecuta una mayor densidad de tiros a canasta. Con esta información podemos

extraer las zonas más cómodas para el jugador, permitiendo así personalizar la defensa sobre un lugar adecuado.

A la derecha se encuentra un gráfico hexbin. Cada región hexagonal posee un color determinado de tal manera que cuanto más oscuro sea mayor porcentaje de acierto tiene el jugador en esa zona. De aquí se puede extraer la información sobre un concepto bastante utilizado en el baloncesto moderno, las zonas calientes del jugador. Estas zonas calientes hacen referencia a las partes de la pista dónde un jugador es capaz de anotar un mayor porcentaje de canastas, no necesariamente coincide con las zonas donde más tiros realiza.

También se incluye un apartado llamado "HELP" en el que se proporciona información relevante de la aplicación. En esta parte se aporta información sobre cómo funciona la predicción de partidos y la clusterización de los jugadores. También se incluye información

CAPÍTULO 8

Implementación

En este capítulo se explica cómo ha sido el proceso de implementación de las dos partes que conforman este sistema de información. Para el desarrollo de la API es necesario la instalación de Python 3.7, que es lenguaje seleccionado para el desarrollo, y se utilizará el IDE Pycharm. El desarrollo de la aplicación móvil que consume la API se debe instalar el el kit de software de desarrollo de Android y el IDE Android Studio, también hay que instalar el paquete que contiene el lenguaje de Kotlin.

Para manejar los entornos virtuales de Python, que permiten que cada proyecto tenga sus propias dependencias evitando así problemas de incompatibilidades, se utiliza Anaconda. Para el proyecto de la API se crea un entorno virtual en el que serán instaladas todas las dependencias necesarias.

8.1. Desarrollo de la API

Para el desarrollo de la API se seleccionó el paquete Django REST Framework, que debe ser instalado. Para empezar el desarrollo lo primero que debe hacerse es crear un proyecto en Pycharm usando la opción que ofrece de crear un proyecto Django, como se observa en la figura 8.1. Una vez se crea el proyecto se generan los ficheros base (figura 8.2), los más importantes son el fichero *settings.py* y el fichero *urls.py*. En el fichero *urls.py* se incluyen todas las rutas disponibles para comunicarse con la API.

En el fichero *settings.py* se incluye la información sobre la configuración del proyecto. Los valores de configuración más importantes son:

- ***INSTALLED_APPS***: es una lista que incluye el nombre de todas las *apps* que utilizará el proyecto. En el entorno de trabajo Django una *app* es un módulo que se encarga de realizar una función específica. En esta lista se incluyen las *apps* creadas para los microservicios y la *app* que contiene la lógica del paquete de extensión Django REST Framework, que tiene el nombre de *rest-framework*.
- ***DATABASES***: lista que contiene un diccionario por cada base de datos conectada al proyecto. Para este proyecto solo se utiliza una base de datos cuyo motor es MySQL. También se debe añadir la información sobre el usuario y contraseña para el acceso a la base de datos.

- ***ALLOWED_HOSTS***: en esta lista se añaden todos los host desde los que se permiten conexiones.

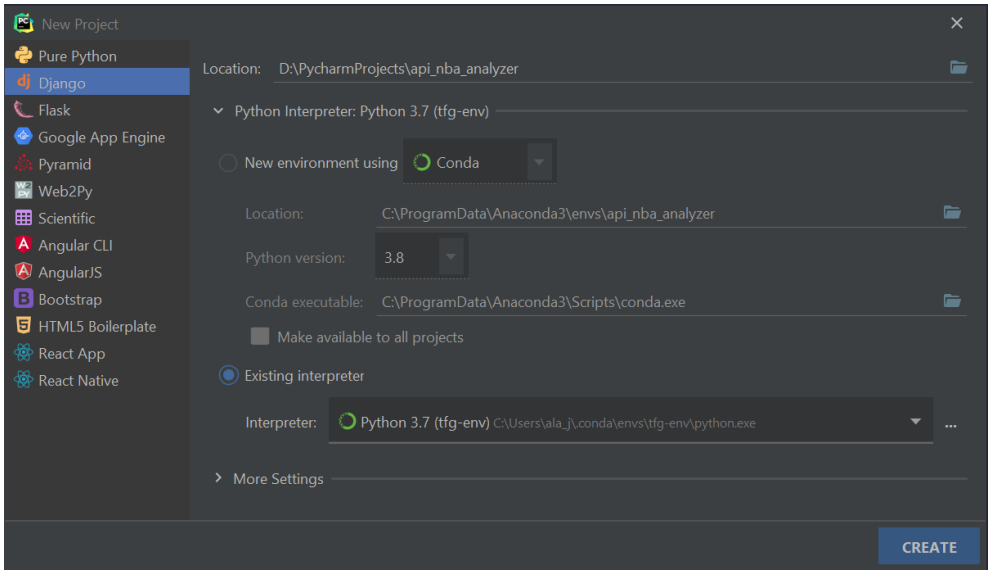


Figura 8.1: Pantalla de creación de proyecto Django.

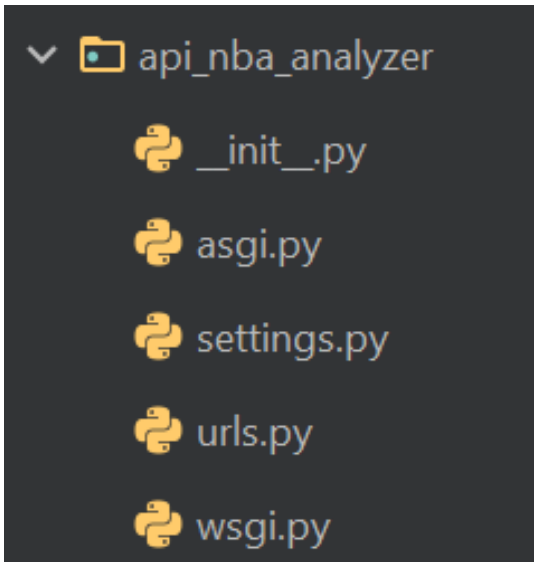


Figura 8.2: Estructura del proyecto Django.

Para la implementación de cada microservicio se crea una *app* de Django. Para crear una *app* es necesario utilizar un comando en el servidor de Django. Los comandos que proporciona Django se encuentran dentro de un fichero que recibe el nombre de *manage.py*, en el que se incluyen funciones para la administración del proyecto. Para crear una aplicación utilizamos el comando: *python manage.py startapp name*.

Una vez que se han creado las *apps* para cada microservicio se incluyen los canales de comunicación (***endpoint***) que corresponden a cada microservicio. Para visualizar los métodos disponibles para microservicio se utiliza el paquete *Swagger*, que es un paquete que genera documentación sobre la API de manera automática. Al acceder a la página principal de la API se vería:

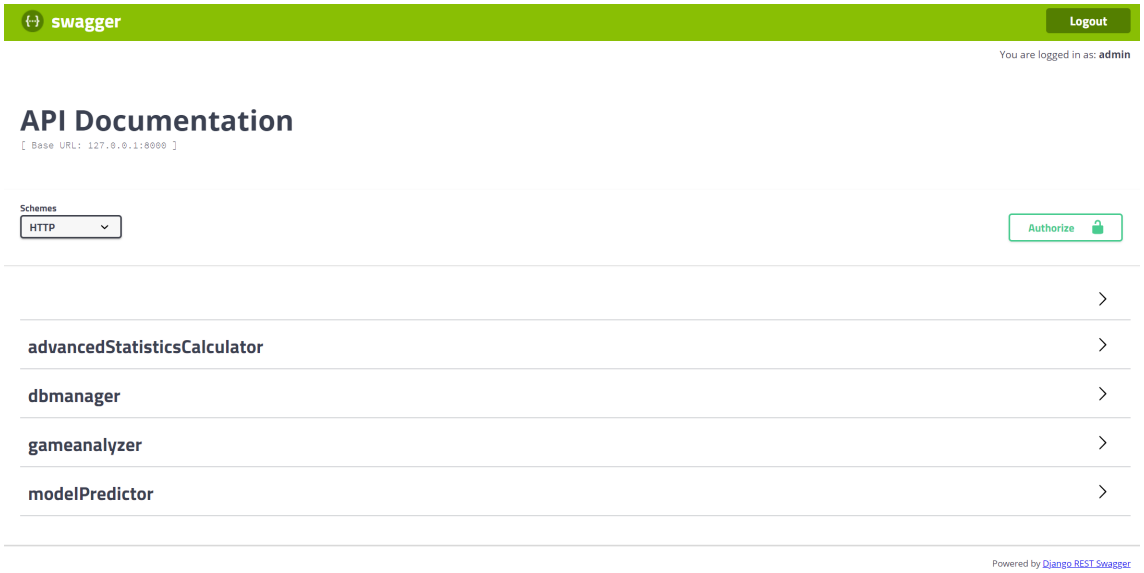


Figura 8.3: Página principal de la API.

8.1.1. Gestor de base de datos

La *app* creada para contener la lógica del microservicio recibe el nombre de *dataBaseManager*. Este microservicio será el único capaz de comunicarse con la base de datos, así que los modelos son creados dentro de esta *app*.

Los modelos creados se incluyen dentro del fichero *models.py*. Los modelos son creados mediante una clase en Python, que hereda de la clase *Model* del paquete Django. Para definir las columnas del modelo se declaran atributos dentro de la clase que toman como valor clases proporcionados por Django. Al crear el modelo para almacenar la información de los equipos nos quedaría una clase similar a:

```
class Team(models.Model):
    team_id = models.IntegerField(primary_key=True, null=False)
    name = models.CharField(max_length=255, null=False)
    abbreviation = models.CharField(max_length=255, null=False)
```

Los modelos que se han creado son:

- ***Team***: almacena la información de los equipos. Para almacenar la información general, como nombre, abreviación, etc. se almacena en campos de tipo char (*CharField*). El total de las estadísticas simples se almacenan en campos numéricos y las estadísticas por partido se almacenan en campos decimales (*DecimalField*).
- ***Player***: en este modelo se almacena la información de los jugadores. Los campos de la tabla son similares a los del modelo de los equipos, ya que gran parte de la información que almacenan es similar.
- ***Game***: almacena información sobre los partidos. El partido tiene un identificador numérico y un campo booleano para saber si la información del partido ha sido cargada en la base datos. También se incluyen tres campos que almacenan el identificador de un equipo y permiten identificar al equipo local, al visitante y al ganador del encuentro

- ***PlayerShotCharts*** este modelo permite almacenar las gráficas de tiro de los jugadores. Este modelo tiene una clave primaria, que coincide con la de los jugadores, y dos campos de imagen (*ImageField*) para almacenar las gráficas como archivos png.

Para cada modelo es necesario un serializador que sea capaz de transformar la información de la base de datos a formato JSON y viceversa. Estos serializadores se incluyen en el fichero *serializers.py*. Los serializadores se declaran de la siguiente forma:

```
class TeamSerializer(serializers.ModelSerializer):
    class Meta:
        model = Team
        fields = '__all__'
```

Para crear el servidor se crea una clase que hereda de *ModelSerializer*, que es una clase que se encuentra en el paquete de Django. Dentro de esta clase se encuentra otra clase con el nombre *Meta* en el que se añade la metainformación sobre a qué modelo se serializa y qué campos de este modelo se incluyen. Al heredar de la clase *ModelSerializer* se deben crear métodos que permiten crear y modificar registros de la base de datos.

El método de creación es el mismo para todos los serializadores, se crea un nuevo registro en la base de datos con la información que se almacena en la instacia del serializador. Para la creación de un registro en el modelo de equipo el método sería:

```
def create(self, validated_data):
        return Team.objects.create(**validated_data)
```

Para el resto de modelos es igual, cambiando la clase por la correspondiente a cada modelo. El método de actualización recibe los mismos parámetros más uno extra que es *instance*, el que se encuentran los valores actuales para el registro que se desea modificar. El método desarrollado para la actualización de un equipo y el que actualiza la información de un jugador es igual. En ambos casos la lógica añadida no permite la modificación de todos los campos, solo se pueden modificar las estadísticas simples. Hay que tener en cuenta que la información de un equipo o un jugador será modificada una vez que se hayan analizado los ficheros de un partido, y se actualizarán las estadísticas simples. En el método *update* de los serializadores del equipo y el jugador se suman los valores almacenados en el registros a los que se pasan a través del parámetro *validated_data*.

En el serializador del modelo de los partidos solo se actualiza el campo que indica si la información del partido ha sido analizada o no.

Los puntos de conexión con este microservicio se definen en un fichero con el nombre *views.py*. Para cada punto de conexión se crea una clase que herede de *ApiView*, que es una clase que se encuentra en el módulo Django Rest Framework. Dentro de la clase se define un método stático con el nombre del tipo de petición HTTP (get, put o post) que utilizará el punto de conexión. Dentro de este método se incluye la lógica necesaria para cada punto de acceso.

En ester fichero se ha añadido la lógica suficiente que permite la realización de las operaciones de creación, actualización y lectura de los modelos existentes. También se han añadido métodos que permiten listar partidos en función del equipo local, visitante o ganador. Otro método que se incluye permite obtener la información defensiva de un jugador respecto a los clúster que existen en su posición. Esta información se obtiene a partir de un

fichero CSV que ha sido generado según se comenta en la sección 6.4. Se ha desarrollo un método que permite extraer para un equipo el mejor quinteto para defender a un equipo dado siguiendo la lógica del método comentado anteriormente.

Una vez tenemos implementados los métodos necesarios para los puntos de conexión es necesario incluir las rutas para cada uno en el fichero *urls.py* de la *app*. Para incluir una ruta se tiene que proporcionar una URL y la clase que alberga el método. Por ejemplo, para añadir la ruta para obtener a un equipo por su identificador es necesario añadir:

```
path('team/<int:team_id>', GetTeamById.as_view(),
        name='get-team-by-id')
```

En esta ruta se ha añadido como parámetro en la propia ruta el identificador del equipo. Todas las rutas de este microservicio se añaden en una lista y son enrutadas desde el fichero *urls.py* del proyecto Django.

8.1.2. Análisis de partidos.

Para el microservicio de análisis de partidos se crea una *app* llamada *gameAnalyzer*. En esta *app* solo se encuentra un punto de conexión que permite el análisis de los ficheros de un partido. Los ficheros deben ser subidos con anterioridad al servidor para que puedan ser analizados.

Cuando se realiza una llamada al punto de acceso en primer lugar se realiza una comprobación del registro del partido en la base de datos, se comprueba si el partido ha sido ya analizado mediante el campo *information_loaded*. Si no ha sido analizado el próximo paso es recuperar los registros de los equipos y los jugadores involucrados en el partido. Al tener la API dividida en microservicios, no se puede acceder a la base de datos directamente desde este punto de acceso. Para obtener esta información se realiza una petición al punto de acceso que responde con los jugadores de un equipo, la petición se realiza utilizando el paquete ***requests***:

```
url = 'http://127.0.0.1:8000/dbmanager/players-from-team/'
home_players = requests.get(url + game_json["local_team"]).json())
```

Cuando ya han sido recuperados los jugadores de ambos equipos se procede a analizar el fichero play-by-play y se calculan las estadísticas simples de dichos partidos para todos los participantes. Para realizar la actualización se debe realizar otra petición de tipo PUT al microservicio gestor de la base de datos con las estadísticas obtenidas al analizar el fichero del partido.

Cuando se analiza el fichero de partido también se debe actualizar el CSV que contiene la información defensiva. Cuando se registra un tiro en el fichero play-by-play se almacena el identificador del evento, el identificador del jugador que realizó el tiro y si fue acertado o no. Una vez que se termina de analizar el fichero play-by-play se analiza el fichero JSON que contiene las coordenadas del partido para encontrar para cada tiro el defensor más cercano. Posteriormente esta información es introducida en el fichero CSV y se añaden las posiciones y clúster del tirador y el defensor.

