

NBAPP

Autor: Cristian García Lagar
Tutor: Raquel Cerdá Losa
Ciclo Formativo: DAM
Instituto: UNIR

Índice general

1. Definiendo el proyecto	4
1.1. Motivación	4
1.2. Objetivo del proyecto.....	4
2. Palabras clave	6
3. Módulos formativos aplicados en el trabajo.	8
4. Herramientas/Lenguajes utilizados.....	9
4.1. Entorno de desarrollo: Android Studio	9
4.2. Lenguaje de programación: Java	9
4.3. Servicios REST.....	10
4.4. LibreriaVolley	10
4.5. API	11
4.6. Sistemas de control de versiones: GIT y GITHUB	12
4.7. Herramienta de construcción de proyectos: Gradle	13
5. Comenzando con el proyecto.....	15
5.1. Buscando fuentes de datos	15
5.2. Organizando el proyecto	17
5.2.1. Diagrama de casos de uso	17
5.2.2. Diagrama de clases	19
6. Diseñando el proyecto	21
6.1. Diseño de las interfaces	21
6.2. Arquitectura Modelo-Vista-Controlador (MVC).....	22
7. Desarrollando el proyecto.....	25
7.1. MainActivity	25
7.2. PlayerListActivity	27
7.3. StatsActivity.....	30
7.4. TeamActivity.....	34
7.5. InfoPlayerActivity.....	36
7.6. TeamInfoActivity	37
7.7. MapActivity	40

7.8. GameActivity	41
8. Bibliografía	44

1. Definiendo el proyecto

1.1. Motivación

Fue un 1 de Noviembre de 1946, todavía bajo el nombre de BAA, Basketball Association of America, y fuera del territorio estadounidense cuando se jugó el primer partido de lo que conocemos en la actualidad como la NBA.

Durante las décadas de 1990 y 2000, la NBA experimentó un crecimiento significativo en términos de popularidad y número de espectadores. A medida que la liga se expandía y surgían nuevas estrellas, más personas comenzaron a seguir el baloncesto profesional en todo el mundo.

En general, el crecimiento de espectadores de la NBA en los años 90 y 2000 refleja el aumento de interés y la creciente popularidad del baloncesto profesional en todo el mundo.

La expansión de la NBA por el mundo gracias a los derechos televisivos ha sido un factor clave en el crecimiento global de la liga. La transmisión de los partidos de la NBA a nivel internacional ha permitido que millones de personas en diferentes países puedan seguir y disfrutar del baloncesto profesional.

A lo largo de las décadas de 1990 y 2000, la NBA trabajó arduamente para expandir su alcance televisivo a nivel global, lo que permitió la difusión de los partidos de la NBA en todo el mundo. Esto ha ayudado a popularizar el baloncesto en diferentes países, generando un creciente interés en el deporte y atrayendo a nuevos seguidores.

La expansión de la NBA ha permitido que en España podamos disfrutarla y me han permitido seguir los campeonatos, por lo que, para este trabajo fin de ciclo, quería desarrollar una aplicación que permita consultar datos estadísticos de esta liga.

1.2. Objetivo del proyecto

La NBA es conocida por recopilar y analizar una amplia variedad de estadísticas para medir el rendimiento de los jugadores, equipos y la liga en general. Estas estadísticas proporcionan información detallada sobre diferentes aspectos del juego y son utilizadas por analistas, entrenadores, jugadores y aficionados para evaluar el desempeño y comparar a los jugadores y equipos.

Algunas de las estadísticas más comunes utilizadas en la NBA incluyen:

Puntos: La estadística más básica y ampliamente reconocida en el baloncesto es el número de puntos anotados por un jugador o un equipo durante un partido o una temporada.

- Rebotes: Mide el número de veces que un jugador obtiene el balón después de un tiro fallido. Se divide en rebotes ofensivos (obtenidos por el equipo atacante) y rebotes defensivos (obtenidos por el equipo defensor).
- Asistencias: Mide el número de pases realizados por un jugador que resultan en una canasta de un compañero de equipo.
- Robos: Mide el número de veces que un jugador arrebata el balón a un oponente.
- Tapones: Mide el número de veces que un jugador bloquea un tiro de un oponente.
- Porcentaje de tiros: Calcula el porcentaje de tiros realizados con éxito por un jugador, ya sea de dos puntos, tres puntos o tiros libres.
- Eficiencia: Se utiliza para evaluar la efectividad general de un jugador. La eficiencia toma en cuenta el número de puntos, rebotes, asistencias, robos y tapones, así como los tiros fallados y las pérdidas de balón.

Estas son solo algunas de las estadísticas más comunes, pero la NBA recopila y analiza una amplia gama de datos más avanzados, como el porcentaje de uso, índices de eficiencia avanzada, porcentaje de tiros de campo efectivo, entre otros. Estas estadísticas avanzadas proporcionan una visión más completa y detallada del desempeño de los jugadores y equipos.

Las estadísticas en la NBA desempeñan un papel fundamental en la comprensión y análisis del juego, y son una parte integral del disfrute y la apreciación del baloncesto profesional.

El objetivo es desarrollar una aplicación para dispositivos móviles con el sistema operativo Android que permita al usuario consultar:

- Estadísticas de los jugadores de la NBA en la temporada 2022-2023.
- Resultados de los partidos jugados durante dicha temporada.
- Información sobre cada franquicia de la NBA.

2. Palabras clave

En este apartado se van a describir los distintos elementos de software utilizados durante el desarrollo de la aplicación.

- **Activity**: En el contexto de desarrollo de aplicaciones para Android, una "Activity" es un componente crucial de la arquitectura de Android. En términos sencillos, una Activity representa una única pantalla con una interfaz de usuario. Se puede pensar en una Activity como una ventana en la cual los usuarios pueden realizar acciones, interactuar con los elementos de la interfaz y recibir información.
- **View**: Esta clase representa el bloque básico para los componentes de la interfaz de usuario. Una View ocupa un área rectangular en la pantalla y es responsable de dibujar y manejar eventos. View es la clase base para widgets, que se utilizan para crear componentes interactivos de la interfaz de usuario (botones, campos de texto, etc.).
- **Fragment**: Un Fragment representa una parte reutilizable de la IU de tu app. Un fragmento define y administra su propio diseño, tiene su propio ciclo de vida y puede administrar sus propios eventos de entrada. Los fragmentos no pueden existir por sí solos. Deben estar alojados por una actividad u otro fragmento. La jerarquía de vistas del fragmento forma parte de la jerarquía de vistas del host o está conectada a ella.
- **ConstraintLayout**: Un ConstraintLayout es un android.view.ViewGroup que te permite posicionar y dimensionar widgets de una manera flexible.
- **FrameLayout**: Es un tipo de ViewGroup en Android diseñado para ser un administrador de diseño simple que contiene una única vista secundaria. Es un contenedor liviano que ocupa un área rectangular en la pantalla. La característica clave de FrameLayout es que permite que las vistas secundarias se apilen una encima de la otra, siendo la última vista añadida visualmente la que se encuentra en la parte superior.
- **CoordinatorLayout**: Es otro tipo de ViewGroup en Android que ofrece funcionalidades más avanzadas en comparación con FrameLayout. Se utiliza para coordinar interacciones complejas entre vistas secundarias, proporcionando un marco flexible para la implementación de comportamientos específicos.

- **LinearLayout**: Un diseño que organiza otras vistas ya sea horizontalmente en una sola columna o verticalmente en una sola fila.
- **TabLayout**: Layout que proporciona un diseño horizontal para mostrar pestañas.
- **ViewPager2**: Un administrador de diseño que permite al usuario pasar páginas de datos hacia la izquierda y la derecha.
- **CardView**: Un CardView es un FrameLayout con un fondo de esquina redondeada y sombra.
- **ImageView**: Muestra recursos de imágenes, como Bitmap o recursos Drawable. ImageView también se utiliza comúnmente para aplicar tintes a una imagen y gestionar el escalado de imágenes.
- **Button**: Un elemento de interfaz de usuario que el usuario puede tocar o hacer clic para realizar una acción.
- **EditText**: Un elemento de interfaz de usuario para introducir y modificar texto.
- **TextView**: Un elemento de interfaz de usuario que muestra un texto.
- **RecyclerView**: Una vista flexible para proporcionar una ventana limitada a un conjunto de datos extenso.
- **CalendarView**: Esta clase es un widget de calendario para mostrar y seleccionar fechas.
- **Toast**: Un "toast" es una vista que contiene un mensaje breve para el usuario. La clase Toast ayuda a crear y mostrar estos mensajes de manera sencilla.

3. Módulos formativos aplicados en el trabajo.

El módulo formativo principal sobre el que se apoyará el trabajo fin de ciclo es el de **Programación Multimedia y dispositivos móviles** ya que es el que ha profundizado en el desarrollo de una aplicación móvil tanto a nivel de backend como de frontend en entorno Android.

No obstante de forma indirecta, se hará uso de los aprendido en distintas asignaturas tanto para el desarrollo de la app como para la elaboración de la memoria. A continuación detallan las misma:

- **Programación:** Los fundamentos adquiridos en esta asignatura son en los que se cimienta este ciclo, ya que ha transmitido conocimiento sobre el lenguajes de programación utilizado, concretamente Java, sobre programación orientada a objetos, la aplicación de lógica de programación, diseño de arquitectura del software, etc.
- **Entornos de desarrollo:** En el desarrollo de la aplicación se han utilizado tres elementos estudiados en la asignatura, como son un Entorno de Desarrollo Integrado (IDE), un sistema de control de versiones y el estándar utilizado para el modelado de visual de software denominado Lenguaje de Modelado Unificado (UML).

Para desarrollar el proyecto se ha utilizado Android Studio como IDE, git se ha usado como herramienta de control de versiones y github como plataforma para alojar el repositorio git para almacenar el código fuente a entregar, por último, en este documento se han incluido un diagrama de uso y de clases para representar el diseño, arquitectura y la implementación del software.

- **Lenguajes de marcas y sistemas de gestión de la información:** Se han utilizado los conocimientos adquiridos en el intercambio de información. Los datos utilizados por la aplicación se obtiene en formato JavaScript Object Notation (JSON), desde las APIs de Balldontlie y SportsDataIOS
- **Programación de servicios y procesos:** Se utilizan servicios REST para conectar las peticiones realizadas por la aplicación empleando la librería HTTP denominada Volley, de la que se hablará en el siguiente apartado.
- **Acceso a datos:** La información obtenida es mapeada a objetos de clases java.

4. Herramientas/Lenguajes utilizados.

4.1. Entorno de desarrollo: Android Studio

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.