8.1.3. Cálculo de estadísticas.

Para el microservicio encargado de calcular las estadísticas avanzadas se crea una *app* con el nombre *advancedStatisticsCalculator*. Los puntos de conexión de este microservicio son:

Figura 8.4: Endpoints del microservicio para el cálculo de estadísticas avanzadas.

Hay un punto de acceso que calcula las estadísticas para un equipo, cuyo identificador es facilitado en la URL de la petición. También existe un punto de comunicación similar que calcula las estadísticas avanzadas del equipo y de sus jugadores. Para calcular estas estadísticas se utilizan las fórmulas que se explican en 6.2.2.

El otro punto de conexión se encarga de realizar un informe dados los identificadores de ambos equipos. Este informe contiene las estadísticas avanzadas de ambos equipos y la predicción del partido. Para obtener la predicción es necesario hacer una petición al microservicio de inteligencia artificial. En el cuerpo de la predicción se envía en formato JSON las estadísticas avanzadas de los dos equipos. La respuesta obtenida de esta llamada se pone en conjunto a las estadíticas avanzadas de ambos equipos y se devuelve como respuesta en formato JSON.

8.1.4. Inteligencia Artificial.

Este microservicio es contenido en la *app* de nombre *modelPredictor*. Como se comenta en 5.2.3 en este microservicio se almacenan los modelos de aprendizaje automático que intervienen en la API. Los modelos son previamente entrenados, ya que entrenarlos en el propio servidor supondría una carga excesiva y haría que bajase considerablemente el rendimiento de la API.

Una vez que los modelos han sido entrenados se guardan utilizando el paquete *joblib* que permite guardar objetos de Python en ficheros y poder reutilizarlos. Para guardar un modelo entrenado utilizamos el método *dump*. Por ejemplo, para guardar el modelo de clusterización de los bases es necesaria la siguiente línea de código:

```
joblib.dump(KMeans_pg_advanced, 'Models/kmeans_pg_advanced.sav')
```

Para utilizar los modelos se cargan con el método *load*. Para evitar estar teniendo que cargar los modelos cada vez que se realice una llamada la carga se realiza en el momento en el que se activa el microservicio. Para ello en el fichero *apps.py* del microservicio se define una clase que hereda de *AppConfig*, que es una clase de Django que permite cargar ciertos parámetros u objetos Python. Esta clase se ejecuta solamente cuando se inicia el microservicio y los objetos son instanciados para que puedan usarse en todo momento. Además de los modelos también se cargan los objetos que se encargan de escalar los datos de entrada de cada modelo.

8.2. Aplicación Android

El primer paso en el desarrollo de la aplicación Android es crear un proyecto en el IDE Android Studio. Al crear el proyecto podemos editar la información básica, la información más importante sobre el proyecto es el lenguaje en el que se realiza y el SDK mínimo en el que funcionará la aplicación. Para este proyecto se utiliza Kotlin como lenguaje básico y como versión mínima de SDK se utiliza Android 10.0.0, ya que se utilizan características que sólo se encuentran en esta versión y posteriores.

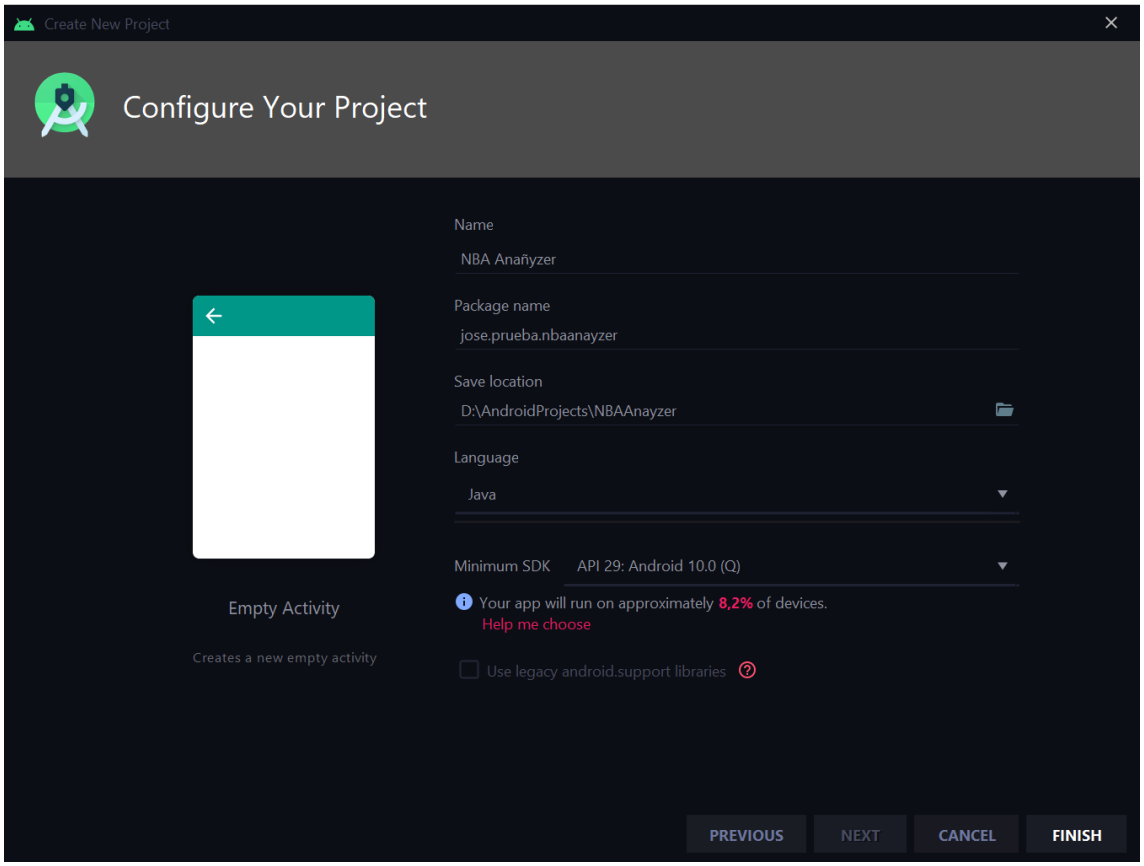


Figura 8.5: Creación del proyecto en Android Studio.

En Android para poder realizar las interfaces de usuario se debe crear un fichero XML que contenga los datos de los objetos que se verán en pantalla, en Android Studio se proporciona una ventana de previsualización para facilitar la tarea de diseño. Este fichero XML debe ser cargado desde el *Activity* que se encarga de gestionar la interfaz de usuario, es decir para cada pantalla de la aplicación se necesitan mínimo dos ficheros. Esto se complica al añadir el menú con pestañas, ya que cada pestaña se implementa como un *Fragment* y tiene asociado otro fichero XML. A esto hay que sumarle que para mostrar visualmente elementos en forma de lista se necesita de un *RecyclerView* dentro del fichero XML al que se le asocia otro fichero con la vista individual que tendrán los elementos.

Al tener tantos ficheros distintos se dificulta el proceso de identificación de errores. Para evitarlo, la estructura de ficheros del proyecto Android debía ser ordenada. Para ello se dividieron los ficheros en paquetes que comparten funcionalidad. Los ficheros relativos a clases en Kotlin se guardan en la carpeta *java/com.example.nbaanalyzer*. En la raíz de

esta carpeta se guardan las clases que son utilizadas por varias pantallas. Dentro de esta carpeta se crean los siguientes directorios:

- ***api***: en este directorio se guardan las clases que contienen la lógica que permite comunicarse con la API.
- ***models***: almacena las clases que representan los modelos necesarios para almacenar información que posteriormente debe ser representada en la aplicación.
- ***ui***: contiene todas las clases que se encargan de gestionar las interfaces de usuario. Este directorio está a su vez dividido en un directorio para los distintas vistas que hay en la aplicación.

En Android los *Activity* son los encargados de crear una ventana para que el usuario pueda interactuar con la aplicación [33]. Todos los proyectos creados con Android Studio tienen un fichero que contiene el *Activity* principal de la aplicación, que es cargado cuando la aplicación comienza a ejecutarse. Cada *Activity* se encarga de añadir los elementos declarados en un *layout* [1] mediante el método *setContextView(layoutID)*.

Al iniciar la aplicación por primera vez a la aplicación es necesario seleccionar el equipo al que pertenece el usuario, esta acción se realiza en *SelectYourTeamActivity*. Para mostrar los equipos seleccionables divididos en dos pestañas, tal y como se ve en el capítulo sobre las interfaces de usuario 7. Para poder mostrar las pestañas se utilizan varios objetos importantes de android:

- ***Fragment***: En la documentación oficial lo definen así: "Un *Fragment* representa un comportamiento o una parte de la interfaz ... Puedes combinar varios fragmentos en una sola actividad para crear una IU multipanel" [34].
- ***FragmentStatePagerAdapter***: es una clase que permite gestionar distintos *Fragments* en páginas [35].

FragmentStatePagerAdapter es una interfaz, por lo que para poder utilizarlo es necesario implementarlo en una clase. Esta implementación se hacer en la clase *TabsAdapter*, esta clase contiene una lista con los *Fragments* que serán cargados como páginas y en otra lista se almacenan los títulos de las pestañas.

Para la selección de equipo hay dos pestañas, por lo que son necesarios dos *Fragments*. La lógica que incluyen ambos es igual y lo único que cambia son los equipos que se cargan. Para cargar los equipos se añade en en layout un elemento de tipo *ReciclerView*. Este tipo de elemento permite cargar los datos de una lista siguiendo un patrón, en este caso se carga a la izquierda el logo y a la derecha el nombre del equipo.

Una vez que el usuario selecciona el equipo se guarda el identificador del equipo en las preferencias compartidas de la aplicación. Para poder guardar el identificador primero tenemos que recuperar las preferencias compartidas de la aplicación. Primero accedemos al contexto de la aplicación mediante el objeto *applicationContext*, este objeto contiene el método que permite obtener las preferencias compartidas: *getSharedPreferences*. Este método recibe como parámetro la etiqueta que sirve para identificar los parámetros que contienen esta preferencia. A continuación utilizamos el método *edit* para poder añadir el identificador del equipo seleccionado.

[1] Un *layout* es un fichero XML que contiene la información sobre el diseño de los elementos gráficos que se muestran en una pantalla de la aplicación

Cuando el equipo es seleccionado y esta selección ha sido almacenada es necesario volver a la *Activity* principal. Para ello se realiza un *intent* que permite navegar entre distintos *Activity* y enviar información. Una vez que la aplicación inicia la *Activity* principal hay que inicializar los elementos visuales. En este caso hay dos elementos principales que deben ser inicializados:

- **Menú de pestañas**. Se inicializa como se ha comentado anteriormente y lo único que cambia son los *Fragments* que se cargan.
- **Menú lateral**. Para el menú lateral se utiliza en primer lugar el elemento *NavigationView* que contiene el menú de la aplicación. Para poblar el menú con las distintas opciones hay que utilizar el elemento *DrawerLayout* que es un contenedor de más alto nivel que permite poblar las opciones del menú con distintos *Activities*. Las opciones visibles en el menú son: My Team, Teams, Help.

En la opción My Team se muestra la información sobre el equipo seleccionado por el usuario. En esta pantalla se muestra un menú con dos pestañas: Team Preview y Boxscore. En la pestaña Team Preview se muestran varias gráficas con información sobre el equipo. Para obtener los datos es necesario realizar peticiones a la API. Para realizar estas peticiones se utiliza la librería *Retrofit*, para añadir esta librería hay que incluirla en el fichero *gradle* de nivel de aplicación:

```
implementation "com.squareup.retrofit2:retrofit:2.9.0"
implementation "com.squareup.retrofit2:converter-moshi:2.9.0"
```

La librería *Moshi* se utiliza para poder traducir las respuestas en formato JSON a objetos Kotlin y poder trabajar con mayor facilidad. Para poder realizar las peticiones necesarias se debe instanciar un cliente *Retrofit* que será el encargado de realizar las peticiones HTTP. Para que el cliente funcione correctamente es necesario implementar distintos objetos. En primer lugar se crea un fichero que contiene todos los modelos que contendrán los datos proporcionados en la respuesta del servidor. Cada modelo es en realidad un *data class* de Kotlin, que son clases especiales que solo tienen atributos. Para que *Moshi* pueda detectar a qué parte de la respuesta equivale cada atributo se utiliza la anotación *@field:Json(name = Response field")*, en la variable *name* se proporciona el nombre del campo de la respuesta que se asocia al atributo de la clase.

Cuando los modelos de las respuestas están creados hay que crear el servicio *Retrofit*, que se incluye en una interfaz en la que la lógica es añadida automáticamente en tiempo de compilación por *Retrofit*. En esta interfaz se añade una función por cada punto de conexión que será consumido. Cada función tendrá una anotación que indica el método HTTP de conexión y la URL del punto de acceso, por ejemplo: *@GET(."dvancedStatisticsCalculator/team-players-stats/teamID")*. Al definir la función los parámetros de esta son anotados para indicar el tipo de parámetro, los puntos de conexión que se utilizarán en la aplicación solo tendrán parámetros en la URL, por lo que la anotación es de la siguiente forma: *@Path("teamID") team_id: Int*. La función para obtener las estadísticas avanzadas de un equipo y sus jugadores queda así:

```
@GET("advancedStatisticsCalculator/team-players-stats/{teamID}")
fun getTeamStats(@Path("teamID") team_id: Int): Call<TeamResponse>
```

Por último, se utiliza una clase más para definir e inicializar el cliente *Retrofit* en un único sitio y no en todos los ficheros que realicen llamadas a la API. Las opciones que

de configuración que se proporcionan al cliente son la URL base de la API y el conversor *Moshi*. Es importante que se le concedan permisos de internet a la aplicación, para ello debemos incluir en el manifiesto de la aplicación:

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

Con estas clases implementadas ya se pueden realizar llamadas a la API desde la aplicación. En la pestaña de Team Preview se realizan dos llamadas, una para obtener las estadísticas avanzadas del equipo y sus jugadores y otra para la obtención de la distribución de puntos del equipo. Tal y como funciona el gestor de memoria Android las llamadas a la API no pueden realizarse en el hilo principal de ejecución, ya que bloquearían la aplicación. Para realizar la llamada en otro hilo se utiliza la librería *Anko* que permite la creación de un hilo de manera sencilla, solo es necesario incluir el código que se desea ejecutar en otro hilo en un bloque *doAsync*{}. Los datos obtenidos de la respuesta son cargados en las gráficas, que son objetos especiales de la librería *MPAndroidChart*. Esta librería permite la creación de gráficas dinámicas, de manera que se crean en el momento en el que la vista se inicia.

En la pestaña Boxscore se muestra una tabla con todas las estadísticas del equipo y sus jugadores. Para poder mostrar esta vista con aspecto tabular es necesario utilizar otro tipo de *RecyclerView* al que se le proporciona como entrada una lista de objetos que tienen las estadísticas como atributos. El *RecyclerView* se encarga de dibujar los elementos de la vista, además a cada elemento se le ha añadido un *listener* para que cuando se pulse sobre él se lance un *Activity* que muestre la información sobre el jugador. Para iniciar esta *Activity* se envían las estadísticas del jugador para evitar hacer otra llamada a la API. Las estadísticas se añaden al *Intent* como extras, cuando el *Activity* se inicia estos extras se recuperan y se guardan como atributos del *Activity*.

En el *Activity* del jugador se reutiliza el menú de pestañas y se incluyen dos *Fragments*. El primer *Fragment* que añade muestra información a modo de resumen en la parte superior y en la parte inferior se muestran dos gráficas. La información sobre las estadísticas que se añade en esta vista se recoge de los extras del *Intent* que inicia al *Activity*. Para obtener la información defensiva sobre el jugador se obtiene realizando una llamada a la API y se carga en la gráfica.

En el otro *Fragment* de la vista de jugadores son cargadas las gráficas de tiro del jugador. Estas gráficas son imágenes que se crean en el servidor de la API utilizando la librería de Python *Matplotlib* y se almacenan en el servidor. Para cargar estas imágenes se cargan en la aplicación Android se utiliza la librería *Picasso* que permite cargar y visualizar una imagen dada su URL.

En la opción Teams del menú lateral se muestran los equipos divididos en conferencias para poder visualizar las estadísticas de otro equipo. Para mostrar los equipos se reutiliza el código para la selección del equipo del usuario. La diferencia es que al hacer click sobre un equipo se inicia un *Activity* nuevo para cargar la información sobre el equipo seleccionado. En el *Intent* que inicia la nueva pantalla se envía el identificador del equipo que ha sido pulsado para que se pueda realizar la llamada correspondiente.

Este *Activity* también contiene un menú con cuatro pestañas: Team Preview, Boxscore, Inform y Defend Inform. Las dos primeras pestañas reutilizan el código explicado en el *Activity* que muestra la información del equipo seleccionado por el usuario. El *Fragment* que carga la pestaña Inform realiza una llamada al punto de conexión de la API que

se realiza el informe dado el identificador de dos equipos. En la pestaña Defend Info se muestran los titulares del equipo rival y a la derecha quinteto titular del equipo del usuario que mayor eficacia defensiva tiene para los jugadores rivales, en función de los clústers de los jugadores rivales. La llamada que se necesita realizar también tiene como parámetro los identificadores de los dos equipos. El identificador del equipo del usuario se obtiene leyendo las preferencias de la aplicación y el del equipo rival se obtiene del *Intent* que inició el *Activity*.

La última opción que aparece en el menú lateral es Help. En esta vista se proporciona información que ayuda al usuario a entender la información que se muestra en la aplicación. Se muestra un menú de pestañas en la que en cada pestaña muestra ayuda sobre un tema diferente. Las pestañas que se incluyen son:

- **Charts Info**: en esta pestaña se muestra la explicación de las distintas gráficas que se muestran en la aplicación.
- **Game Predictions**: esta pestaña se utiliza para informar al usuario sobre cómo se realizan las predicciones de los partidos. Se explica cómo ha sido entrenado el modelo de aprendizaje que se encarga de las predicciones.
- **Play Style**: en esta pestaña se incluye la información sobre la clusterización de los jugadores según su posición. Se añade la descripción sobre cada clúster que se forma, explicando cuales son las cualidades de los jugadores de cada clúster.

La información que se incluye en estas pestañas se hace mediante texto fijo ya que se ofrece de manera informativa sobre la propia aplicación.

CAPÍTULO 9

Conclusiones y Lineas futuras

A lo largo de este capítulo se realiza una valoración del proyecto desarrollado, incluyendo los problemas que se han encontrado durante el desarrollo. En la conclusión se tratará de evaluar el proyecto en función de los requisitos definidos en este documento y se hablará sobre aspectos importantes del sistema desarrollado. No se pretende realizar una justificación de las decisiones tomadas, ya que para eso están todos los capítulos anteriores.

También se hablará de las líneas futuras, es decir, de cómo puede evolucionar el proyecto a medio largo plazo. Se pretende establecer una hoja de ruta para la evolución del proyecto hasta que llegue a cubrir campos que no han podido ser abarcados en el proyecto.

9.1. Dificultades solventadas durante el desarrollo

El proyecto se dividía en dos partes, la API y la aplicación Android, que usan tecnologías muy distintas por lo que el desarrollo de cada parte ha sido muy distinto. En cada parte las dificultades que se han encontrado tienen que ver con el lenguaje utilizado y las librerías.

El desarrollo de la API se ha realizado con el paquete de Python Django REST Framework y el principal problema al que me he enfrentado durante el desarrollo es el tratamiento de ficheros. En este paquete no se puede crear una respuesta ni una URI para un objeto que no esté almacenado en la base de datos. Esto dificultó la creación de las gráficas de tiro, ya que la idea era que se creasen las gráficas de tiro en la parte del servidor mediante la librería *Matplotlib* y enviarla como respuesta a la aplicación. Para solventar este inconveniente se tuvieron que crear las gráficas de tiro previamente y almacenarlas en la base de datos para que Django puediera generar una URI y que fuesen accesibles mediante una URL. Además se añadió la lógica que permite volver a generar las gráficas y almacenarlas. Por lo general, no ha habido más problemas ya que Python es un lenguaje con una fuerte comunidad detrás y del que es bastante fácil encontrar documentación y ejemplos.

Desarrollando la aplicación móvil si que se tuvieron más problemas debido a que Kotlin es un lenguaje relativamente nuevo y, a pesar de que tiene un fuerte soporte por parte de Google, todavía es complicado encontrar información sobre algunos aspectos. Durante el desarrollo del menú lateral hubo complicaciones, ya que el objeto que se utilizar se añadió este mismo año. Anteriormente se realizaba de otra forma que ya está deprecada y que

no funcionaba en la versión en la que se desarrolla la aplicación. El problema principal fue que al ser un objeto tan reciente la documentación existente era prácticamente nula.

También se encontraron problemas a la hora de realizar las llamadas a la API, ya que tienen que ser realizadas en un hilo diferente del principal. Utilizar hilos siempre es algo complicado, sobre todo cuando dependes de información de un hilo secundario para actualizar elementos de la interfaz. El descubrimiento de la librería *Anko* solventó este problema, ya que simplifica mucho el uso de hilos y la actualización de elementos de la interfaz de usuario.

El desarrollo Android necesita de un flujo de trabajo claro, ya que se necesita de varios ficheros para implementar las funciones de la aplicación . Es por esto que por lo general hay ciertos puntos en los que el desarrollo suele ser confuso y causar problemas de menor medida, que suelen ser solventados en poco tiempo.

9.2. Líneas futuras

Una de las principales mejoras que se puede incluir es un panel de administrador que permita la carga de los ficheros de los partidos. Actualmente estos ficheros deben ser añadidos directamente al servidor donde se encuentra instalada. En este panel se permitiría subir ficheros de formato CSV y JSON y mediante una llamada a la API. La lógica para almacenar un fichero puede ser reutilizada del método que almacena las gráficas de tiro. El panel de administrador puede ser incluido en una vista añadida al que ya proporciona Django.

La información del usuario se guarda de manera local en la aplicación Android, es decir, si el usuario cambia de dispositivo pierde la información sobre el equipo que había seleccionado. Una posible mejora sería añadir un sistema de login de usuario. Para este login sería necesario añadir un nuevo modelo a la base de datos que se encargue de almacenar la información sobre el usuario. Se incluiría una clave foránea para relacionar a cada usuario con el equipo que ha seleccionado. Para almacenar la contraseña del usuaria debe utilizarse alguna técnica que permita encriptar la contraseña.

Otro punto que se puede mejorar en el futuro es el modelo de predicción de partidos. Debido a que el alcance del proyecto cubría muchos más campos a parte del modelo de predicción no se ha podido dedicar el tiempo suficiente a la creación de un modelo a medida. Para mejorar la predicción de partidos se puede desarrollar una red neuronal y personalizar todos los parámetros posibles para que se ajuste lo mejor posible a los datos de entrada. Esto proceso es largo y deben de realizarse muchas mediciones y comparaciones para evaluar distintos modelos con combinaciones de parámetros y configuraciones distintas y seleccionar la mejor.

Un gran avance para este proyecto sería la utilización de datos de la otras ligas de baloncesto. Mientras se definía el proyecto se evaluó la posibilidad de utiliza datos de la liga española en lugar de los datos de la NBA. Investigando esta posibilidad nos encontramos con el problema de que en tema de datos la liga española aún está verde. En un futuro que esto cambie se puede adaptar el sistema para que también sea capaz de analizar y mostrar los datos de otras ligas. Esto dependerá del avance que se hagan en las respectivas ligas

y realicen en el campo de captación de datos, ya que la aplicación depende de los datos generados por las ligas.

Los ficheros de posicionamiento se utilizan para realizar las gráficas de tiro y encontrar al defensor de cada tiro. Este uso podría extenderse hacia detectar jugadas de ataque para cada tiro, de manera que se puedan detectar todos los movimientos realizados por el equipo atacante en cada tiro. De esta manera se podría extraer información sobre las jugadas más típicas de cada equipo y de qué forma se pueden defender con mayor eficacia.

9.3. Conclusiones

El objetivo principal del proyecto era generar información útil a partir de los datos que proporciona la NBA. Para ello se definían una serie de requisitos que debían ser implementados. La lógica necesaria para que el sistema resultante cumpla con los requisitos. El sistema está dividido en dos partes, API y aplicación Android, de manera que en cada parte se implementa la lógica referente a los requisitos que deba implementar. Con el trabajo conjunto de ambas partes del sistema se ha conseguido crear una aplicación Android que muestre de manera sencilla la información que se extrae y procesa de los ficheros proporcionados por la NBA. Se ha conseguido que, a pesar de que la información que se muestra en la aplicación se obtenga de una manera compleja analizando ficheros extensos, se muestre de una manera clara y visual al usuario. Además, se le muestra al usuario información necesaria para identificar los puntos fuertes y débiles de los rivales, facilitando así la labor de análisis previo de los partidos.

Se decidió realizar un desarrollo de microservicios que forman la API ya que permitía modularizar las funcionalidades del sistema. También permite aislar errores, de manera que si se produce algún problema en un microservicio los demás pueden seguir funcionando con normalidad. El lenguaje que se decidió utilizar para los microservicios fue Python, a pesar de que la división en microservicios permite utilizar distintos lenguajes para los distintos microservicios. Python es un lenguaje muy versátil y con mucho soporte hoy en día, y resulta ser muy legible. También ha sido seleccionado porque se adapta muy bien a las necesidades del proyecto. El desarrollo en Python no ha sido difícil ya que las librerías utilizadas, como Django, facilitan mucho el proceso haciendo que solo hay que preocuparse por desarrollar la lógica propia.

El desarrollo de los modelos de inteligencia artificial también se realizó sobre el lenguaje Python. La librería de *scikit-learn* ha facilitado el proceso ya que proporciona métodos para crear los modelos. En estos métodos resulta muy fácil parametrizar los modelos, lo que ha permitido que se evalúen distintos modelos. Para entender los modelos evaluados se ha realizado una investigación y así poder seleccionar los valores de los parámetros de manera adecuada.

El desarrollo Android ha sido tedioso debido a la gran cantidad de ficheros que se necesitan. Para solventar este inconveniente se utiliza un sistema de paquetes de manera que en cada paquete se incluyen los ficheros que cumplen una función similar. Durante el desarrollo se han reutilizado partes de código que definen una funcionalidad que ha sido utilizada en varias partes de la aplicación, como el menú con pestañas.

El desarrollo general del proyecto ha sido positivo y se han cumplido todos los objetivos

propuestos. La realización de la memoria también ha sido positiva y ha servido para planificar de forma correcta el proyecto. Definir los requisitos y aspectos clave del proyecto en la memoria ha facilitado el desarrollo, ya que ha permitido tener claro el objetivo de cada parte que conforma el sistema.

Bibliografía

1. NBA, *National Baskteball Association - Wikipedia* (https://es.wikipedia.org/wiki/National_Basketball_Association), (accessed: 01.09.2016).
2. NBA, *NBA Basketball Analytics Hackathon* (https://hackathon.nba.com).
3. SportEasy, *SportEasy web page* (https://www.sporteasy.net/es/sports/basket/).
4. E. S. Basketball, *Easy Stats Basketball web page* (https://www.easystatsapp.com/).
5. B. Basketball, *Breakthrough Stats App web page* (https://www.breakthroughbasketball.com/apps/stats/).
6. H. A. Baskteball, *Hudl Assists web page* (https://www.hudl.com/products/assist/basketball).
7. S. Vu, *Sort Vu Basketball web page* (https://www.statsperform.com/team-performance/basketball/).
8. Oracle, *MySQL Reference Manual* (https://dev.mysql.com/doc/refman/8.0/en/).
9. Python, *Python Documentation* (https://docs.python.org/3/).
10. Django, *Django Project* (https://www.djangoproject.com).
11. Scikit-Learn, *Scikit-Learn* (https://scikit-learn.org/stable/).
12. NumPy, *NumPy* (https://numpy.org).
13. Pandas, *Pandas* (https://pandas.pydata.org).
14. Matplotlib, *Matplotlib* (https://matplotlib.org).
15. JetBrains, *Kotlin Programming Language* (https://kotlinlang.org).
16. Google, *Google I/O* (https://android-developers.googleblog.com/2017/05/google-io-2017-empowering-developers-to.html).
17. Google, *Pautas de diseño para Material Design* (https://material.io/design/).
18. Wikipedia, *Diseño de sistemas* (https://es.wikipedia.org/wiki/Dise%C3%B1o_de_sistemas).
19. Wikipedia, *Transferencia de Estado Representacional* (https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional).
20. Oracle, *¿Qué es una base de datos relacional?* (https://www.oracle.com/es/database/what-is-a-relational-database/).

21. D. Oliver, *Basketball on Paper*.

22. Wikipedia, *Regresión Lógica* (https://es.wikipedia.org/wiki/Regresi%C3%B3n_log%5C%C3%5C%ADstica).

23. Wikipedia, *Máquina de Soporte Vectorial* (https://es.wikipedia.org/wiki/M%C3%A1quinas_de_vectores_de_soporte).

24. Wikipedia, *Bosque Aleatorio* (https://es.wikipedia.org/wiki/Random_forest).

25. Wikipedia, *Red Neuronal Artificial* (https://es.wikipedia.org/wiki/Red_neuronal_artificial).

26. Wikipedia, *Validación Cruzada* (https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada).

27. Wikipedia, *Matriz de Confusión* (https://es.wikipedia.org/wiki/Matriz_de_confusi%C3%B3n).

28. Wikipedia, *Análisis de grupos* (https://es.wikipedia.org/wiki/An%C3%A1lisis_de_grupos).

29. Wikipedia, *K-medias* (https://es.wikipedia.org/wiki/K-medias).

30. Wikipedia, *DBSCAN* (https://es.wikipedia.org/wiki/DBSCAN).

31. Wikipedia, *OPTICS algorithm* (https://en.wikipedia.org/wiki/OPTICS_algorithm).

32. SKLearn, *SKLearn metrics* (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise_distances.html#sklearn.metrics.pairwise_distances).

33. Google, *Developers Documentation Activity* (https://developer.android.com/reference/android/app/Activity).

34. Google, *Developers Documentation Fragment* (https://developer.android.com/guide/components/fragments).

35. Google, *Developers Documentation FragmentStatePagerAdapter* (https://developer.android.com/reference/androidx/fragment/app/FragmentStatePagerAdapter).

Apéndices

APÉNDICE A

Glosario

TMOffRtg	Rating ofensivo de un equipo.
TeamDefRtg	Rating defensivo de un equipo.
NetRtg	Rating neto.
PTS	Puntos anotados.
TMPTS	Puntos anotados por un equipo.
CPTS	Puntos concedidos por un equipo.
FGA	Tiros de campo intentados.
FGM	Tiros de campo anotados.
TMFGA	Tiros de campo intentados por un equipo.
CFGA	Tiros de campo intentados que concede un equipo.
TMFGM	Tiros de campo anotados por un equipo.
CFGM	Tiros de campo anotados que concede un equipo.
CFG %	Porcentaje de acierto en los tiros concedidos por un equipo.
3FGA	Triples intentados.
C3FGA	Triples intentados contra un equipo.
FG3M	Triples anotados.
TMFG3M	Triples anotados por un equipo.