¿Qué ofrece Android Studio?

- Un entorno de desarrollo claro y robusto.
- Facilidad para testear el funcionamiento en otros tipos de dispositivos.
- Asistentes y plantillas para los elementos comunes de programación en Android.
- Un completo editor con muchas herramientas extra para agilizar el desarrollo de nuestras aplicaciones.



Fuente: <https://android-developers.googleblog.com/2023/05/android-studio-io-23-announcing-studio-bot.html>

4.2. Lenguaje de programación: Java

Java es una plataforma de software y un lenguaje de programación orientado a objetos ampliamente utilizado que se ejecuta en miles de millones de dispositivos, incluidos ordenadores portátiles, dispositivos móviles, consolas de videojuegos y dispositivos médicos, entre muchos otros. Las reglas y la sintaxis de Java se basan en los lenguajes C y C++.

Una de las principales ventajas de desarrollar software con Java es su portabilidad. Una vez escrito el código de un programa Java en un ordenador portátil, es muy fácil trasladarlo a un dispositivo móvil. Cuando James Gosling, de Sun Microsystems (más

tarde adquirido por Oracle), inventó este lenguaje en 1991, su objetivo principal era poder "escribir una vez, ejecutar en cualquier lugar".



Fuente: <https://www.genbeta.com/desarrollo/java-el-lenguaje-mas-usado-y-su-evolucion>

Para crear una aplicación mediante Java, debe descargar el Kit de desarrollo de Java (JDK), que está disponible para Windows, macOS y Linux. Se escribe el programa en el lenguaje de programación Java, y luego un compilador convierte el programa a un código de bytes de Java, el conjunto de instrucciones para Java Virtual Machine (JVM) que forma parte del entorno de tiempo de ejecución Java (JRE). El código de bytes de Java se ejecuta sin modificaciones en cualquier sistema que admita JVM, de modo que el código Java se puede ejecutar en cualquier lugar.

4.3. Servicios REST

REST es una interfaz para conectar varios sistemas basados en el protocolo HTTP (uno de los protocolos más antiguos) y nos sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON.

El formato más usado en la actualidad es el formato JSON, ya que es más ligero y legible en comparación al formato XML. Elegir uno será cuestión de la lógica y necesidades de cada proyecto.

REST se apoya en HTTP, los verbos que utiliza son exactamente los mismos, con ellos se puede hacer GET, POST, PUT y DELETE. De aquí surge una alternativa a SOAP.

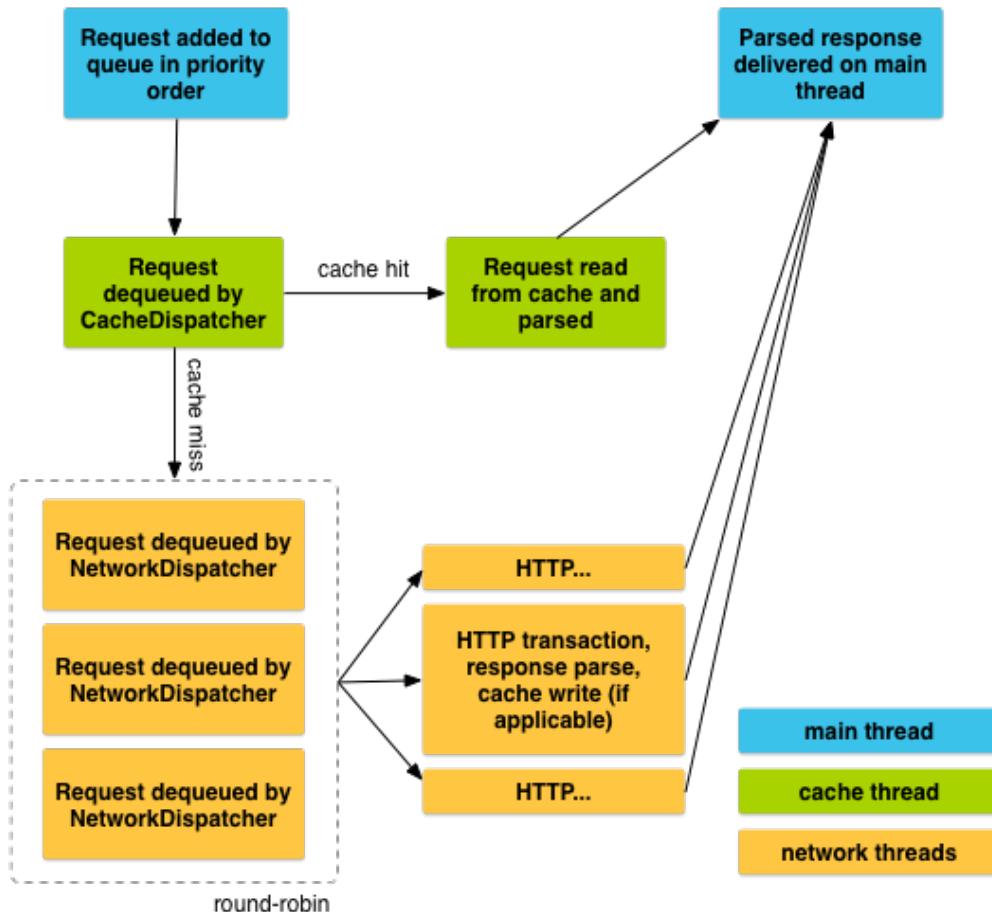
4.4. Librería Volley

Volley es una biblioteca HTTP que facilita y agiliza el uso de redes en apps para Android

Esta librería destaca por sus operaciones de tipo RPC que se usan para completar la IU, por ejemplo, obtener una página de resultados de la búsqueda como datos estructurados.

Se integra fácilmente con cualquier protocolo y, además, incluye compatibilidad con strings sin procesar, imágenes y JSON. Dado que proporciona compatibilidad integrada con las funciones que necesitas, Volley elimina la necesidad de escribir código estándar y te permite concentrarte en la lógica que es específica de tu app.

La siguiente imagen muestra la arquitectura de su funcionamiento.



Fuente <https://umhandroid.momrach.es/volley-conexiones-http-eficientes-y-configurables/>

4.5. API

El término API es una abreviatura de Application Programming Interfaces, que en español significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

Así pues, se puede hablar de una API como una especificación formal que establece cómo un módulo de un software se comunica o interactúa con otro para cumplir una o muchas funciones. Todo dependiendo de las aplicaciones que las vayan a utilizar, y de los permisos que les dé el propietario de la API a los desarrolladores de terceros.

Una de las principales funciones de las API es poder facilitarle el trabajo a los desarrolladores y ahorrarles tiempo y dinero. Por ejemplo, si estás creando una aplicación que es una tienda online, no necesitarás crear desde cero un sistema de pagos u otro para verificar si hay stock disponible de un producto. Podrás utilizar la API de un servicio de pago ya existente, por ejemplo PayPal, y pedirle a tu distribuidor una API que te permita saber el stock que ellos tienen.

4.6. Sistemas de control de versiones: GIT y GITHUB

Un sistema de control de versiones o VCS (Version Control System, por sus siglas en inglés), rastrea el historial de cambios conforme las personas y los equipos colaboran juntos en los proyectos. Conforme los desarrolladores hacen cambios al proyecto, cualquier versión anterior de este puede recuperarse en cualquier momento.

Los desarrolladores pueden revisar el historial de proyectos para averiguar:

- ¿Qué cambios se hicieron?
- ¿Quién los hizo?
- ¿Cuándo se hicieron?
- ¿Por qué se necesitaban?

Los VCS le proporcionan a cada contribuyente una vista consistente y unificada de un proyecto, lo cual muestra el trabajo que ya está en progreso. El ver un historial de cambios transparente, quién los hizo y cómo contribuyen al desarrollo de un proyecto, ayuda a que los miembros de los equipos se alineen mientras trabajan independientemente.

Por otro lado, GitHub hospeda repositorios de Git y proporciona a los desarrolladores las herramientas para generar un código mejor mediante características de línea de comandos, propuestas (debates en hilo), solicitudes de cambio, revisión de código o el uso de un conjunto de apps gratuitas y de pago en GitHub Marketplace. Con las capas de colaboración tales como el flujo de GitHub, una comunidad de 100 millones de

desarrolladores y un ecosistema con cientos de integraciones, GitHub cambia la forma en la que se crea el software.

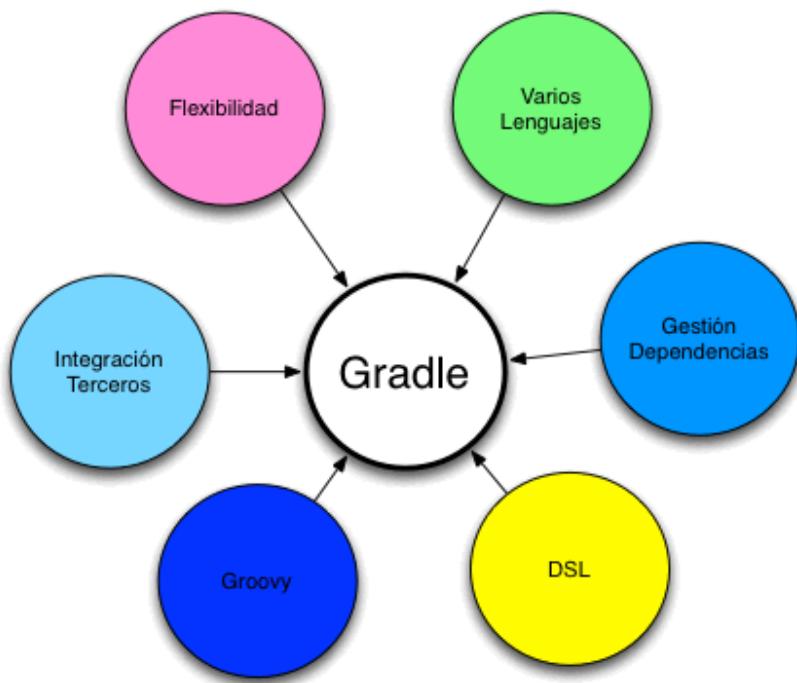
GitHub crea una colaboración directamente en el proceso de desarrollo. El trabajo se organiza en los repositorios donde los desarrolladores pueden describir los requisitos o la dirección y marcar las expectativas para los miembros de los equipos. Posteriormente, utilizando el flujo de GitHub, los desarrolladores simplemente crean una rama para trabajar en las actualizaciones, confirmar cambios para guardarlos, abrir una solicitud de cambios para proponerlos y debatirlos y fusionar solicitudes de cambio una vez que todos estén de acuerdo.