OREB	Rebotes ofensivos.
DREB	Rebotes defensivos.
TMOR %	Porcentajes de rebotes ofensivos de un equipo.
COR %	Porcentaje de rebotes concedidos por un equipo.
COREB	Rebotes ofensivos concedidos.
CDREB	Rebotes defensivos concedidos.
FTA	Tiros libres intentados.
CFTA	Tiros libres intentados que concede un equipo.
FTM	Tiros libres anotados.
CFTM	Tiros libres anotados que concede un equipo.
TMFTA	Tiros libres intentados por un equipo.
TMFTM	Tiros libres anotados por un equipo.
FT %	Porcentaje de acierto en tiros libres.
AST	Asistencias.
TMAST	Asistencias repartidas por un equipo.
STL	Robos.
BLK	Tapones.
TMBLK	Tapones realizados por un equipo.
TOV	Pérdidas.
TMTOV	Pérdidas cometidas por un equipo.
CTO	Total de pérdidas cometidas contra un equipo.
MIN	Minutos jugados.
TMMIN	Minutos jugados por un equipo.

APÉNDICE B

Guía de Instalación

En este apéndice se explican los pasos que se deben seguir para instalar las dos partes que conforman este proyecto. La instalación se realizará sobre un equipo con Windows 10 de 64 bits como sistema operativo. En primer lugar se instalará el motor de bases de datos, que en este caso es MySQL. El próximo paso es la instalación de Python y las herramientas necesarias para poder instalar la API. Una vez que la API está instalada y operativa hay que instalar Android Studio y los complementos necesarios para poder ejecutar la aplicación en un dispositivo virtual. Tanto el código de la API como el de la aplicación Android se encuentran en repositorios de GitHub.

Es necesario descargar ambos proyectos y extraerlos para poder ejecutarlos posteriormente. Primero abrimos la página del repositorio y veremos una página como la siguiente:

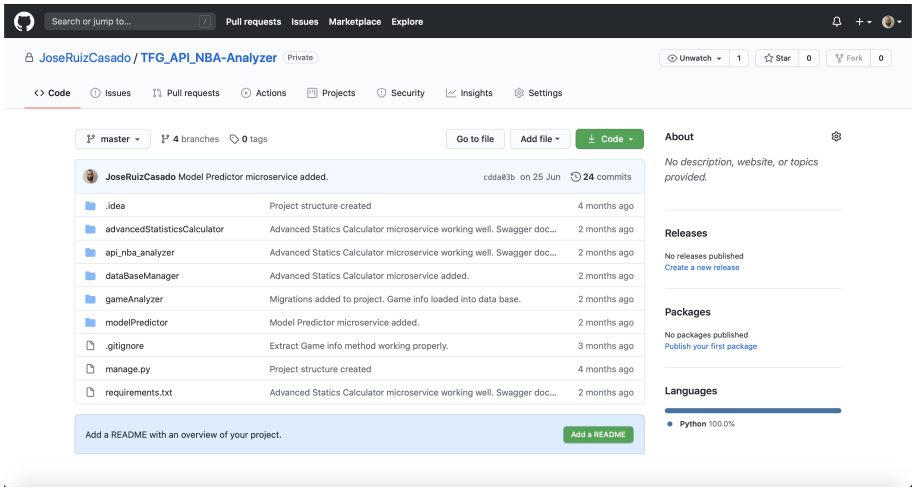


Figura B.1: Página de repositorio del proyecto de la API.

En esta página pulsamos sobre el botón *CODE* se abre un desplegable en el que nos aparece una opción para descargar el proyecto.

En este desplegable pulsamos la opción *Download ZIP*, que descarga el proyecto comprimido en un archivo de extensión ZIP. Una vez se descarga el fichero debemos descomprimirlo en un directorio para abrirlo posteriormente. El sistema operativo Windows 10 incluye

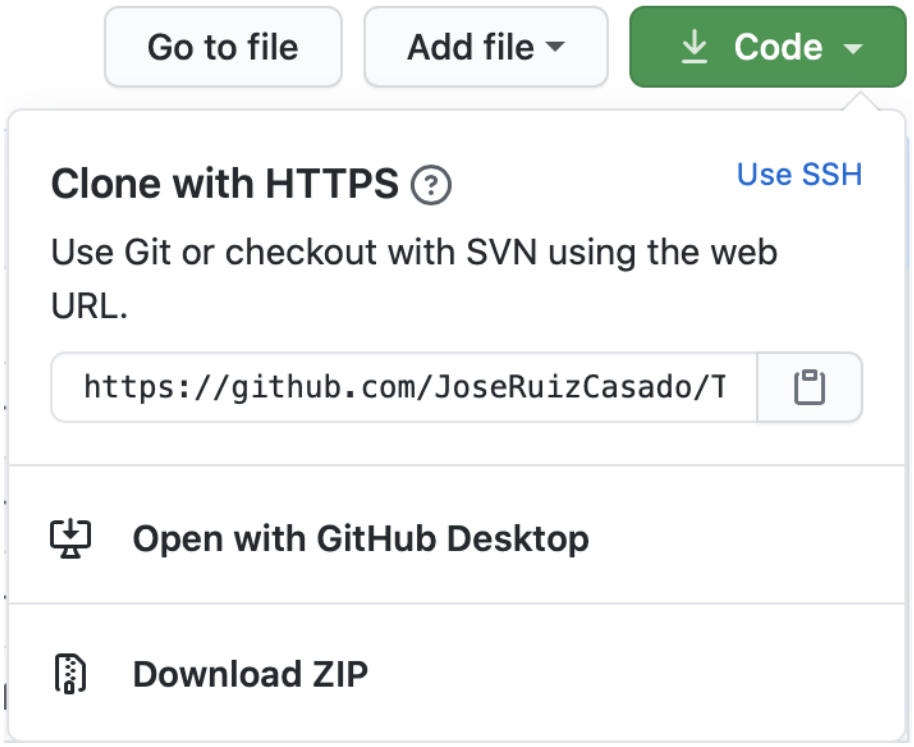


Figura B.2: Página de descarga de MySQL.

una herramienta que permite descomprimir archivos, haciendo click derecho en el archivo descargado y pulsamos sobre la opción *Extraer todo*.

Para descargar el proyecto con la aplicación Android se sigue el mismo proceso realizado para el proyecto de la API. La página del repositorio se puede acceder pulsando este enlace.

Instalación de MySQL.

La instalación de MySQL comienza descargando el instalador en la página de descarga de MySQL. Esta página de descarga es para el instalador para el sistema operativo Windows de 32 bits, ya que no hay versión para 64 bits.

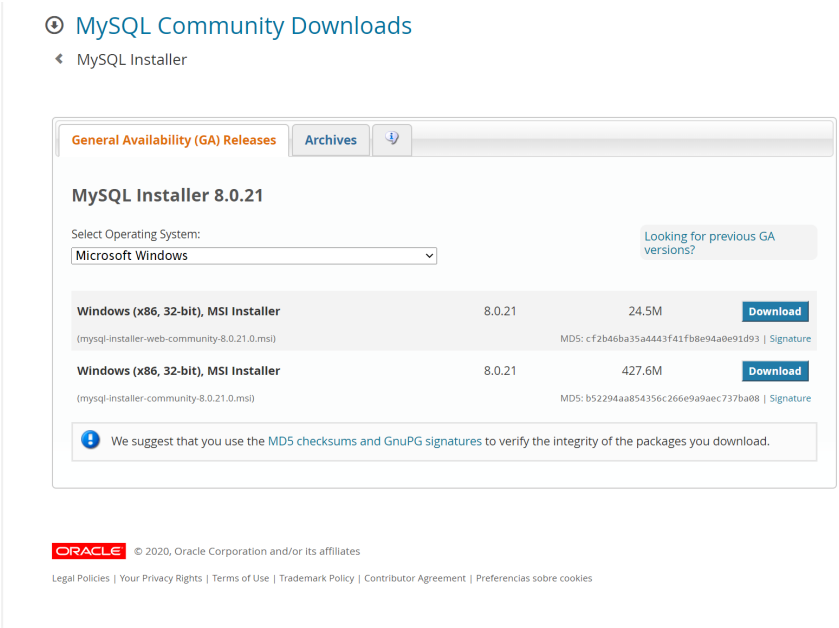
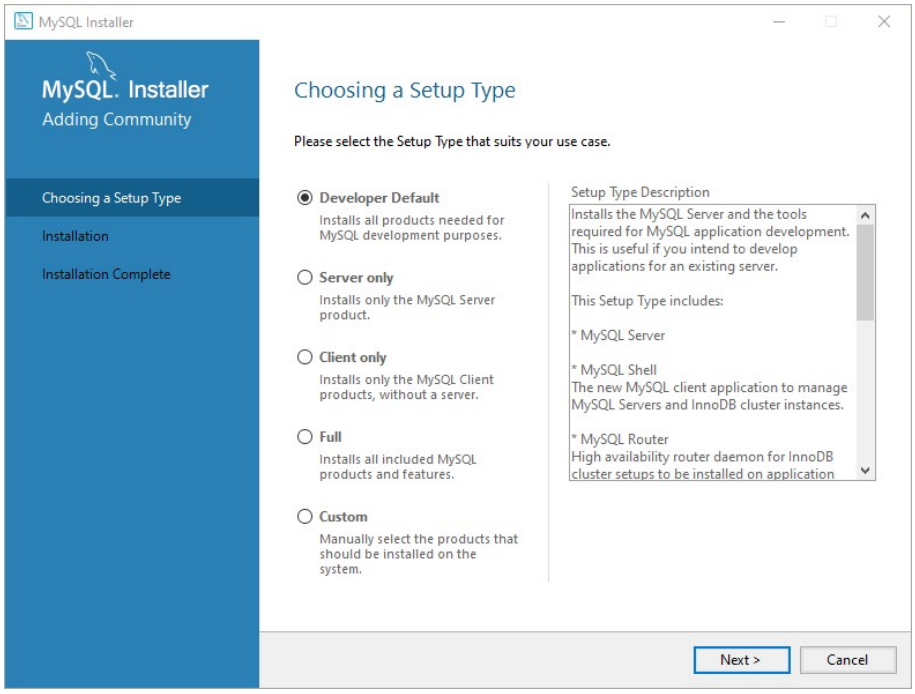
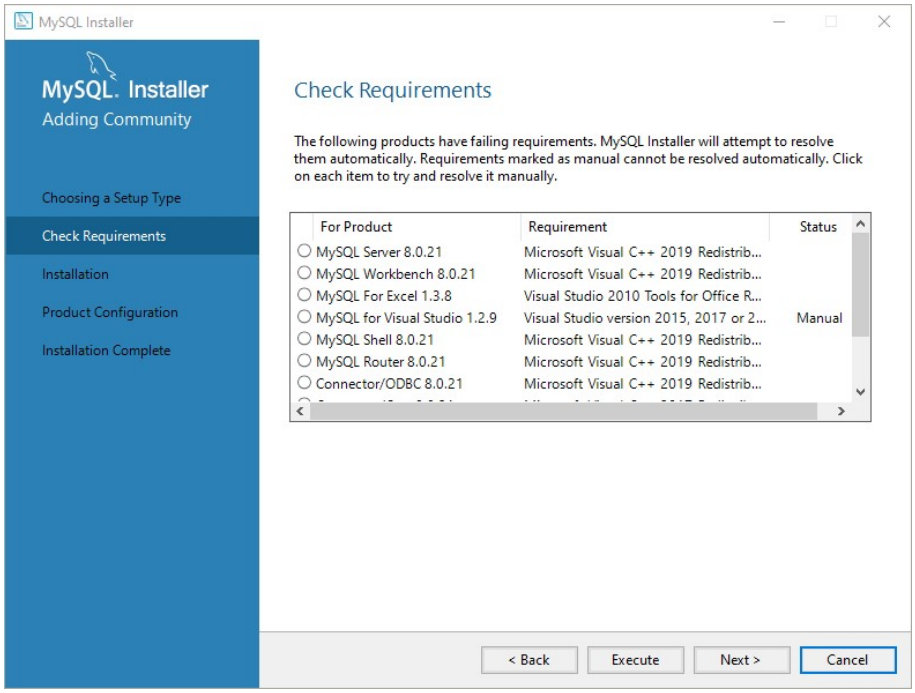


Figura B.3: Página de descarga de MySQL.

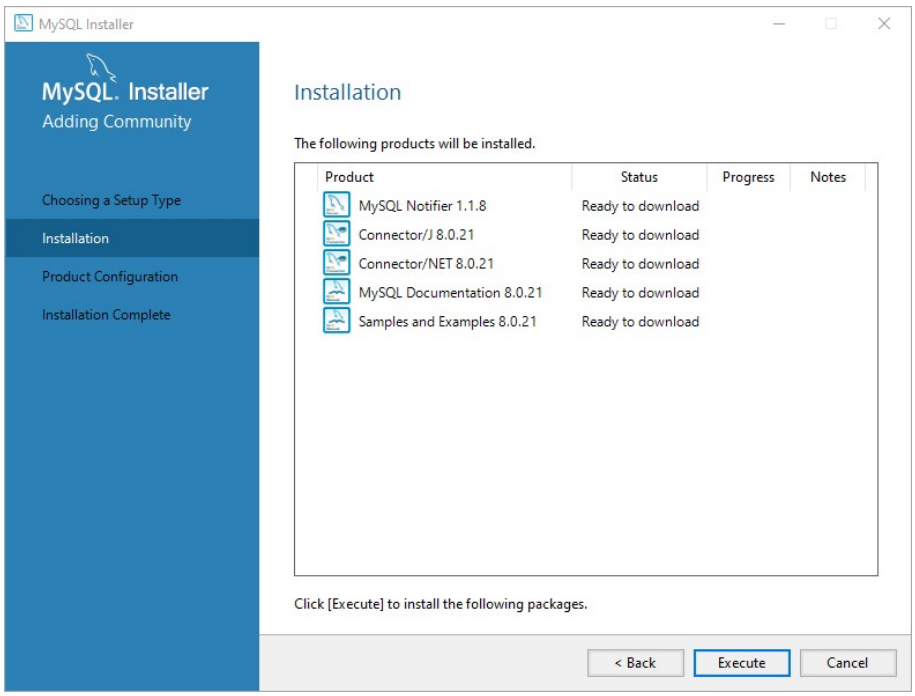
Una vez se descarga el instalador lo ejecutamos y empieza a cargar. Una vez termina de cargar aparece una pantalla para seleccionar el tipo de instalación que queremos realizar.



Es recomendado instalar la versión de desarrolladores, ya que instala los componentes que son necesarios para desarrollo software. En esta versión se incluyen diversos conectores que permiten empotrar instancias de bases de datos MySQL dentro de sistemas de información. Hacemos click en el botón *next* y se nos muestra un resumen con todas las herramientas que se van a instalar.



En esta pantalla volvemos a hacer click sobre el botón *Next* y aparece una pantalla con el software de MySQL que se va a instalar.



En esta última pantalla hacemos click sobre el botón de *Execute* y comienza la instalación de los productos de MySQL. Una vez que finalice la operación de instalado aparece una pantalla informándonos de que el proceso de instalación ha sido completado con éxito.

Instalación de Python.