Fuente:<https://www.udemy.com/course/git-and-github-crash-course/>

4.7. Herramienta de construcción de proyectos: Gradle

Gradle es una herramienta de automatización de la construcción de nuestro código que bebe de las aportaciones que han realizado herramientas como ant y maven pero intenta llevarlo todo un paso mas allá. Para empezar se apoya en Groovy y en un DSL (Domain Specific Language) para trabajar con un lenguaje sencillo y claro a la hora de construir el build comparado con Maven. Por otro lado dispone de una gran flexibilidad que permite trabajar con ella utilizando otros lenguajes y no solo Java. Dispone por otro lado de un sistema de gestión de dependencias sólido.



Fuente: <https://www.arquitecturajava.com/que-es-gradle/>

5. Comenzando con el proyecto

Para iniciar el trabajo de fin de ciclo se plantean dos premisas principales: cómo y sobre qué temática desarrollarlo.

Elegir entre las distintas opciones de aplicaciones que se han visto durante el ciclo formativo ha sido el primer paso para el desarrollo de este proyecto. Entre estas se encuentran una pagina web, una aplicación de escritorio, una aplicación móvil o un videojuego. Finalmente, se ha elegido una aplicación móvil, debido a que esta tecnología es la que ha motivado la realización de este plan de estudios.

Una vez que se ha concluido cómo desarrollar el proyecto, se plantean las posibles ideas en las que sustentarlo. Inicialmente, se pensó en una aplicación al estilo del clásico juego Comunio o Fantasy, en la que se pudieran crear ligas y equipos de futbol en los que los jugadores fueran los propios amigos y un administrador asignara los puntos a cada uno en función a su desempeño y posición dentro del campo. Esta idea se descartó debido a que se consideró que el sistema de puntuación sería poco objetivo.

La temática finalmente elegida, también centrada en el deporte, se relaciona con el baloncesto. La idea principal es poder consultar datos estadísticos de los jugadores de la NBA, liga de baloncesto más importante del mundo, información sobre los equipos que la forman y resultados de los partidos, todo ello referido a la temporada 2022-2023. La elección fue motivada, entre otros aspectos, porque se planteaba la posibilidad de trabajar con APIs para la obtención de la información y así ampliar los conocimientos adquiridos en el ciclo formativo.

5.1. Buscando fuentes de datos

Para poder comenzar a desarrollar la aplicación primero había que saber desde dónde se podría obtener los datos que esta se le mostrarían al usuario. A través de Google y repositorios de Github se realizó una intensa búsqueda en la que la mayoría de las APIs que disponían de datos adecuados a los criterios buscado eran de pago. Por este motivo, se ha tenido que seleccionar dos fuentes de datos, de las cuales, se puede extraer toda la información necesaria para cumplir con los objetivos marcados.

- Balldontlie**

La primera API que reunía los condicionantes adecuados para ser utilizada en la obtención de datos es Balldonlie, un proyecto open source que permite acceder a

diferentes datos sobre la NBA como es la información sobre jugadores, equipos, partidos y/o estadísticas.

De esta fuente de información he utilizado las estadísticas promedio de los jugadores durante la temporada 2022-2023, lo que permite cumplir el primer objetivo marcado en el proyecto. Aunque es cierto que dispone de más información que podría ser útil para el proyecto, se ha descartado su uso al localizar la siguiente API, cuyo contenido se ha considerado que se adapta mejor a las necesidades del proyecto.

Las estadísticas se han diferenciado en ataque y defensa, de la siguiente forma:

Ataque			
Puntos totales	Tiros encestados	Tiros intentados	Acierto en tiros (%)
Triples encestados	Triples intentados	Acierto en triples (%)	Tiros libres encestados
Tiros libres intentados	Acierto en tiros libres (%)	Rebotes ofensivos	Asistencias

Defensa			
Rebotes defensivos	Robos	Tapones	Faltas cometidas

- **SportsDataIO**

Se trata de una API muy completa, la cual, proporciona numerosos datos de los principales deportes existentes.

Las diferencias fundamentales con la anterior API, que han provocado que finalmente se obtenga más información de esta, es que ofrece enlaces a imágenes frontales de los jugadores, a los escudos de los equipos e información de los estadios.

De esta se ha obtenido información sobre lo que se indica a continuación:

- Jugadores: Nombre completo, equipo en el que juegan y url de su imagen frontal.
- Equipos: Nombre completo, ciudad, conferencia, división, entrenador y url de la imagen de su escudo.
- Estadio: Nombre completo, dirección, ciudad, país, latitud y longitud.
- Partidos: Resultado, fecha del partido, equipo local y visitante.

5.2. Organizando el proyecto

La planificación y organización previa a la ejecución de un proyecto supone una importante fase para asegurar su éxito y eficiencia. En este contexto, la utilización de herramientas visuales se revela como un componente fundamental para estructurar y gestionar de manera efectiva los diversos aspectos que conforman el desarrollo de la aplicación. En el presente apartado, se utiliza el diagrama de casos de uso como herramienta para organizar las ideas sobre las distintas funciones de la aplicación y un diagrama de clases que servirá de plano para su desarrollo.

5.2.1. Diagrama de casos de uso

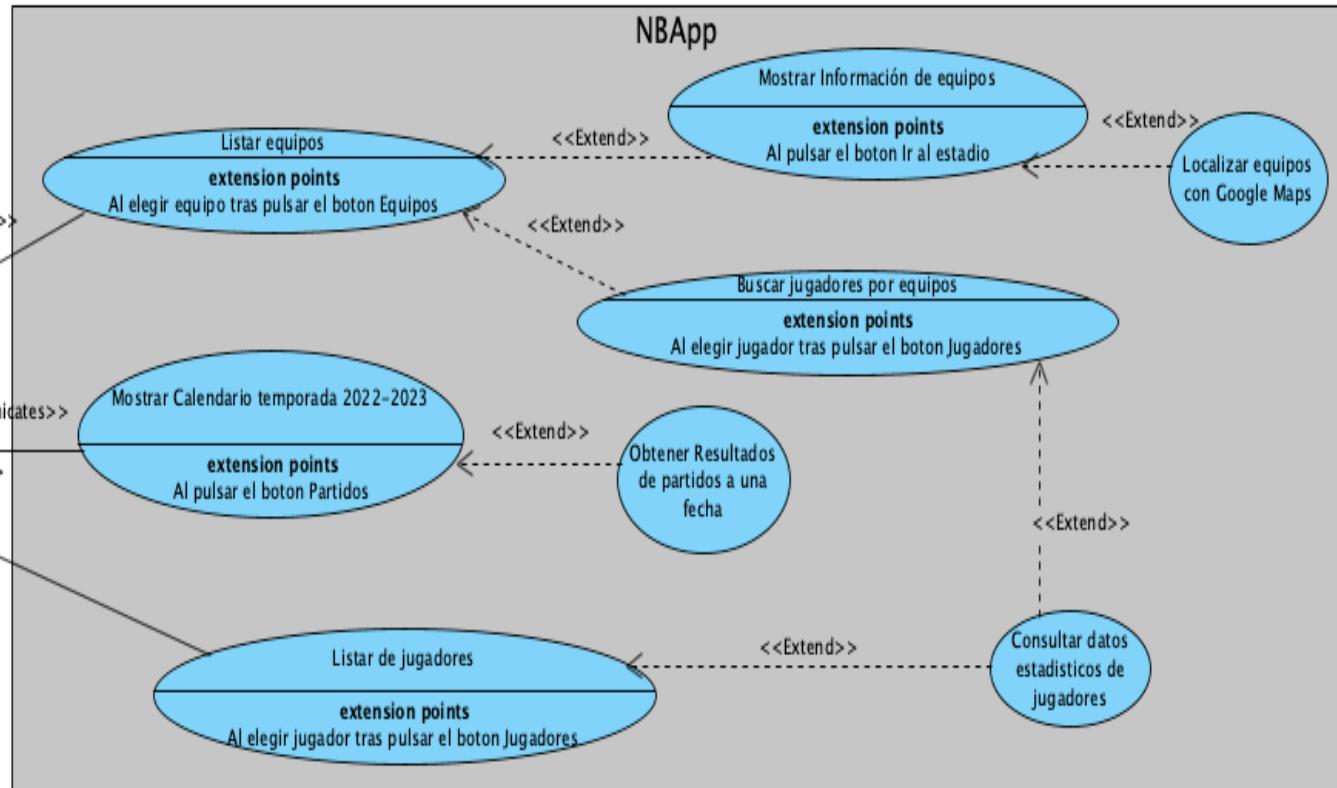
Para organizar el proyecto es importante saber qué funciones tiene que realizar la aplicación y cómo los usuarios interactuarán con ella. Para ello es útil diseñar un diagrama de casos de uso que no es otra cosa que una representación visual que describe cómo interactúan los usuarios con un sistema en particular para lograr un objetivo específico. Estos diagramas son parte de la notación UML (Unified Modeling Language o Lenguaje Unificado de Modelado) y se utilizan comúnmente para capturar y comunicar los requisitos funcionales de un sistema mediante su interacción con los usuarios y/o otros sistemas.

Existen diversos motivos y ventajas de para utilizar un diagrama de casos de uso como herramienta de diseño de un aplicación:

- Claridad en la comunicación: Los diagramas de casos de uso proporcionan una forma visual y clara de comunicar la funcionalidad del sistema a todas las partes interesadas, incluidos clientes, desarrolladores y diseñadores.
- Enfocado en requisitos del usuario: Se centran en los requisitos funcionales del usuario, ayudando a comprender cómo los usuarios interactúan con el sistema y qué funcionalidades son esenciales desde su perspectiva.
- Facilita el entendimiento de la aplicación: Son sencillos, incluso para personas no técnicas, lo que facilita la colaboración entre equipos multidisciplinarios.
- Está orientado a objetivos del usuario: Ayuda a mantener el enfoque en los objetivos y necesidades del usuario, lo que es crucial para el éxito del sistema.

- Documentación Clara y Concisa: Ofrecen una documentación clara y concisa de la funcionalidad del sistema, lo que es beneficioso para futuros desarrolladores y para la comprensión del sistema a lo largo del tiempo.

A continuación se muestra el diagrama de casos de uso de la aplicación que se va a desarrollar en el proyecto:



En el diagrama se puede observar cómo el usuario al iniciar la aplicación tiene la opción de acceder a una lista de jugadores de la NBA, a una lista de Equipos de esta liga o al calendario de la temporada 2022-2023. Según su selección habrá una funcionalidad distinta

- Lista de Jugadores NBA.

El usuario puede explorar la lista completa de jugadores de la NBA.

Después de seleccionar un jugador desde la lista, el usuario puede acceder a una sección que proporciona estadísticas detalladas sobre el desempeño del jugador en la NBA.

Esto puede incluir información como puntos promedio, asistencias, rebotes, etc.

- Lista de Equipos NBA:

El usuario puede explorar una lista de todos los equipos de la NBA.

Al seleccionar un equipo un equipo desde la lista, el usuario puede acceder:

- Por un lado a una sección que proporciona información detallada sobre ese equipo. Esto podría incluir detalles sobre la ciudad, el estadio y otros datos relevantes.
- Desde la información detallada del equipo, el usuario puede acceder a una función que muestra la ubicación del estadio del equipo en Google Maps.

Esto ofrece una experiencia visual para la ubicación geográfica del equipo.

- Por otro lado, permite al usuario buscar jugadores específicos asociados a un equipo particular.

Esto facilita la identificación de los jugadores que forman parte de un equipo específico.

- Calendario Temporada 2022 y 2023:

El usuario puede acceder al calendario de partidos de las temporadas 2022 y 2023.

Permite al usuario seleccionar una fecha específica y consultar los resultados de los partidos que tuvieron lugar en ese día.

Proporciona una forma rápida de acceder a la información de los partidos pasados.

5.2.2.Diagrama de clases

Una clase es un elemento que define las características del objeto que representa, incluidos sus atributos y comportamientos o métodos.

Los diagramas de clases cumplen la misma función que un plano, ya que muestra el diseño y la organización del conjunto de clases que conforman un proyecto de desarrollo de software.

Dicho lo cual, un diagrama de clase tiene dos propósitos principales:

- Visualizar las clases de un sistema y sus propiedades.
- Mostrar y analizar las relaciones entre las clases.

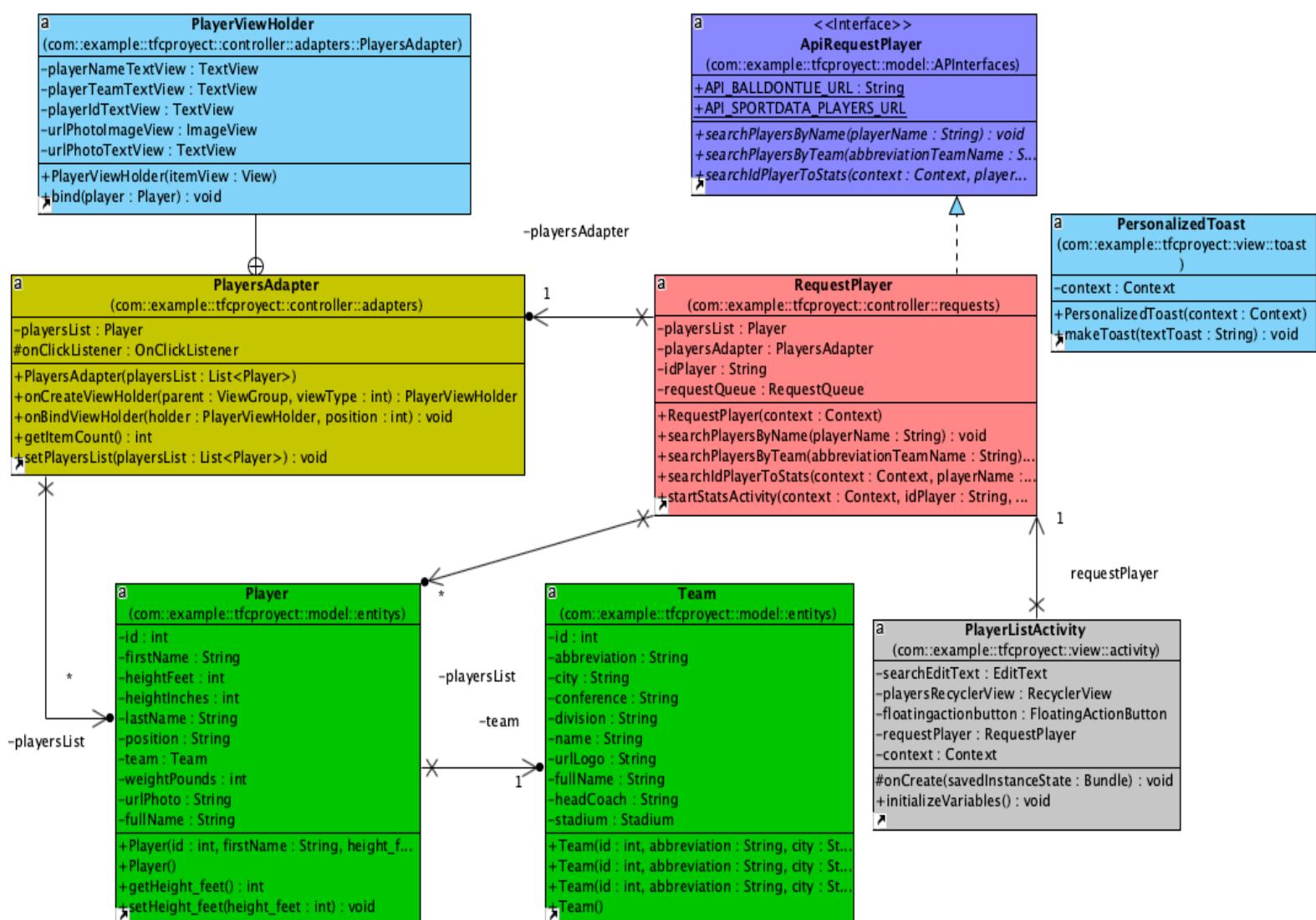
El uso de esta herramienta permite visualizar el sistema, saber cómo funciona, cómo se comporta y cómo se relacionan sus partes.

En el diagrama de clases se han distinguido las clases incluidas en los distintos paquetes que formarán el proyecto por los siguientes colores que muestran las viñetas.

model	view	controller
❖ APIInterfaces	❖ activitys	❖ adapters
❖ entitys	❖ toast	❖ request
		❖ useCases

El proyecto se organizará de la siguiente forma, independientemente de la lógica de negocio a aplicar en cada caso: Una interface que servirá de “plantilla” con los métodos necesarios para realizar peticiones a las APIs, una clase que implementa la interface, las entidades, jugador y equipo en este caso, una clase que sirve como adaptador para comunicar los datos obtenidos con la vista, la activity que representa la ventana que se mostrará al usuario y el resto de clases que tengan que complementar a estas para conseguir el objetivo deseado.

A continuación se muestra el diagrama de clases realizado:



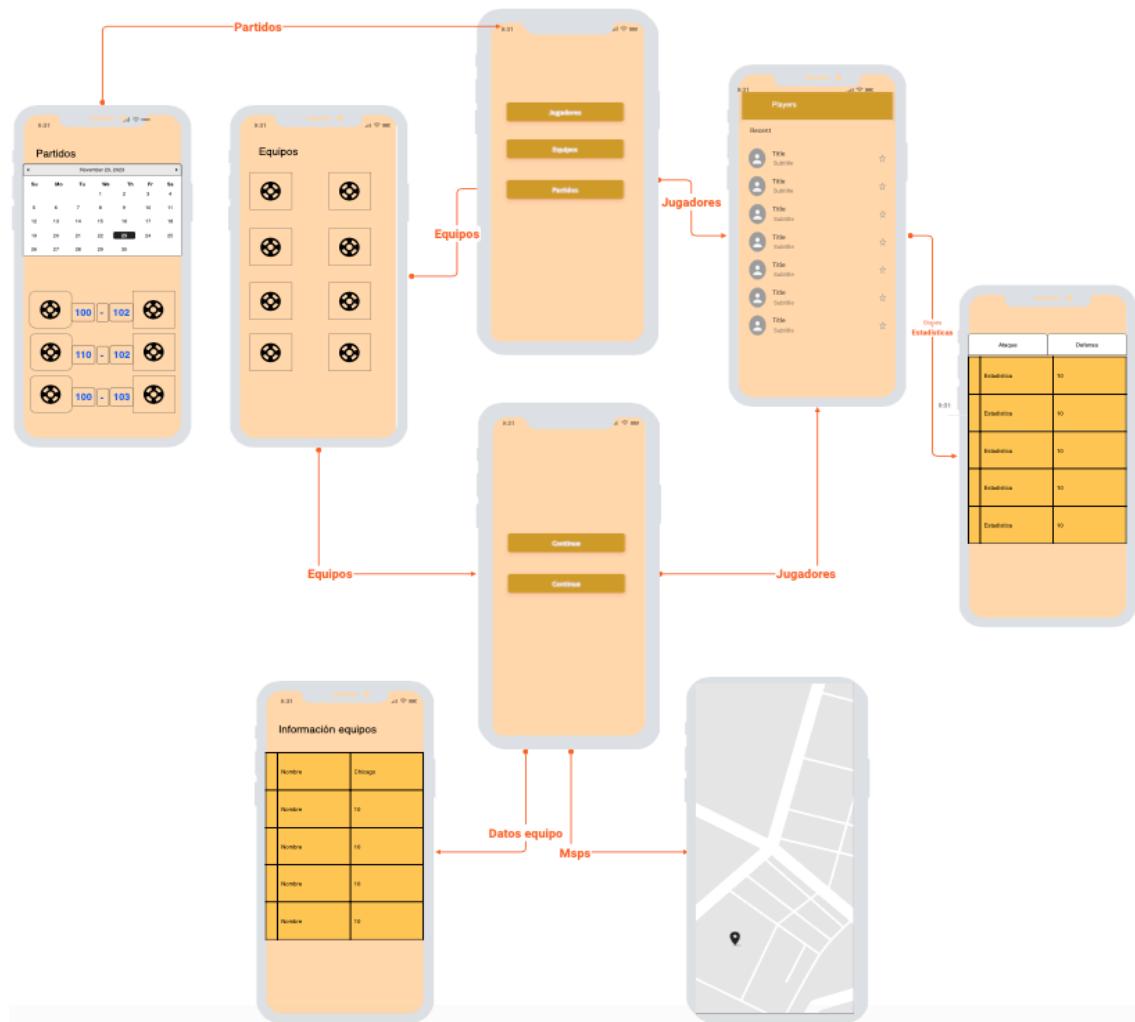
6. Diseñando el proyecto

En la fase de diseño previo a la implementación no solo se trata de planificar la apariencia visual del software, sino también de estructurarlo de manera lógica y eficiente. En este apartado, se trata sobre el diseño visual y el diseño arquitectónico en paquetes, específicamente bajo el enfoque del Modelo-Vista-Controlador (MVC).