Para instalar Python es necesario acceder a la página de descargas de Python. Para poder descargar la versión 3.7 es necesario bajar hasta la tabla de versiones y se selecciona la versión 3.7.7. Al hacer click en el link se abre una página que contiene la siguiente tabla:

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		d348d978a5387512fbc7d7d52dd3a5ef	23161893	SIG
XZ compressed source tarball	Source release		172c650156f7bea68ce31b2fd01fa766	17268888	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	47b06433e242c8eb848e035965a860ac	29163525	SIG
Windows help file	Windows		89bb2ea8c5838bd2612de600bd301d32	8183265	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	6aa3b1c327561bda256f2deebf038dc9	7444654	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	e0c910087459df78d827eb1554489663	26797616	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	d1d09dad50247738d8ac2479e4cde4af	1348896	SIG
Windows x86 embeddable zip file	Windows		29672b400490ea21995c6dbae4c4e1c8	6614968	SIG
Windows x86 executable installer	Windows		e9db9cf43b4f2472d75a055380871045	25747128	SIG
Windows x86 web-based installer	Windows		8b326250252f15e199879701f5e53c76	1319912	SIG

En esta tabla seleccionamos el ejecutable *Windows x86-64 executable installer*. Una vez descargado el fichero se ejecuta y encontramos la siguiente pantalla:

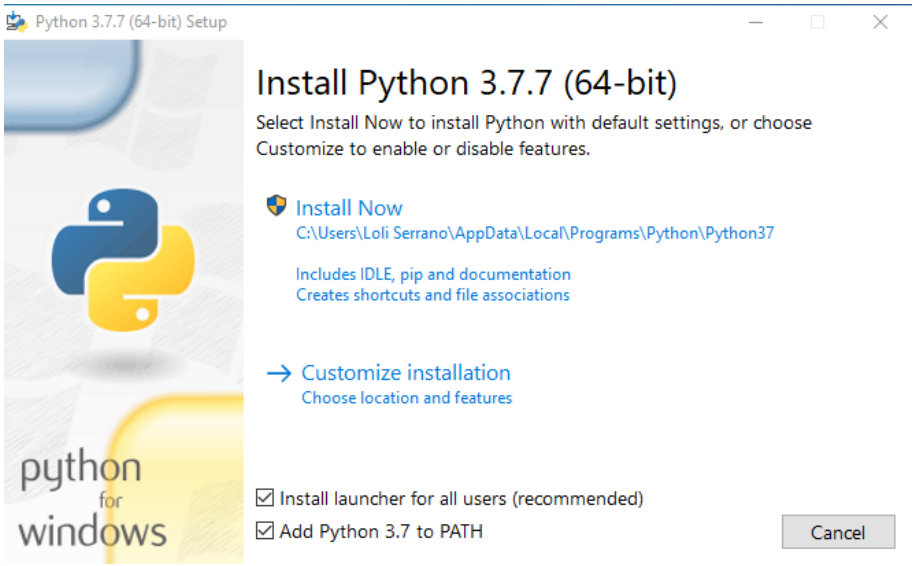


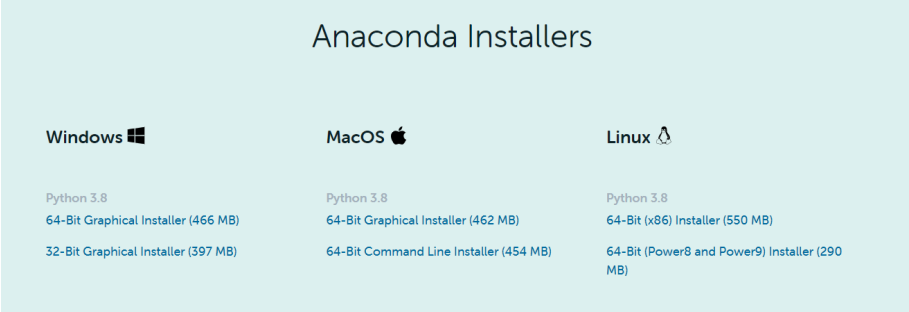
Figura B.4: Página de descarga de Python

Es importante seleccionar la casilla para que se añada Python a las variables del sistemas, para facilitar su uso y evitar posible errores. Se puede personalizar la instalación, aunque para facilitar el proceso es recomendable utilizar la opción por defecto. Esta instalación incluye documentación y el gestor de paquetes *pip* que facilita la instalación de paquetes y librerías. Una vez que se pulsa sobre el botón de *Install Now* comienza la instalación.

Una vez finaliza la instalación de Python podemos proceder a instalar *Anaconda*. Anaconda es un software que permite la creación y gestión de entornos virtuales. Estos entornos virtuales permiten crear un entorno aislado para los proyectos de Python. Esto permite

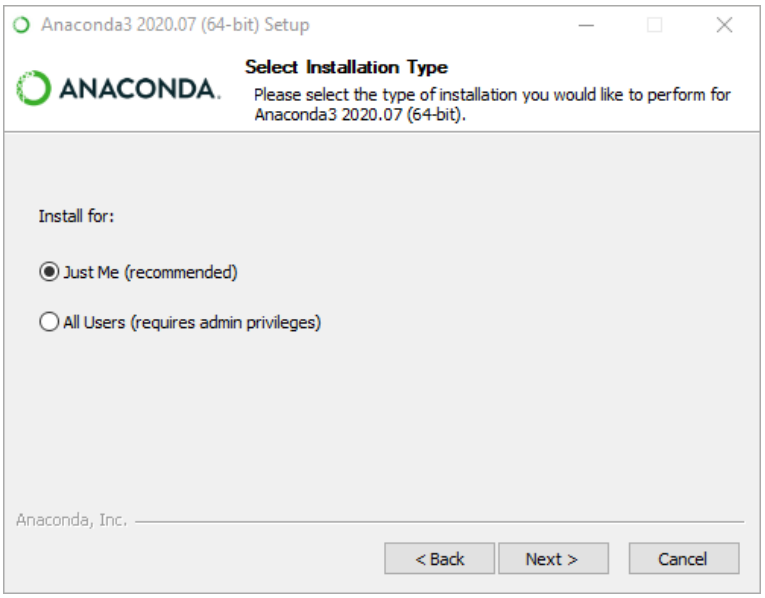
que se pueden separar las dependencias de cada proyecto, evitando incompatibilidades entre paquetes.

Para instalar Anaconda accedemos a la página de descargas de para instalar la versión individual de Anaconda. En esta página buscamos la siguiente sección:



Se seleccionará la versión de Windows de 64 bits. Aunque la la versión de Python que se muestra es la 3.8 pueden crearse entornos virtuales para otras versiones. Cuando ejecutamos el instalador aparace la siguiente pantalla:

Al pulsar sobre el botón *Next* nos muestra la licencia que debemos aceptar para comenzar con el proceso de instalación. Una vez que aceptamos la licencia necesitamos seleccionar si la instalación se va a realizar para todos los usuarios o solo para el usuario actual.



La siguiente pantalla que aparece es para seleccionar la ruta en la que se instalará el software de Anaconda. Es importante seleccionar un directorio en el que posteriormente PyCharm tenga permiso para leer los archivos generados por Anaconda.

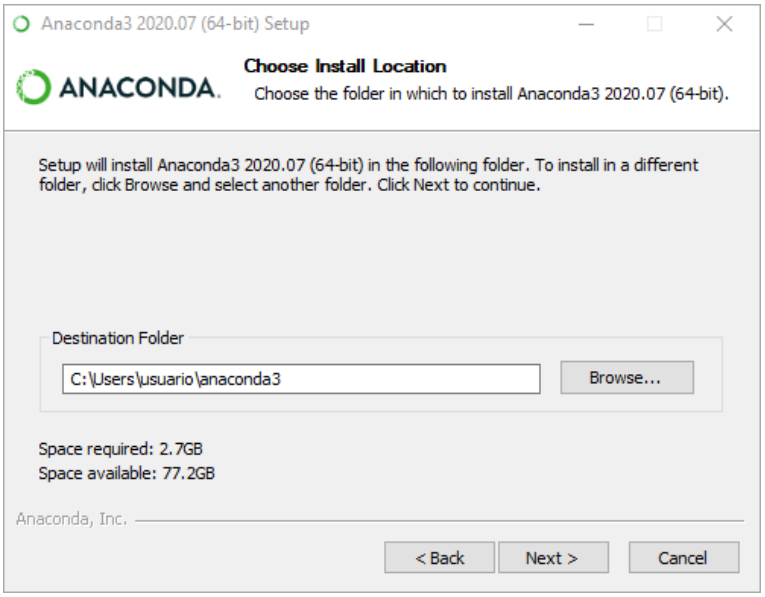
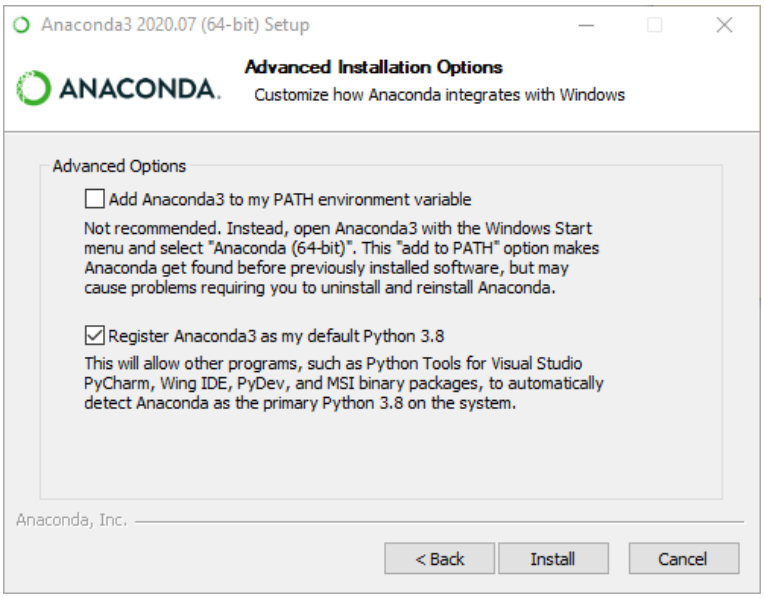
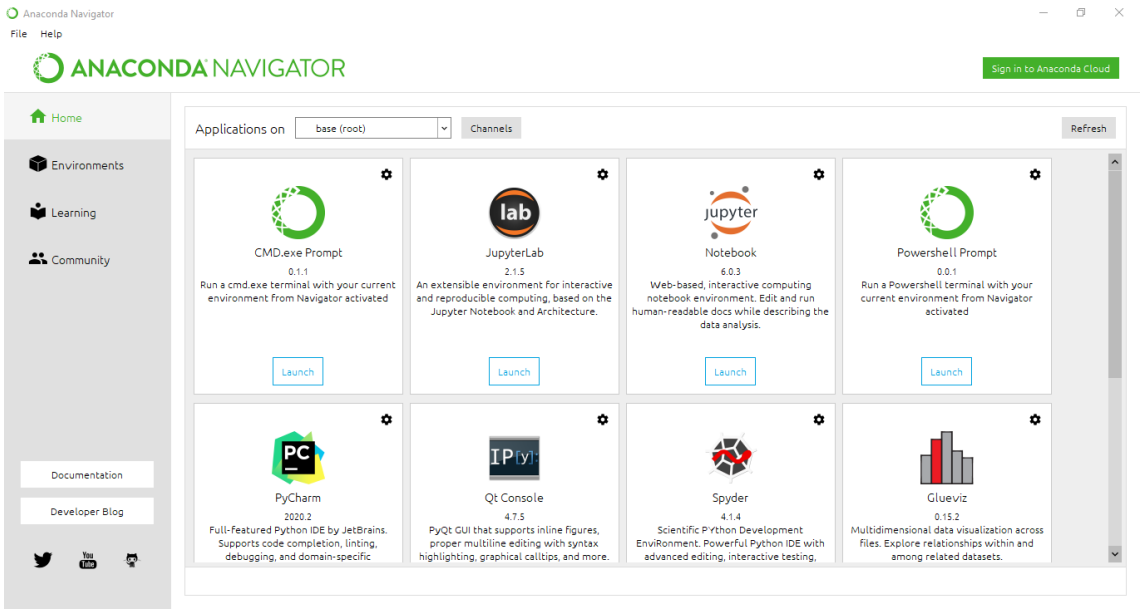


Figura B.5: Primera pantalla de instalación de Anaconda.

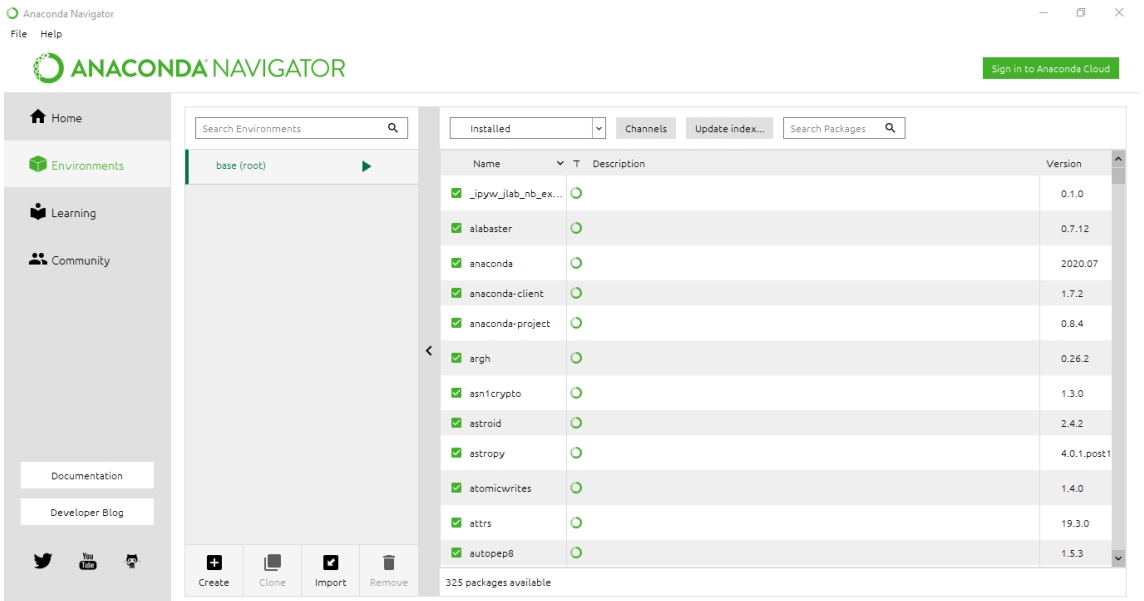
El siguiente paso en la configuración es establecer Anaconda como intérprete por defecto de Python. Esto enlaza la versión de Python con Anaconda e instala herramientas para distintos IDE.



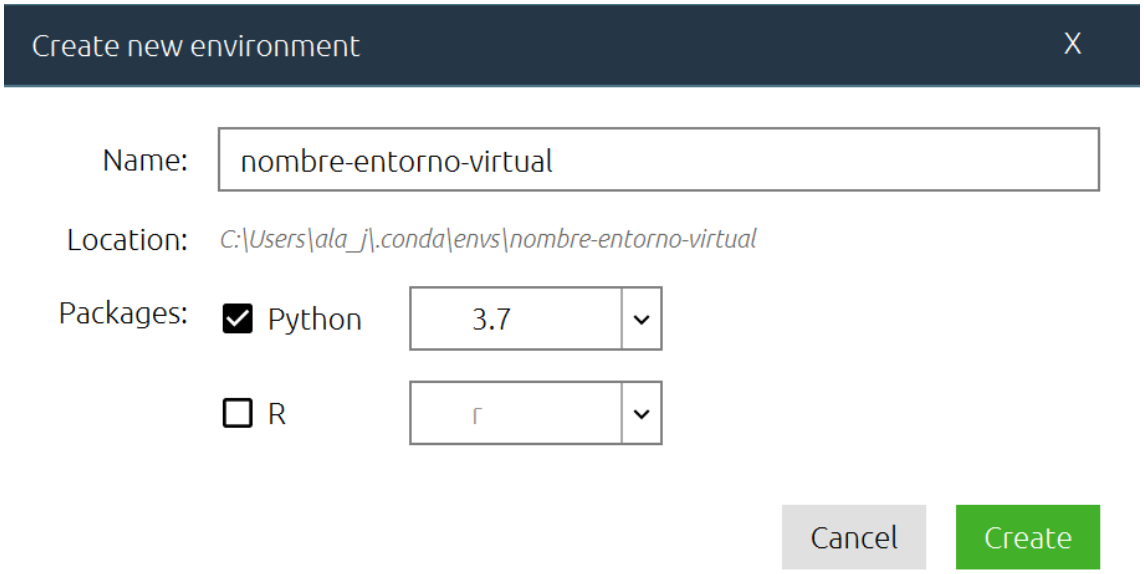
Al hacer click sobre el botón *Install* comienza el proceso de instalación de Anaconda. Cuando ha finalizado la instalación se procede a ejecutar el programa de Anaconda Navigator que es un entorno visual para Anaconda. Al iniciar esta aplicación se observa un pantalla con un menú lateral. En este menú buscamos la opción *Enviroments*, en esta opción se nos permite crear los entornos virtuales.