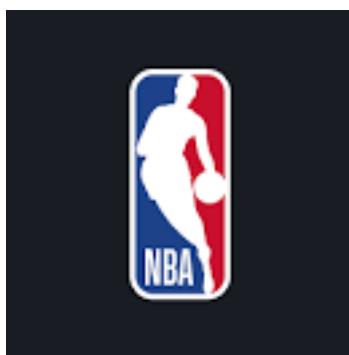
6.1. Diseño de las interfaces

- **Mockup:** Se trata de la representación visual del diseño de la interfaz de usuario lo que permite ver una primera imagen de lo que se quiere construir. Este elemento facilita el trabajo del desarrollo front end y permite realizar cambios y ajustes rápidos en caso de necesidad.

Tal y como se muestra en el mockup diseñado, la aplicación constará de varias ventanas: Una principal que se encuentra en el medio desde la que se puede acceder a otras tres. Además, según la opción elegida, se puede acceder a otras relacionadas con su funcionalidad, ya sea mostrar información de jugadores, equipos o partidos.



- **Paleta de colores:** El color principal con código hexadecimal #996300 (Dorado) se ha seleccionado porque se quería elegir un color que fuera parecido al de una pista de baloncesto. El color #003699 (Azul Oscuro), utilizado como texto para los nombres de los jugadores, estadísticos, resultados, etc. se ha elegido porque se ha considerado que combinaba adecuadamente con el color principal. Por último, el color #FFFFFF (Blanco) se ha utilizado para el texto de los botones, como contraste al color principal. Los dos primeros colores se han utilizado en las tablas expuestas en el presente documento.
- **Fondo de la aplicación:** La imagen utilizadas para todas las ventanas es una pista de baloncesto con las gradas llenas para presenciar un partido.
- **Logo:** Al tratarse de una aplicación relacionada con información asociada con la NBA se ha utilizado su logo, siendo este uno de los más reconocibles a nivel mundial en el ámbito deportivo. Se ha utilizado debido a que se trata de un proyectos con fines académicos, es decir, sin ánimo de lucro.



Fuente: <https://www.nba.com>

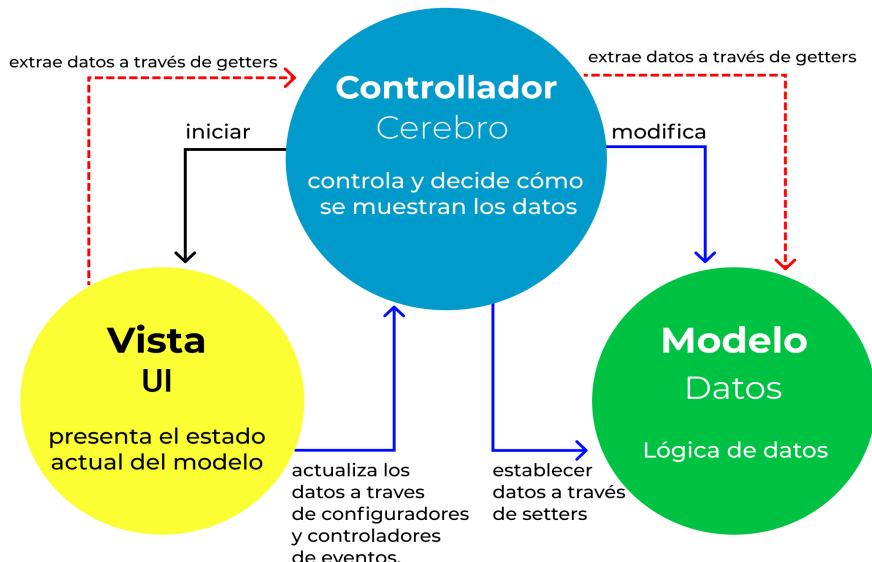
6.2. Arquitectura Modelo-Vista-Controlador (MVC)

El patrón de diseño Modelo-Vista-Controlador (MVC) es una arquitectura comúnmente utilizada en el desarrollo de software que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.

Se ha elegido este patrón por las ventajas que ofrece ya que permite dividir la aplicación en componentes independientes, facilitando la gestión y el mantenimiento del código, facilita la reutilización de código al dividir la aplicación en capas, facilita el mantenimiento de la misma, ya que cambios en la lógica de negocio no afectan directamente la interfaz de usuario, y viceversa, por último, favorece la escalabilidad.

Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador.

Patrones de Arquitectura MVC

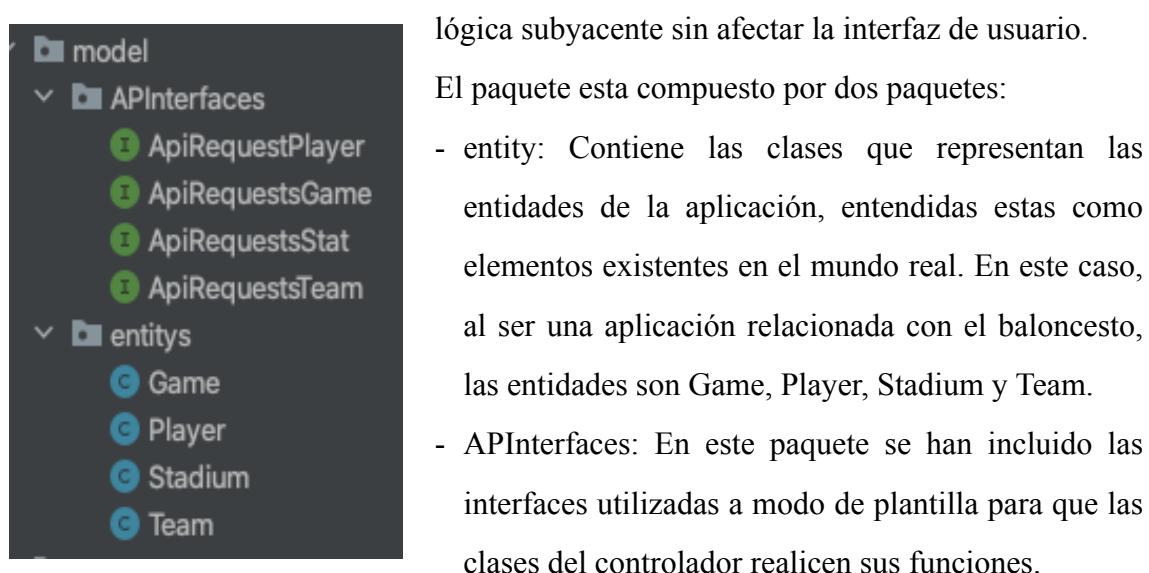


Fuente: <https://www.freecodecamp.org/espanol/news/el-modelo-de-arquitectura-view-controller-pattern/>

- **Modelo (Model)**: Representa la lógica de negocio y los datos de la aplicación. Es responsable de acceder y manipular los datos, así como de contener las reglas de negocio. Al separar esta capa, se facilita la reutilización del código y la gestión de la

lógica subyacente sin afectar la interfaz de usuario.

El paquete está compuesto por dos paquetes:



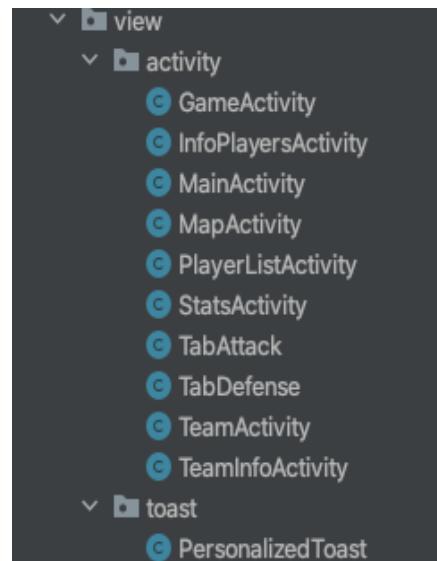
- entity: Contiene las clases que representan las entidades de la aplicación, entendidas estas como elementos existentes en el mundo real. En este caso, al ser una aplicación relacionada con el baloncesto, las entidades son Game, Player, Stadium y Team.
- APIInterfaces: En este paquete se han incluido las interfaces utilizadas a modo de plantilla para que las clases del controlador realicen sus funciones.

- **Vista (View)**: Se encarga de la presentación y la interfaz de usuario. Muestra la información al usuario y recopila la entrada del usuario. La vista no contiene lógica de

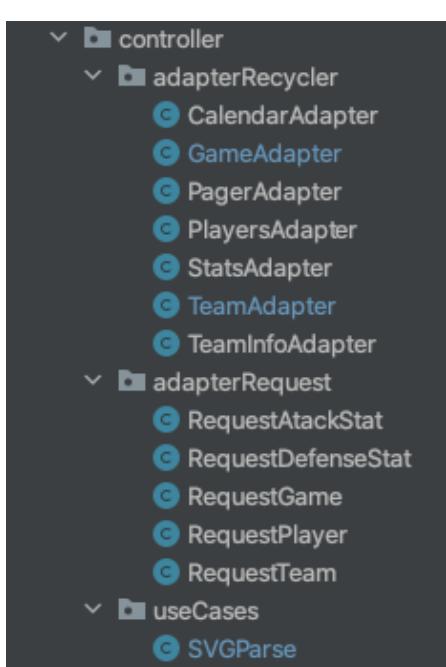
negocio; simplemente muestra los datos proporcionados por el modelo. Esto permite cambios en la interfaz sin afectar a la lógica.

Al igual que el anterior paquete también consta de otros dos:

- activity: Es el paquete que recoge todas las activitys de la aplicación. Estas son un componente clave para la aplicación, ya que representa las distintas pantallas que se mostraran. A diferencia de los paradigmas de programación en los que las apps se inician con un método main(), el sistema Android inicia el código en una instancia de Activity.
- toast: Contiene una única clase, cuya función es mostrar un pequeño texto cuando ocurra un determinado evento.



- **Controlador (Controller):** Maneja la interacción entre el modelo y la vista. Responde a los eventos de usuario y actualiza el modelo o la vista según sea necesario. Al separar la lógica de control, se mejora la modularidad y la capacidad de realizar cambios sin afectar otras partes del sistema.



El controlador contiene tres paquetes:

- requests: Clases que implementan las interfaces diseñadas en el modelo, y desde las que se realizaran las distintas peticiones a las dos APIs.
- adapters: Este paquete contiene los adaptadores utilizados. Estos son componentes que se utilizan para “adaptar” los datos obtenidos de una fuente de información, ya sea de base de datos, API u otros, en la vista.
- usesCases: Contiene la clase SVGParse que se ha utilizado por algunos adaptadores para parsear una imagen de escudo recibida en formato .svg

7. Desarrollando el proyecto

Como ya se ha comentado anteriormente, el presente proyecto de fin de ciclo trata sobre el desarrollo de un aplicación móvil. Esta se compone de distintas activitiys que no son otra cosa que cada ventana mostrada en la interfaz de usuario.

A lo largo de este apartado, se explorará exhaustivamente cada una de las activitys que conforman la aplicación. Para cada una de ellas, según sea posible, se detallarán los siguientes apartados:

- **Funcionalidad:** Se explican las acciones y tareas específicas que se pueden hacer y cómo permite a los usuarios interactuar con ella.
- **Modelo:** En este apartado se comenta sobre las clases entidades con sus atributos y las interfaces con sus métodos abstractos y constantes que componen el modelo de cada activity.
- **Vista:** Se detalla el contenido de los fichero .xml en los que se incluyen los distintos elementos que componen la interfaz y cómo se organizan. Se explica la funcionalidad incluida en las clases que representan la activity, Por último se incluye una imagen de la interfaz.
- **Controlador:** Hay que tener en cuenta que al consumir una API es importante conocer el modelo de datos que se va recibir para poder manipularlo según las necesidades de la aplicación. En este apartado se trata sobre las clases que intervienen en las peticiones realizadas y cómo se gestionan los datos hasta mostrarlos por la interfaz. Se ha incluido imágenes de los distintos JSON recibidos.

El punto de partida es la interfaz principal o de inicio, que sirve de puerta de entrada a los tres casos de usos planteados. Se realizará un recorrido por cada uno de ellos, según el orden de los botones, tal y como, se establece a continuación.

7.1. MainActivity

Funcionalidad

La clase MainActivity sirve como el punto de entrada y es la interfaz principal, la cual contiene tres botones, con el texto jugadores, equipos y partidos, que conducen a diferentes ventanas de la aplicación.

Vista

Las distintas activitays que componen la aplicación tienen asociadas un fichero xml que contiene los distinto elementos que serán los que conformaran la interfaz que se mostrará al usuario.

El fichero relacionado con la clase MainActivity se denomina activity_main.xml, utiliza un ConstraintLayout para organizar las vistas.

Contiene dos elementos ImageView, uno es la imagen de fondo de la interfaz y el otro en una imagen con el escudo y nombre NBA y tres botones.

Estos tres botones tiene como funcionalidad cambiar de activity, es decir, permite la movilidad entre las distintas interfaces de la aplicación.

En la clase MainActivity se han creado tres objetos que representan dichos botones. Estos tienen métodos para “escuchar” distintos eventos. En este caso, al realizar click sobre alguno de ellos, según corresponda, una nueva activity será inicializada.



7.2. PlayerListActivity

Funcionalidad

Esta interfaz muestra una lista de los distintos jugadores de la NBA. Ademas incluye un botón que genera una entrada de texto, lo que permite buscar a un jugador por nombre. Una vez se haya localizado el jugador deseado se puede hacer click sobre él para ir a una nueva ventana.

Modelo

Las clases e interfaces incluidas en el paquete modelo tienen como objetivo servir de base para la gestión de las peticiones necesarias para mostrar la lista de los jugadores.

Entidades y atributos:

- Player: id, firstName, heightFeet, heightInches, lastName, position, team, weightPounds y urlPhoto.
- Team: id, abbreviation, city, conference, division, name, urlLogo, fullName, headCoach y stadium.

Para el diseño de estas clases se ha tenido en cuenta el concepto agregación, ya que un player pertenece a un team y tanto el uno como el otro pueden existir de manera independiente.

Interfaces, métodos y constantes:

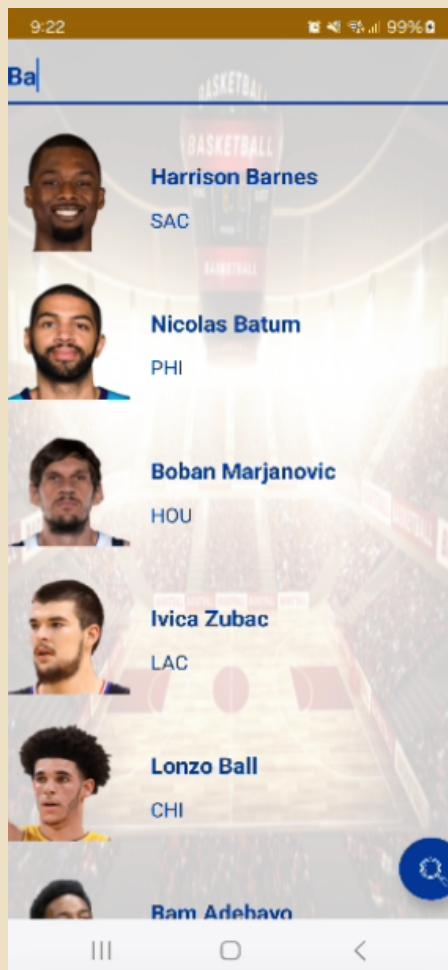
- ApiRequestPlayer:
 - searchPlayersbyName(String playerName): Para obtener los jugadores por nombre.
 - searchPlayersbyTeam(String abbreviationTeamName): Para obtener los jugadores por la abreviación del nombre de su equipo.
 - searchIdPlayerToStats(Context context, String playerName, String urlPhoto, ImageView urlPhotoImageView): Obtiene el id del jugador por el nombre API para poder obtener sus estadísticas y cambiar de actividad.
 - API_BALLDONTLIE_URL = "https://www.balldontlie.io/api/v1/players?per_page=100&search=%s";

- API_SPORTDATA_PLAYERS_URL = "https://api.sportsdata.io/v3/nba/scores/json/Players?key=86ec5c1581744920a90b566123618d07";

Estas dos constantes contienen las url de las APIs donde se realizarán las peticiones.

Vista

- **activity_player_list:** Esta activity se organiza en un RelativeLayout y se compone de la ImageView de fondo, un EditText, un RecyclerView y un botón flotante.
- **item_player:** El RecyclerView muestra una lista de los jugadores, para lo que tendrá que “inflar” este fichero, el cual contiene las vistas necesarias para mostrar en la interfaz una foto, nombre completo y abreviatura del nombre del equipo de cada jugador.



El botón flotante permite que aparezca el EditText para realizar la búsqueda de

jugadores por sus nombres.

La clase PlayerListActivity tiene un objeto requestPlayer que se encarga de gestionar las peticiones para obtener la información mostrada. Al cambiar a la siguiente activity, envía en un Intent el id, en nombre y la url de la foto del jugador. Hay que añadir que al volver a esta activity, se puede observar una transición denominada changeImageTransform, con la que la imagen del jugador elegido se desplaza hasta su posición original.

Controlador

La clase RequestPlayer, que implementa la interface ApiRequestPlayer, se encarga de realizar las distintas peticiones de información sobre los jugadores. En este caso se han tenido que utilizar las dos API mencionadas en el apartado del modelo.

La petición realizada a la API de Sportdataio contenía numerosos datos, a parte de los que se muestran a continuación:

```
{  
    "PlayerID": 20000441,  
    "SportsDataID": "",  
    "Status": "Active",  
    "TeamID": 29,  
    "Team": "PHO",  
    "Jersey": 3,  
    "PositionCategory": "G",  
    "Position": "SG",  
    "FirstName": "Bradley",  
    "LastName": "Beal",  
    "Height": 76,  
    "Weight": 207,  
    "BirthDate": "1993-06-28T00:00:00",  
    "BirthCity": "St. Louis",  
    "BirthState": "MO",  
    "BirthCountry": "USA",  
    "HighSchool": null,  
    "College": "Florida",  
    "Salary": 46741590,  
    "PhotoUrl": "https://res.cloudinary.com/dan&wilpz/image/upload/c_fill,h_100,w_100/headshots/nba/bradley-beal-20000441-03282f33.png",  
    "...": ...}
```

Para el caso que atañe a esta ventana, solo se han utilizado los campos FirstName, LastName, Team y Photourl.

Los datos obtenidos se han pasado como parámetro a un objeto, utilizado como un adaptador, de la clase PlayersAdapter, encargada esta de hacer llegar la información al RecyclerView de la vista.

Debido a que las estadísticas de los jugadores se encontraban en otra API, se ha tenido que combinar la información de las dos.

Del modelo de datos obtenidos, en la API de Balldontlie

```
"id": 14,  
"first_name": "Ike",  
"height_feet": null,  
"height_inches": null,  
"last_name": "Anigbogu",  
"position": "C",  
"team": {  
    "id": 12,  
    "abbreviation": "IND",  
    "city": "Indiana",  
    "conference": "East",  
    "division": "Central",  
    "full_name": "Indiana Pacers",  
    "name": "Pacers"  
},  
"weight_pounds": null  
},
```

Para acceder a las estadísticas era necesario obtener el id del jugador. Para acceder a este dato, se ha encontrado cómo patrón común el first y last name del jugador. Por tanto, se ha usado parte de la respuesta obtenida en una para realizar una nueva petición y obtener el id que será pasado en un Intent a la siguiente activity.

Utilizando el contexto de PlayerListActivity se ha podido cambiar desde esta clase a StatsActivity, pasándole, además del anterior, el nombre del jugador e información necesaria para mostrar la imagen del jugador en la pantalla.

7.3. StatsActivity

Funcionalidad

Al seleccionar un jugador, se genera una nueva ventana en la que se muestran los diferentes datos estadísticos generados por este durante la temporada 2022-2023. Se pueden consultar por separadas las estadísticas ofensivas y defensivas.

Modelo

Respecto al modelo de StatsActivity, tiene una interface utilizada como base para gestionar las peticiones para obtener las estadísticas del jugador elegido por el usuario.

Interface, método y constante:

- ApiRequestsStat

- searchStats(String id): Para obtener las estadísticas del jugador por id.
- API_BALLDONTLIE_URL = "https://www.balldontlie.io/api/v1/season_averages?season=2022&player_ids[]=%s";

Esta constante contiene la url de la API donde se realizará la petición.

Vista

- **activity_stats:** La interfaz diseñada para mostrar las estadísticas se compone de un TabLayout que se divide en dos pestañas, una con el nombre “Ataque” y otra denominada “Defensa”. El otro elemento que forma la ventana es un ViewPager en el que se crearán los siguientes Fragments, según corresponda:
- TabAttack: Si la pestaña elegida es la de Ataque, mostrará la información de las estadísticas de ataque.



- TabDefense: Si la pestaña elegida es la de Defensa, mostrará la información de las estadísticas de defensa.



En ambos fragments se muestra el nombre y la fotografía del jugador. Por último, la información de las distintas estadísticas se muestra en un TextView dentro de un RecyclerView.