Cuando se abre la pantalla de los entornos virtuales se observa que se existe uno con el nombre de *base*, que es el entorno virtual básico creado automáticamente por Anaconda. Es necesario crear un entorno virtual para poder separar los paquetes instalados para el proyecto.

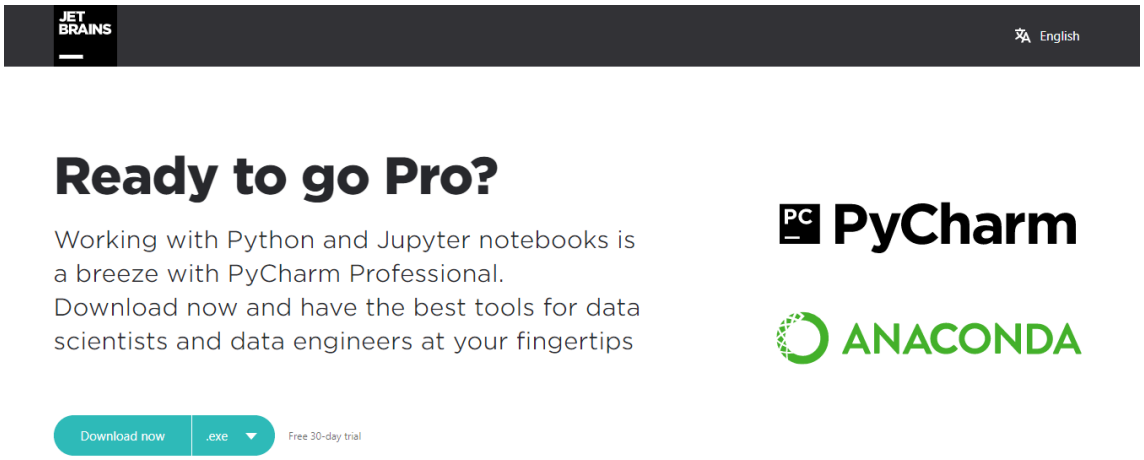


Para crear un nuevo entorno virtual se pulsa el botón *Create*. Al pulsar sobre este botón se abre una ventana emergente en el que hay que realizar cierta opciones sobre el entorno virtual.

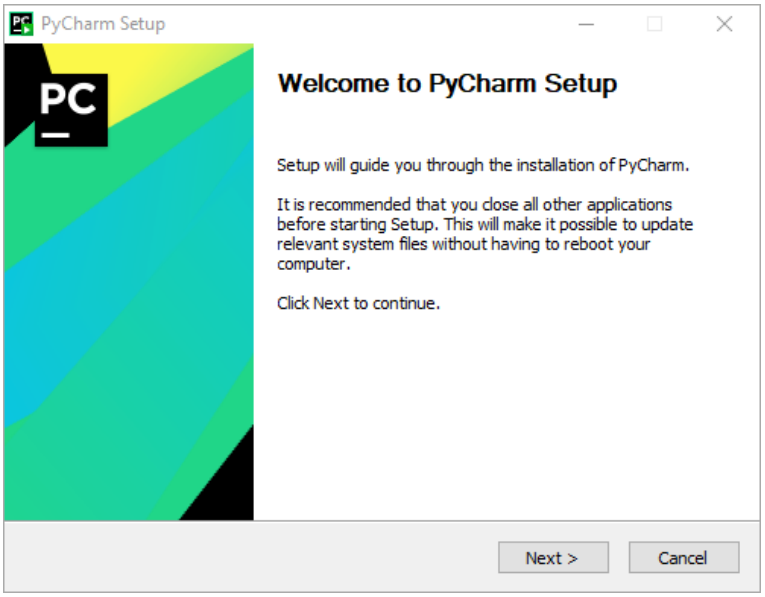


En esta ventana hay que configurar el entorno virtual asignándole la versión de Python en el que se va a crear. El nombre asignado al entorno es importante, ya que permitirá identificar el entorno a la hora de asignarlo al proyecto. Al pulsar sobre el botón *Create* se creará el entorno virtual.

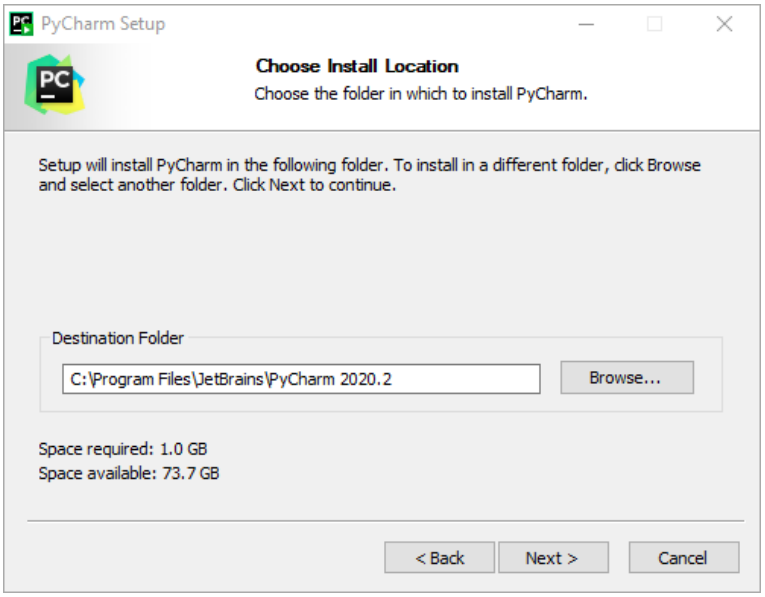
El próximo paso es instalar PyCharm, que es el IDE que se utiliza para la realización y ejecución de la API. Debemos instalar una versión que tiene un plugin de Anaconda. Para descargar esta versión accedemos a la página de descargas de PyCharm. En esta página se descarga la versión profesional de PyCharm, para el desarrollo de este proyecto se utiliza una licencia proporcionada por ser alumno de la UMA.



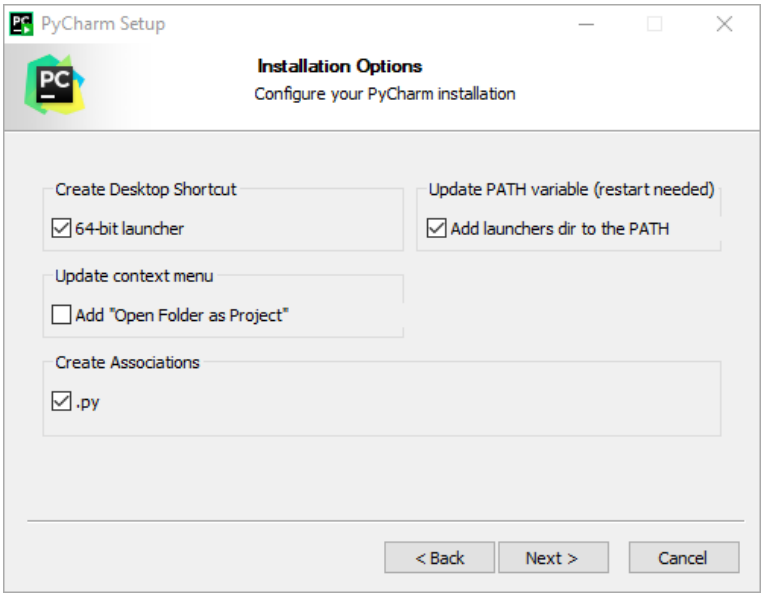
En esta página al principio se encuentra la sección que se muestra en la figura superior. Pulsamos sobre el botón *Download* comienza la descarga del instalador. Cuando se ha descargado lo ejecutamos.



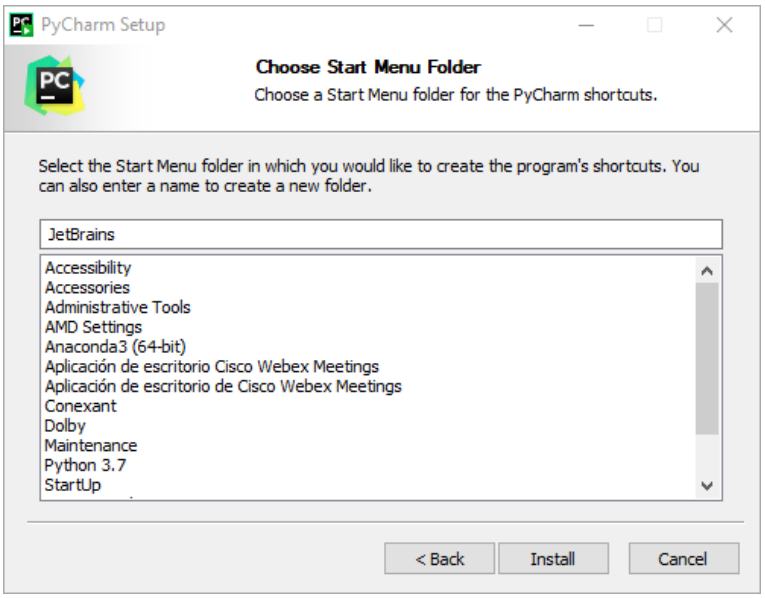
En esta pantalla se nos muestra información general sobre la instalación. Al pulsar sobre el botón *Next* pasamos a la siguiente pantalla de instalación.



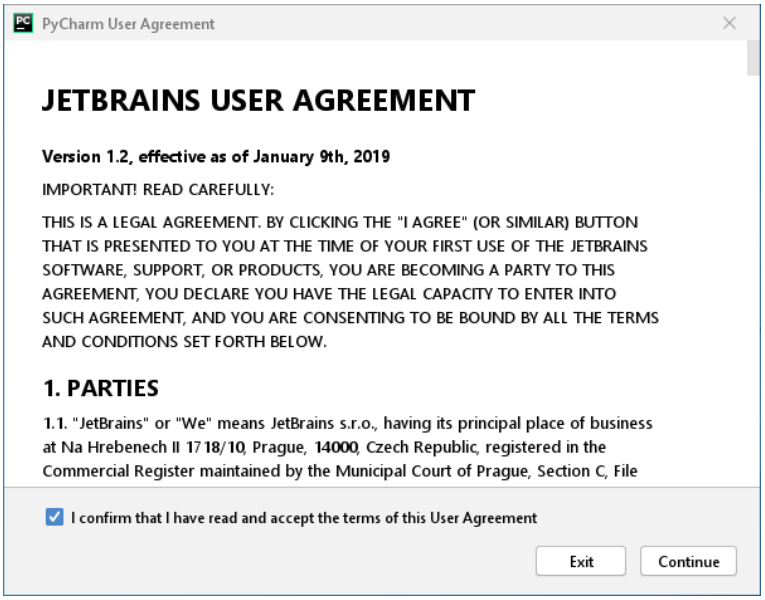
En esta pantalla debemos seleccionar la ruta en la que se instalará el software. Una vez seleccionada la carpeta de instalación pulsamos sobre el botón *Next* para continuar con la instalación



En esta pantalla se configuran ciertos parámetros de la instalación. Es importante seleccionar la opción para añadir la ruta de PyCharm a las variables de entorno del sistema, cosa que necesita de reiniciar el equipo una vez que se complete la Instalación. También es recomendable marcar la opción para asociar la extensión PY a PyCharm. Al hacer click en el botón *Next* continuamos a la siguiente pantalla de instalación.

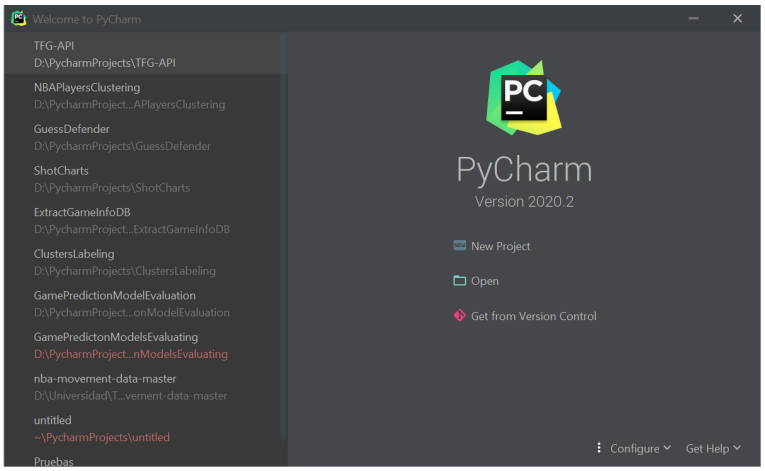


En esta pantalla solo hay que seleccionar la carpeta del menú en la que añadir PyCharm. Al pulsar sobre el botón *Install* comienza la instalación. Una vez acabe la instalación y se reinicie el equipo es necesario ejecutar PyCharm.

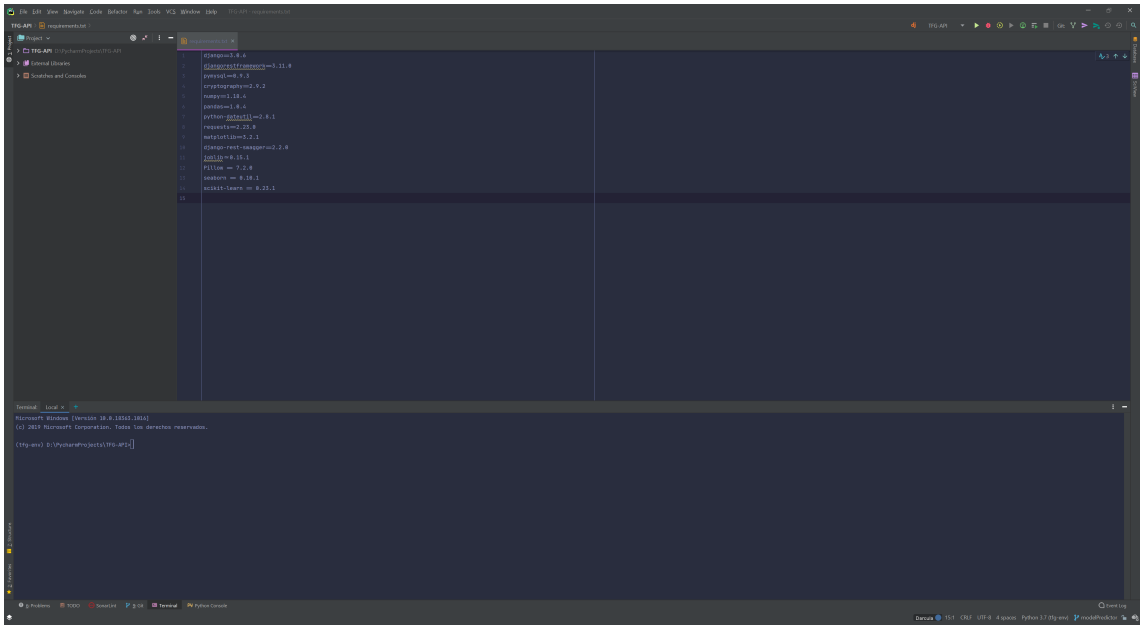


Al iniciar PyCharm por primera vez es necesario aceptar las licencias de Jetbrains. Cuando son aceptadas pide activar una clave, esto no es necesario ya que podemos realizar una prueba gratuita durante un mes.