Se ha controlado que en caso de no existir datos estadísticos para el jugador, se genere un toast que indica “No existen estadísticas para este jugador.”

Controlador

En el paquete controlador se han incluido dos clases que se encargan de realizar las peticiones para obtener la información y otras dos que realizan las gestiones necesarias con el fin de que la misma pueda mostrarse en la interfaz de usuario.

1. Por un lado, las clases RequestAttackStat y RequestDefenseStat, que

implementan la interface ApiRequestStat.

Estas clases se encargan de realizar peticiones a la API de Balldontlie, incluida como constante en dicha interface, utilizando como parámetro el ID del jugador.

A través de la librería Volley se crea una cola de solicitudes con una instancia de RequestQueue y se realiza una solicitud usando REST con el verbo GET del protocolo HTTP.

La respuesta obtenida es un objeto JSON con la siguiente información:

```
"data": [
    {
        "games_played": 61,
        "player_id": 100,
        "season": 2022,
        "min": "32:36",
        "fgm": 7.51,
        "fga": 16.9,
        "fg3m": 2.54,
        "fg3a": 7.52,
        "ftm": 3.28,
        "fta": 4.02,
        "oreb": 1.18,
        "dreb": 2.85,
        "reb": 4.03,
        "ast": 4.43,
        "stl": 0.54,
        "blk": 0.21,
        "turnover": 3.05,
        "pf": 1.97,
        "pts": 20.84,
        "fg_pct": 0.444,
        "fg3_pct": 0.338,
        "ft_pct": 0.816
    }
]
```

Una vez obtenida, se “manipula” la respuesta para utilizar solamente la información que sea necesaria mostrar en la interfaz de usuario según la estadística sea de ataque o de defensa y se añade a la información en una lista.

2. El punto de enlace entre la fuente de datos y la interfaz de usuario es un adaptador. Esta ventana utiliza las clases PagerAdapter y StatsAdapter.

La primera permite crear los fragments, según la pestaña seleccionada, al recibir el id, el nombre y la url de la fotografía del jugador.

Por otro lado, la clase StatsAdapter recibe como parámetro la lista con las estadísticas obtenidas tras las solicitudes descritas en el punto anterior y se

encarga de que la información pueda ser mostrada a través de un RecyclerView.

A continuación, se describen las clases utilizadas para dar cumplimiento al segundo caso de uso de la aplicación, el cual consiste en consultar datos relacionados con los equipos de la NBA.

7.4. TeamActivity

Funcionalidad

Esta interfaz contiene una lista de cada uno de los equipos que conforman la NBA.

Modelo

Las clases e interfaces incluidas en este paquete tienen como objetivo gestionar las peticiones necesarias para mostrar la lista con los distintos equipos.

Entidades y atributos:

- Team: id, abbreviation, city, conference, division, name, urlLogo, fullName, headCoach y stadium.

Interface, método y constante:

- ApiRequestsTeam
 - searchTeams(): Para obtener información sobre los equipos.
 - searchTeams(List<Game> gameList, GameAdapter gameAdapter): Para obtener los escudos de los equipos que se utilizaran en los resultados de los partidos.
 - searchTeamByAbbreviationTeam(String abbreviationTeam): Para obtener información sobre los equipos por la abreviación de su nombre.
 - API_SPORTDATA_TEAM_URL = "<https://api.sportsdata.io/v3/nba/scores/json/teams?key=86ec5c1581744920a90b566123618d07>";

Esta constante contiene la url de la API donde se realizará la petición.

Vista

- **activity_team**: Esta activity se organiza en un RelativeLayout y se compone de la ImageView de fondo y un RecyclerView.

- **item_team:** El contenido del RecyclerView, se organiza en un CardView que contiene la imagen del escudo del equipo y un TextView con el nombre completo del equipo.

La clase PlayerListActivity tiene un objeto requestTeam que se encarga de gestionar las peticiones para obtener la información mostrada. Al cambiar a la siguiente activity, envía en un Intent el nombre abreviado y nombre completo del equipo.



Controlador

En el paquete controlador se ha incluido la clase RequestTeam que incluye toda la lógica para poder realizar peticiones a la API de SportDataIO y obtener información relacionada con los equipos de la NBA.

El modelo de datos obtenido con esta petición es el siguiente:

```
"TeamID": 1,  
"Key": "WAS",  
"Active": true,  
"City": "Washington",  
"Name": "Wizards",  
"LeagueID": 3,  
"StadiumID": 1,  
"Conference": "Eastern",  
"Division": "Southeast",  
"PrimaryColor": "#002B5C",  
"SecondaryColor": "#E31837",  
"TertiaryColor": "#C4CED4",  
"QuaternaryColor": "#FFFFFF",  
"WikipediaLogoUrl": "https://upload.wikimedia.org/wikipedia/en/0/02/Washington_Wizards_logo.svg",  
"WikipediaWordMarkUrl": null,  
"GlobalTeamID": 20000001,  
"NbaDotComTeamID": 1610612764,  
"HeadCoach": "Wes Unseld Jr."  
,
```

La información utilizada para ser mostrada en esta Activity será la que contiene los campos key, name y wikipediaLogoUrl. La misma se añade en una lista de objetos Team que será pasada como parámetro a la clase que se describirá a continuación. La clase TeamAdapter recibe la información comentada anteriormente y la maneja para poder ser mostrada en la interfaz de usuario.

7.5. InfoPlayerActivity

Funcionalidad

Se trata de una pantalla que recibe información sobre el equipo elegido desde la clase TeamActivity. Sirve de puerta de entrada a dos casos de uso distintos: por un lado consultar la información específica del equipo y, por otro lado, acceder a la lista de jugadores disponibles en el equipo. Este tiene el mismo funcionamiento descrito en la clase PlayerListActivity con la diferencia de que únicamente muestra a los jugadores del equipo seleccionado anteriormente.

Vista

- **activity_players_info:** Las vistas utilizadas son la imagen de fondo de la interfaz y dos botones. Estos elementos se encuentran organizados en un

ConstraintLayout.

El botón “Información” permite cambiar de la activity actual a TeamInfoActivity.

Mientras que con el botón “Jugadores” se accede a PlayerListActivity.

En la clase InfoPlayerActivity se han creado dos objetos que representan dichos botones. Al hacer click en algunos de ellos activaran sus métodos OnClickListener, lo que permitirá inicializar algunas de las dos activitays comentadas anteriormente.



7.6. TeamInfoActivity

Funcionalidad

Si se pulsa en el botón Información, se genera una nueva ventana en la que se muestran datos sobre el equipo elegido en la interfaz TeamActivity. Además, tiene un botón que permite acceder a la ubicación del estadio del equipo en cuestión.

Modelo

Las clases e interfaces incluidas en este paquete tienen como objetivo gestionar las peticiones necesarias para mostrar la información del equipo. Además de la clase Team y de la Interface ApiRequestsTeam, descrita en el apartado TeamActivity, se

incluye:

Entidad y atributos:

- Stadium: id, name, address, city, country y latLng (objeto LatLng que incluye la latitud y la longitud del estadio).

Para el diseño de estas clases se ha tenido en cuenta el concepto agregación, ya que un team está asociado a un estadio y tanto el uno como el otro pueden existir de manera independiente.

Vista

- **activity_team_info:** Esta activity se organiza en un RelativeLayout y se compone de la ImageView de fondo, un TextView que muestra el nombre del equipo elegido, un RecyclerView y un botón con el texto “Ir al estadio” que permite cambiar a otra ventana que presenta la posición geográfica del estadio.
- **ítem_team_info:** El RecyclerView está formado por varios LinearLayout que contienen, orientados horizontalmente, dos TextViews, uno que actúa como etiqueta de la información a mostrar y el otro contiene la propia información.



La clase TeamInfoActivity tiene como variable de instancia un objeto de la clase RequestTeam, el cual se encargará de buscar la información necesaria, por la abreviatura del nombre del equipo, para mostrarla por la pantalla.

Con el botón “Ir al estadio”, se inicializa una nueva activity. Previamente se ha obtenido, la latitud, longitud y nombre del estadio para enviarlo a esta en un objeto de la clase Intent.

Controlador

Al igual que en el apartado en el que se ha descrito la ventana de TeamActivity, en esta también se ha utilizado la clase RequestTeam. En primer lugar, se ha buscado la información del equipo, por su abreviatura, realizando la petición a <https://api.sportsdata.io/v3/nba/scores/json/teams?key=86ec5c1581744920a90b566123618d07>, como ya se había comentado. No obstante la información relativa al estadio, se encontraba en <https://api.sportsdata.io/v3/nba/scores/json/Stadiums?key=86ec5c1581744920a90b566123618d07>. Por tanto, se ha tenido que realizar una segunda petición para obtener toda la información a mostrar.

El modelo de datos obtenido con esta petición es el siguiente:

```
{  
    "StadiumID": 1,  
    "Active": true,  
    "Name": "Capital One Arena",  
    "Address": "601 F St. N.W.",  
    "City": "Washington",  
    "State": "DC",  
    "Zip": "20004",  
    "Country": "USA",  
    "Capacity": 20290,  
    "GeoLat": 38.898056,  
    "GeoLong": -77.020833  
},
```

La información utilizada para ser mostrada en esta Activity será la que contiene los campos name, address, City y country. Hay que indicar GeoLat y GeoLong no se muestran en pantalla pero son utilizadas para poder acceder a la ubicación

geográfica del estadio en la siguiente pantalla.

La información sobre el equipo, se almacena en una lista que será pasada como parámetro al adaptador que actuará como enlace para que la información pueda ser mostrada en la vista.

7.7. MapActivity

Funcionalidad

Se encarga de mostrar en Google Map la ubicación del estadio del equipo, seleccionado anteriormente, en tiempo real. Además, dispone de dos botones. El primero traslada la ventana a la localización exacta del estadio mientras que el segundo permite poner una marca en el estadio.