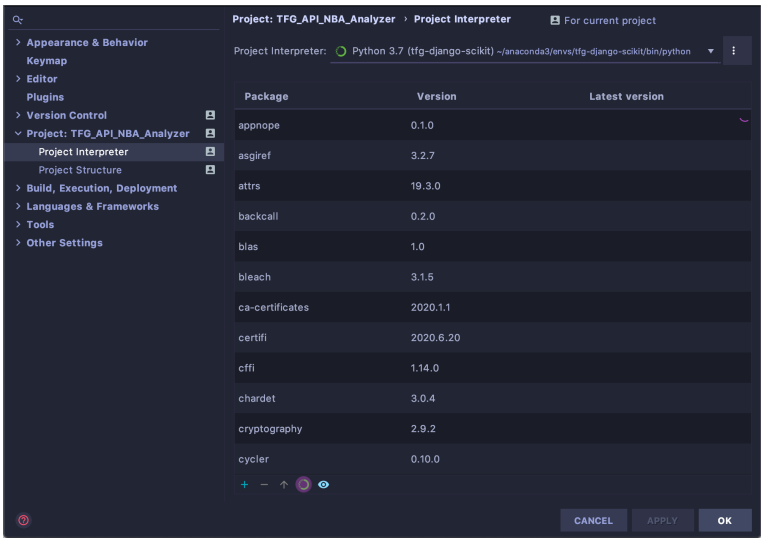
Para abrir el proyecto de la API es necesario haber descargado el código fuente y descomprimirlo. En la pantalla de PyCharm encontraremos un opción con el nombre *open* que nos abrirá un explorador de archivos que nos permite seleccionar la carpeta del proyecto.



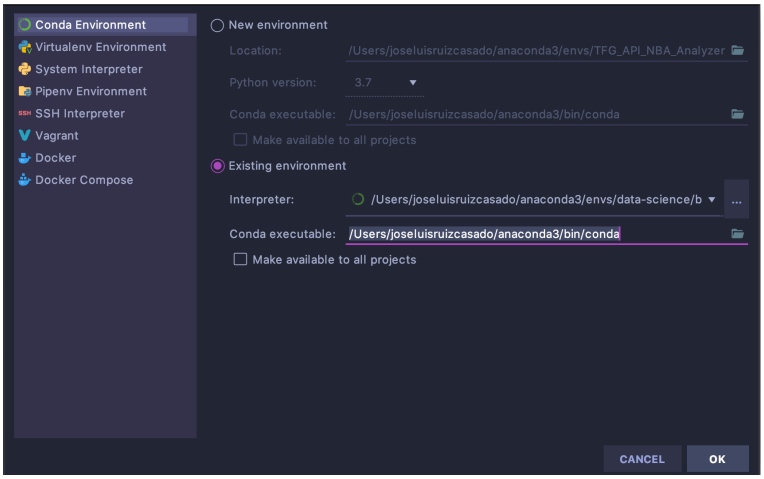
Cuando el proyecto termine de cargar aparece la siguiente pantalla:



El primer paso que debemos realizar es elegir como intérprete del proyecto el entorno virtual creado en Anaconda. Para ello hay que pulsar sobre la opción *File* y luego sobre *settings*. En estas opciones buscamos *Project Interpeter*, y veremos la siguiente pantalla:



Es necesario hacer click a los tres puntos verticales y pulsar sobre las opción *Add*. Esto nos muestra la siguiente ventana:



En esta ventana seleccionamos en el menú la opción de *Conda Enviroment*. Desde esta ventana nos permite crear un nuevo entorno virtual, también es necesario tener Anaconda instalado previamente, o seleccionar un entorno existente. Al seleccionar la opción para añadir un entorno existente en el desplegable aparecerán todos los entornos disponibles, de los cuales seleccionamos el que se ha creado previamente.

Una vez seleccionado el entorno es necesario añadir las dependencias. Las dependencias del proyecto están enumeradas en el fichero *requirements.txt*, en el que también se definen las versiones de cada librería. Para instalar las dependencias abrimos el terminal de Python asociado al proyecto desde PyCharm y ejecutamos el siguiente comando:

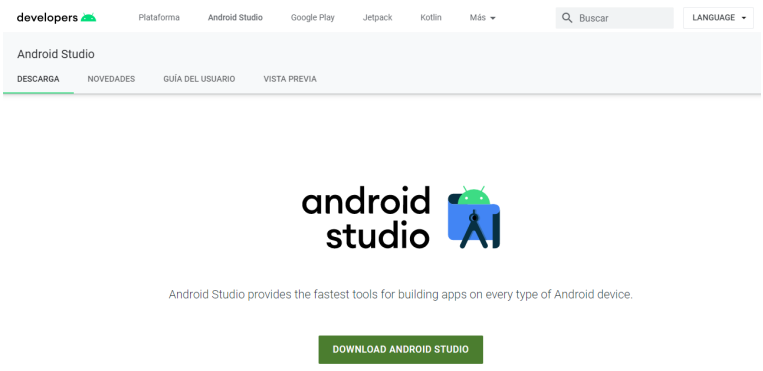
```
pip install -r requirements.txt
```

Una vez que acaben de instalarse las dependencias para que se ejecute la API hay que utilizar el comando:

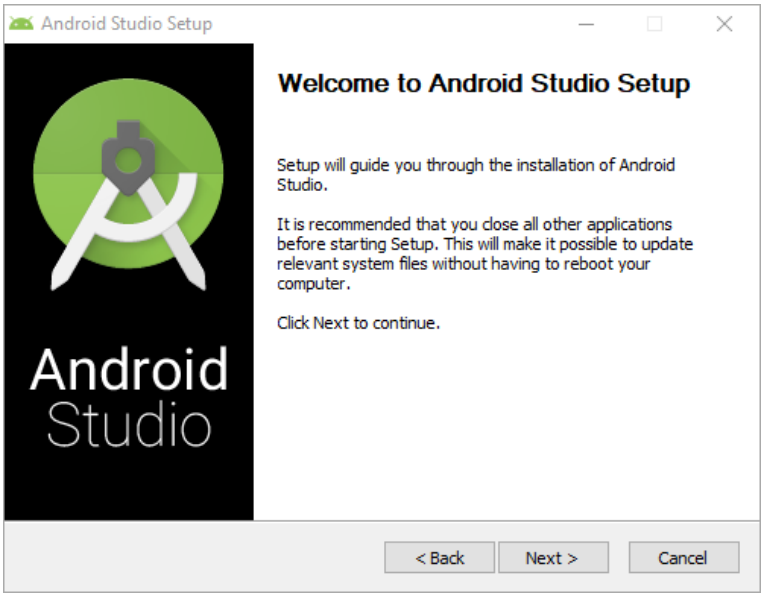
```
python manage.py runserver
```

Instalación Android Studio

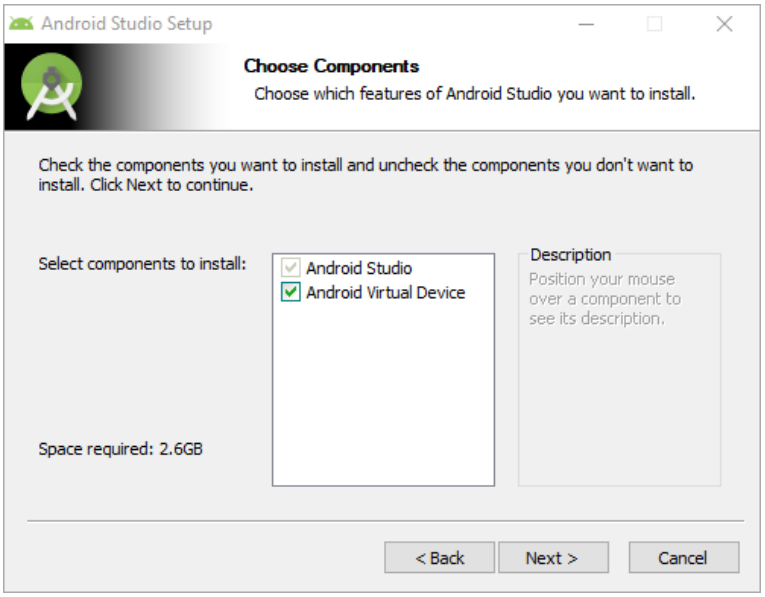
Para poder ejecutar el proyecto con la aplicación Android es necesario descargar Android Studio. Durante el proceso de instalación de este IDE también se instala el sdk de Android y un dispositivo virtual que permite la ejecución de la aplicación en el equipo. La página de descarga tiene el siguiente aspecto:



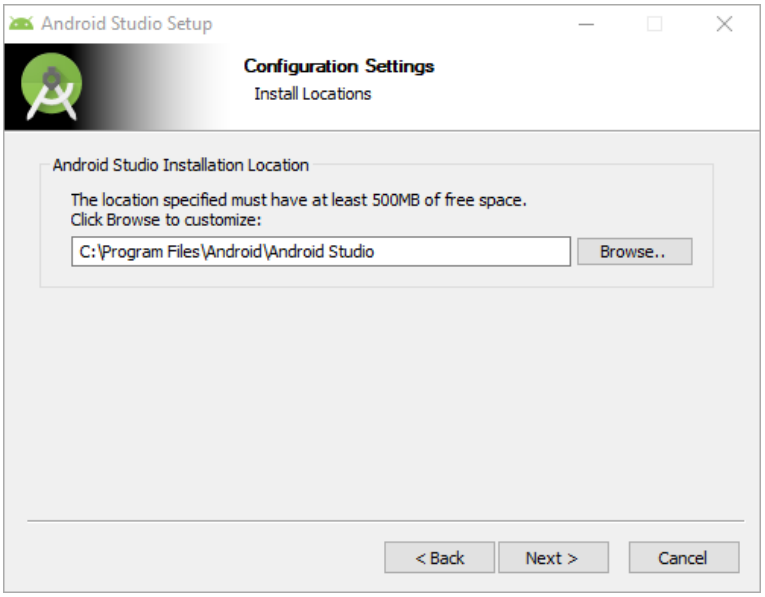
Al pulsar sobre el botón central de la página comienza la descarga del instalador. Cuando ejecutamos el instalador se inicia y nos muestra una ventana con información general sobre Android Studio.



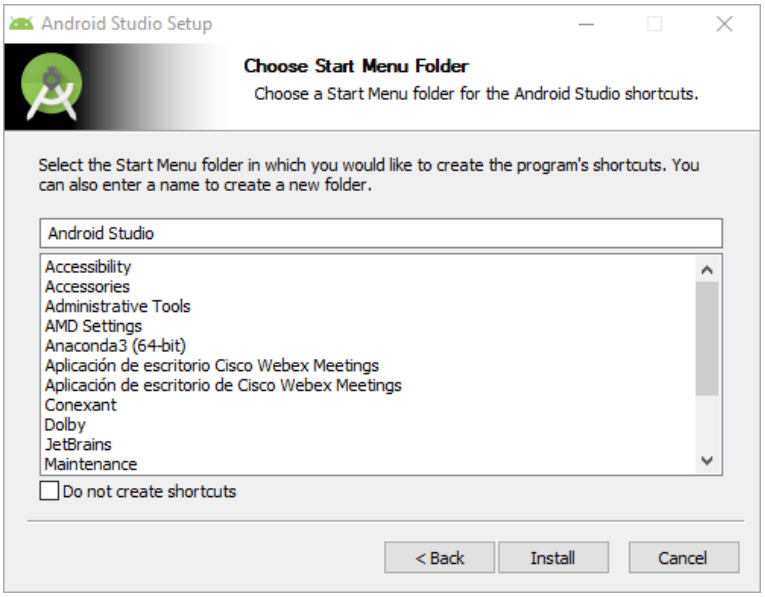
Para comenzar con el proceso de instalación hay que pulsar el botón *Next*. La siguiente ventana que nos aparece nos permite seleccionar qué elementos queremos instalar.



Es importante asegurarse que está marcada la opción de *Android Virtual Device*, que instala el dispositivo virtual en el que se ejecutará la aplicación. Si no se selecciona se debe conectar un dispositivo Android con las opciones de desarrollador y la depuración USB activadas para poder ejecutar la aplicación. Cuando seleccionemos las dos casillas pulsamos el botón *Next*.

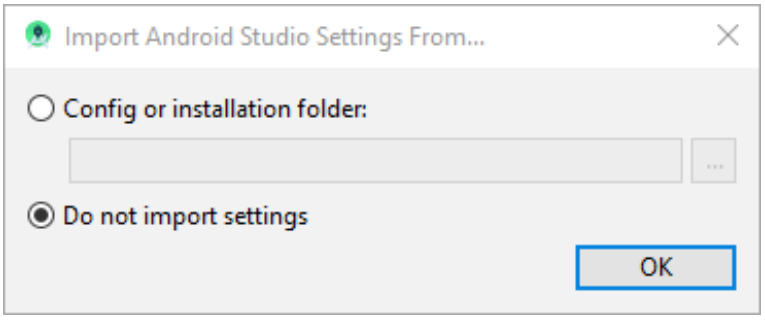


En la siguiente ventana se selecciona la ruta de instalación de Android Studio. Una vez seleccionada la ruta pulsamos sobre el botón *Next* para continuar con la instalación.

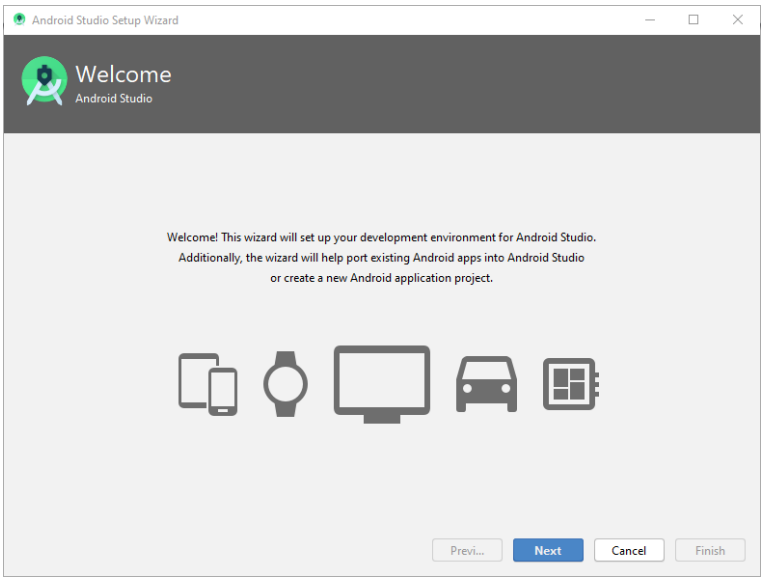


La siguiente ventana que aparece es en la que se selecciona la carpeta del menú de inicio en la que se guardan los datos de Android Studio. Cuando se ha seleccionado una hay que hace click sobre el botón *Next*, y comenzará el proceso de instalación.

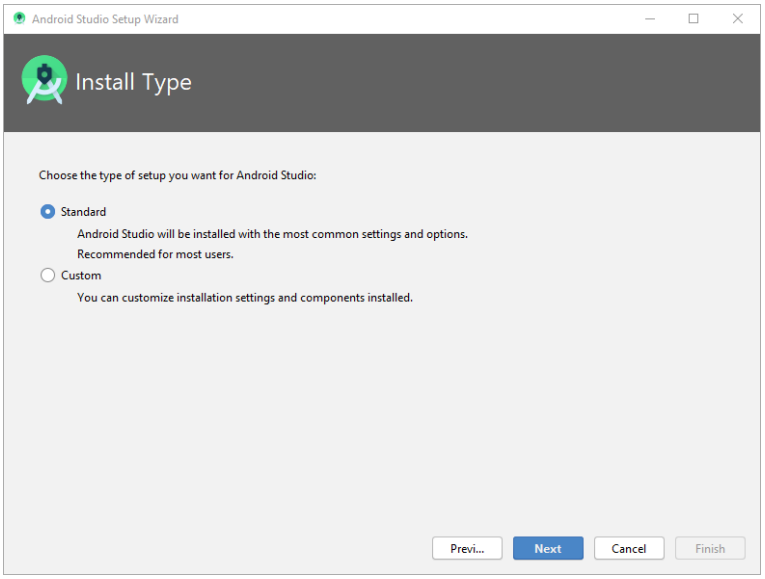
Una vez la instalación haya finalizado debemos iniciar Android Studio, si es la primera vez que se inicia nos aparecerá la siguiente ventana:



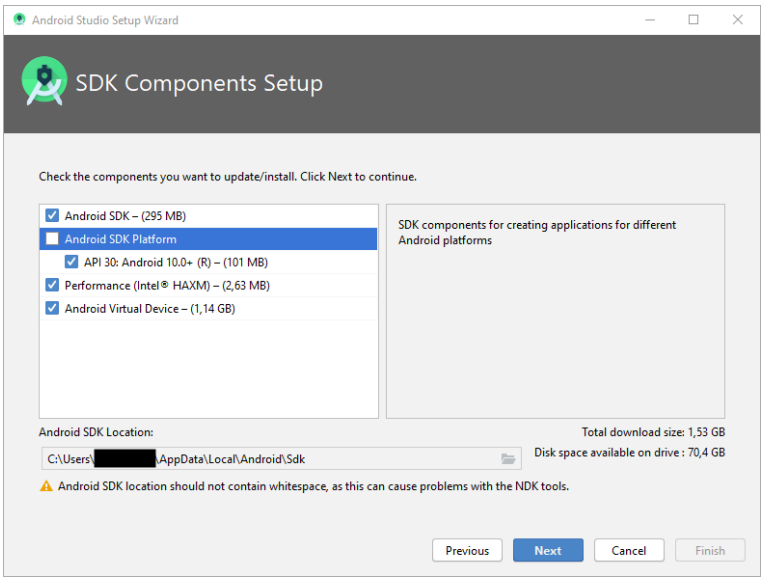
Seleccionamos la opción de no importar configuración, ya que si es la primera vez que se instala no habrá ninguna configuración disponible en el equipo. Al pulsar sobre el botón OK empezará el proceso de configuración de Android Studio.



En esta venta inicial solo sirve para informar de que va a comenzar el proceso de configuración. Por lo que pulsamos sobre el botón *Next*.

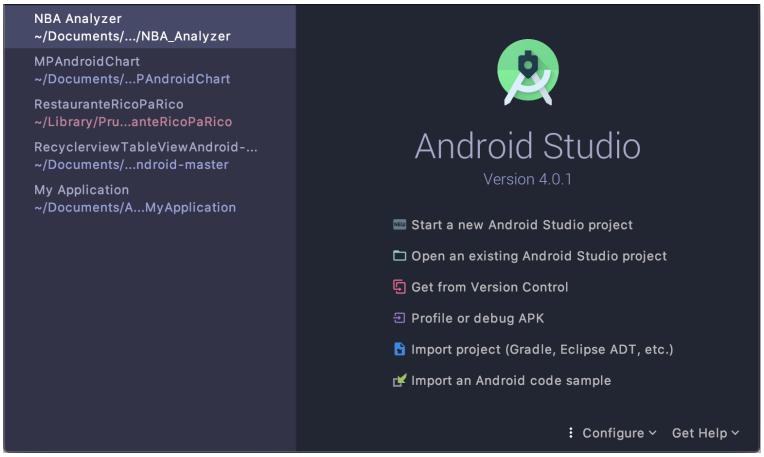


En esta ventana se muestran los dos tipos de configuración que se ofrecen. Una personalizada o una estándar, para este proyecto es suficiente con la configuración estándar. Al seleccionar la configuración se hace click sobre el botón *Next* y se nos muestra una ventana con los componentes que se van a instalar. Los componentes que se deben instalar para el correcto funcionamiento de este proyecto son:



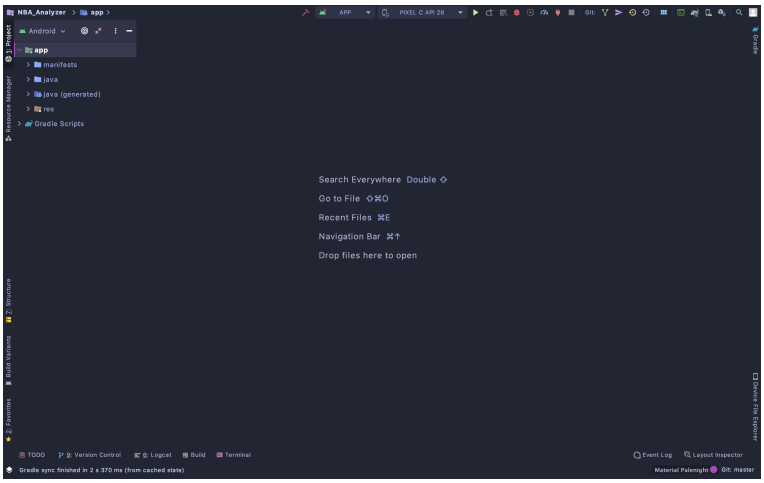
Es importante que en la ruta de instalación del sdk no aparezcan espacios, ya que esto puede causar problemas en algunas librerías de Android. Al pulsar sobre el botón *Next* comienza la instalación.

Cuando la instalación haya acabado nos aparecerá la siguiente ventana:

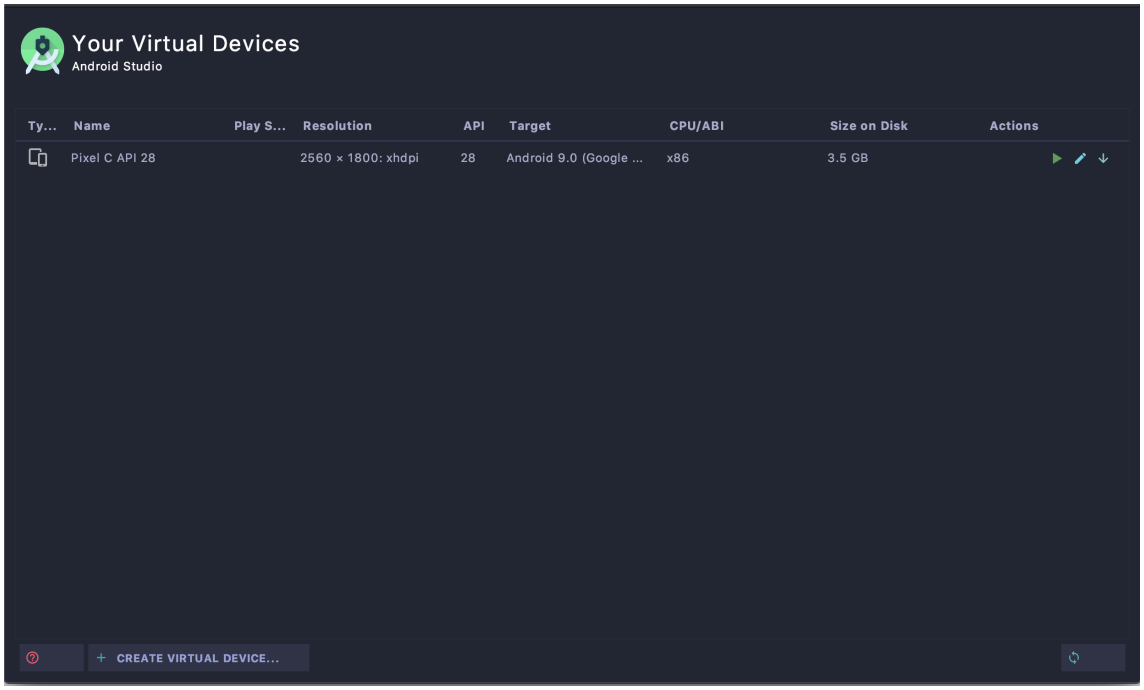


Para abrir el proyecto con la aplicación Android pulsamos sobre la opción *Open an existing Android Studio project*. Esto abrirá una pestaña con el explorador de archivos del equipo, para abrir el proyecto buscamos el directorio donde ha sido extraído el proyecto.

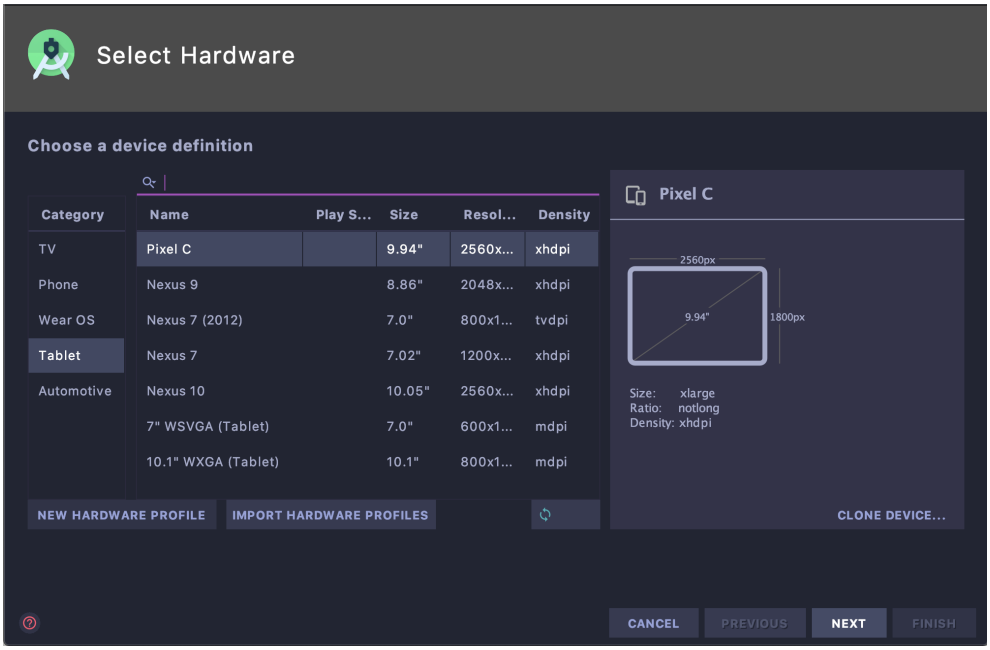
Cuando se abre el proyecto nos aparece una ventana como la siguiente:



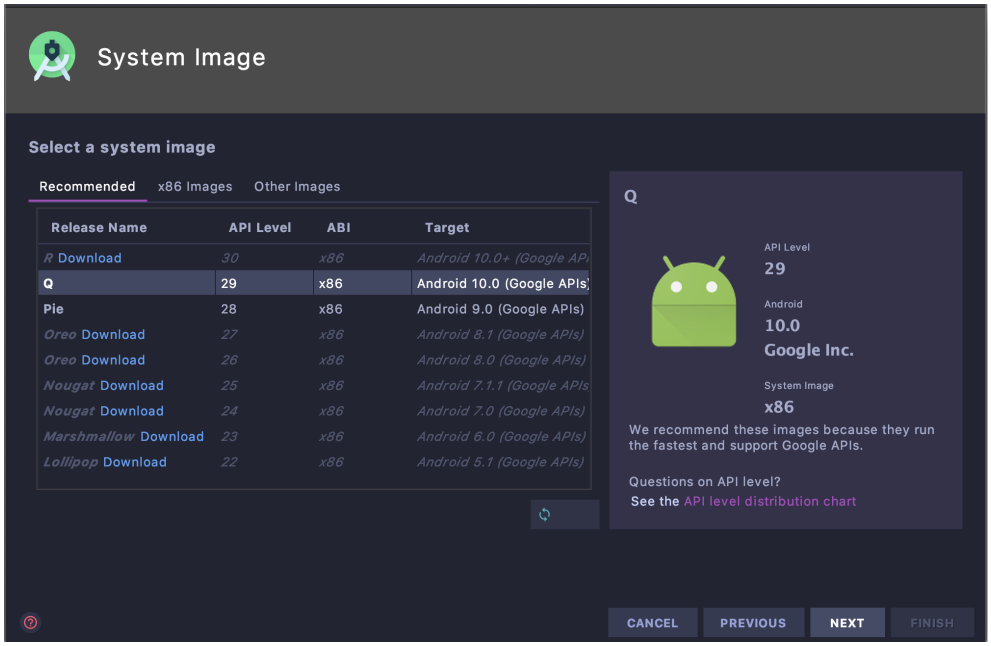
Antes de proceder a la ejecución de la aplicación es necesario descargar e instalar un dispositivo virtual. Para ello pulsamos sobre el botón con forma de móvil que hay en la esquina superior derecha. Esto abre el panel de control de los dispositivos virtuales y nos muestra una ventana como ésta:



Si nunca se ha instalado un dispositivo virtual en el equipo no aparecerá nada en la lista. Para añadir un nuevo dispositivo pulsamos en el botón situado en la parte inferior izquierda en el que aparece el símbolo +.

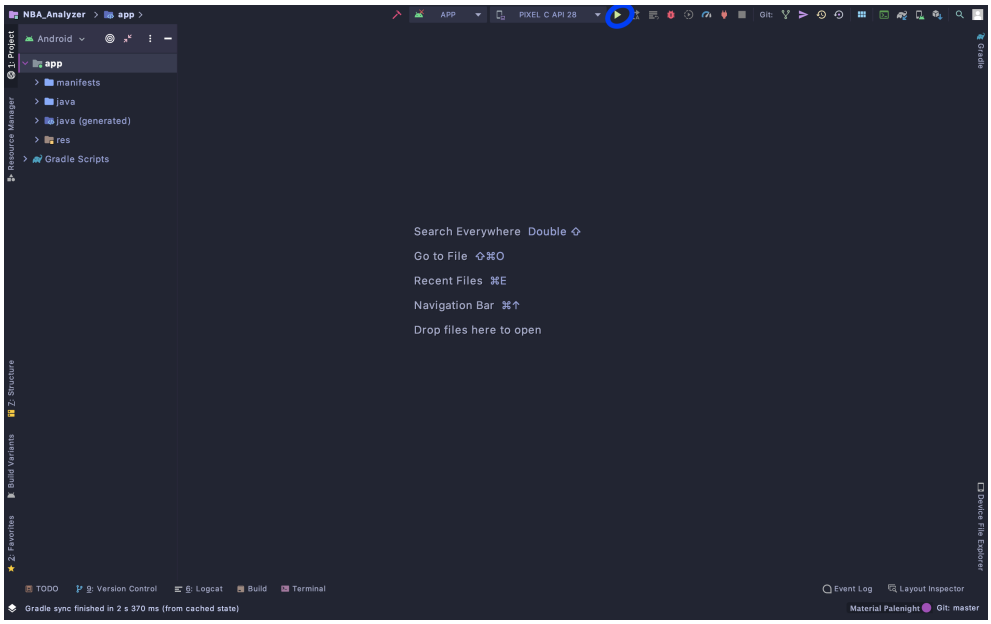


En la ventana que nos aparece seleccionamos en el menú lateral la opción de *Tablets* y seleccionamos el dispositivo Pixel C, ya que la aplicación está pensada para su ejecución en tablets. Pulsamos sobre el botón *Next* y veremos una ventana con las distintas versiones de Android que se pueden instalar en el dispositivo virtual.



Para poder ejecutar el proyecto la versión que debemos seleccionar es Android Q. Al pulsar sobre el botón *Next* se muestra la información sobre el dispositivo que se va a instalar, para comenzar la instalación pulsamos sobre el botón *Finish* y comenzará la instalación.

Una vez que se haya instalado el dispositivo virtual volveremos a ver la siguiente ventana:



Para ejecutar la aplicación solo debemos pulsar sobre el botón marcado en la imagen superior.