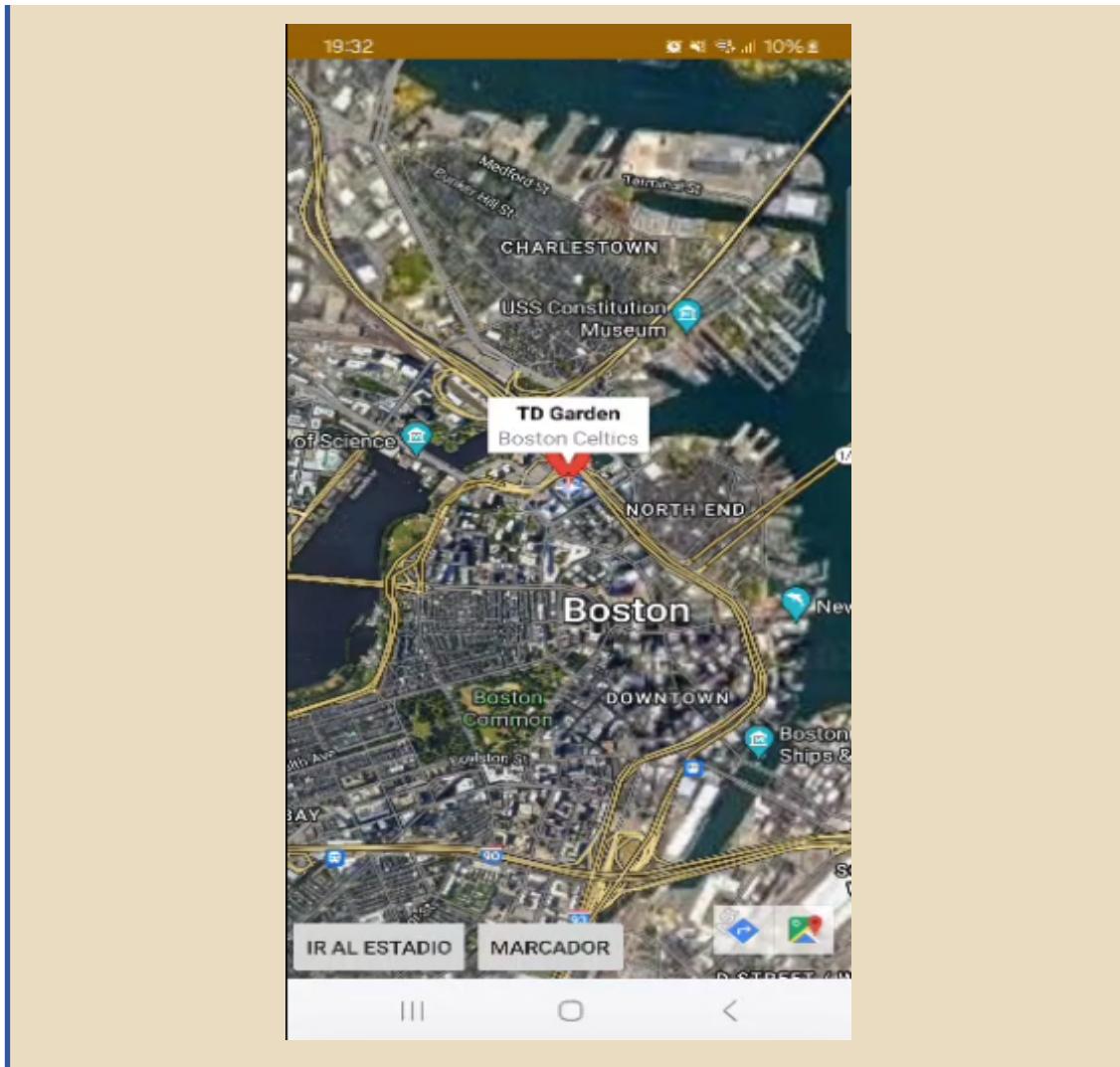
Vista

- **google_maps_api:** En primer lugar se ha obtenido la clave de Google Maps desde la pagina web de Google Cloud y se han habilitado las opciones necesarias para su uso desde la aplicación.
- **activity_map:** El mapa se va a mostrar en un fragment. Además se han añadido dos botones con los textos “Ir al estadio” y “Marcador” para implementar las funcionalidades anteriormente descritas.

La clase MapActivity implementa las interfaces OnMapReadyCallback y OnMapClickListener para poder trabajar con Google Map.

La latitud y longitud obtenidas desde la anterior activity se utilizaran para acceder al objeto googleMap que será insertado en el fragment.

El mapa es cargado asíncronamente, a través de la implementación del método OnMapReady, que será llamado cuando el mapa esté listo. En este método se configura el objeto googleMap según las preferencias de la aplicación. Por ejemplo, indicar en la etiqueta el nombre del equipo y del estadio.



Por último, el tercer caso de uso de la aplicación, se accede al pulsar el tercer botón incluido en MainActivity y permite consultar los resultados de los partidos jugados durante la temporada 2022-2023.

7.8. GameActivity

Funcionalidad

Contiene un calendario que al pulsar en cualquier día muestra los resultados de los partidos jugados en ese día o un toast indicando “No ha habido partidos en esa fecha” si no se ha jugado ninguno.

Modelo

El modelo consta de la siguiente clase e interface:

Entidad y atributos:

- Game: awayTeam, homeTeam, awayTeamId, homeTeamId, awayTeamScore, homeTeamScore, urlLogoAwayTeam y urlLogoHomeTeam.

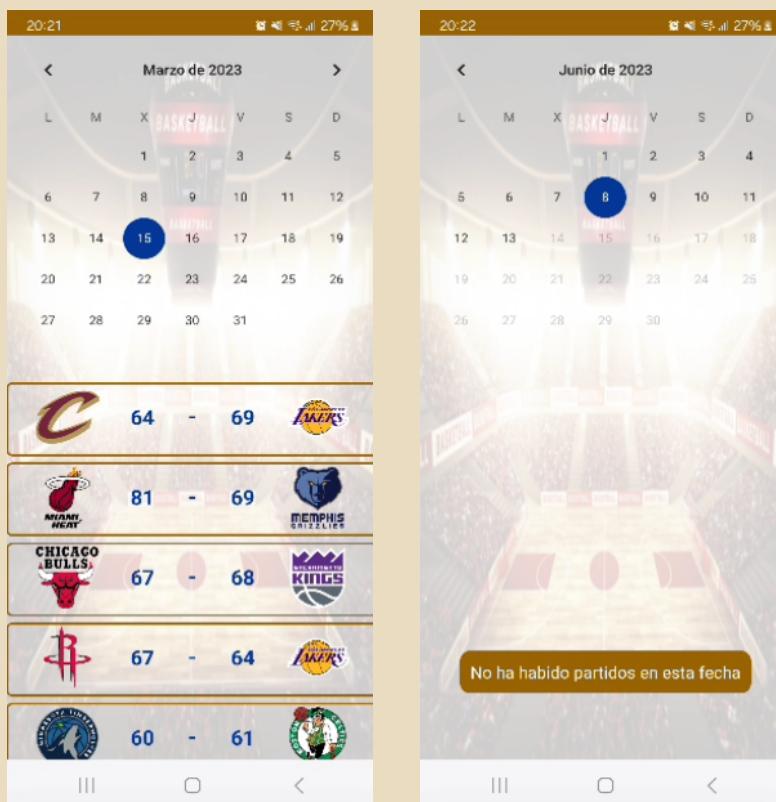
Interface, método y constante:

- ApiRequestsTeam
 - searchGames(int year, int month, int Day): Para obtener información sobre los partidos jugados en un dia concreto.
 - API_SPORTDATA_GAME_URL = "https://api.sportsdata.io/v3/nba/scores/json/GamesByDate/%s-%s-%s?key=86ec5c1581744920a90b566123618d07";

Esta constante contiene la url de la API donde se realizará la petición.

Vista

- **activity_game**: Esta activity se organiza en un RelativeLayout y se compone de la ImageView de fondo, un CalendarView y un RecyclerView
- **ítem_game**: Este fichero contiene dos imágenes para mostrar los escudos de los equipos que hayan jugado el partido y tres textos necesarios para mostrar un resultado, como por ejemplo “100 - 105”. Estos elementos sirven para cargar cada registro de la lista que recibirá el Recyclerview.



En la clase GameActivity, el objeto CalendarView escucha el evento de cambio de fecha y busca el partido en la nueva fecha seleccionada.

Controlador

La clase RequestGame implementa la interface ApiRequestGame, por tanto, se ha tenido que implementar su único método abstracto searchGames. En este se realiza una petición GET a la url almacenada en la constante API_SPORTDATA_GAME_URL en la fecha pasada como parametro, y se añade en una cola. En la respuesta, se recibe un Array de JSON que contiene, a parte de otros, los siguientes datos:

```
"GameID": 19147,  
"Season": 2023,  
"SeasonType": 1,  
"Status": "Final",  
"Day": "2023-03-07T00:00:00",  
"DateTime": "2023-03-07T19:00:00",  
"AwayTeam": "MIL",  
"HomeTeam": "ORL",  
"AwayTeamID": 15,  
"HomeTeamID": 5,  
"StadiumID": 5,  
"Channel": "BSF",  
"Attendance": 16110,  
"AwayTeamScore": 132,  
"HomeTeamScore": 122,
```

Se ha filtrado por aquellos cuyo status es “Final” y se han utilizado los campos AwayTeam, HomeTeam, AwayTeamID, HomeTeamID, AwayTeamScore y HomeTeamScore para instanciar un nuevo objeto por cada elemento y añadirlo a una lista de Game.

Posteriormente, con el objetivo de añadir la url del escudo de los equipos que juegan los partidos, se ha realizado una nueva petición a API_SPORTDATA_TEAM_URL

a través de una instancia de un objeto RequestTeam. En este caso, en la respuesta se han obtenido las urls de los escudos y se han asignado a los equipos de cada partido por el ID de cada uno.

Por último, la lista de partidos, se ha pasado como parámetro al a un objeto GameAdapter para adaptar la información recibida al RecyclerView y poder montarla por la pantalla.

8. Bibliografía

- *Diagrama UML: qué es, cómo hacerlo y ejemplos | Miro.* (s. f.). <https://miro.com/#:~:text=Un%20diagrama%20UML%20es%20una,de%20sistemas%20de%20software%20complejos>
- *Sl, U. T. (2023, 20 noviembre). Android Studio para Windows - descarga gratis en Uptodown.* Uptodown. <https://android-studio.uptodown.com/windows>
- *colaboradores de Wikipedia. (2023, 27 septiembre). Android Studio.* Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Android_Studio
- *¿Qué es Java? | IBM.* (s. f.). <https://www.ibm.com/es-es/topics/java>
- *Moncayo, J. M. R. (2023, 17 abril). Qué es REST: conoce su potencia.* OpenWebinars.net. <https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/>
- *Fernández, Y. (2019, 23 agosto). API: qué es y para qué sirve.* Xataka. <https://www.xataka.com/basics/api-que-sirve>
- *Acerca de Git - documentación de GitHub.* (s. f.). GitHub Docs. <https://docs.github.com/es/get-started/using-git/about-git>
- *El gran libro de Android (Jesus Tomas Gironés y Jaime Lloret Mauri)*
- *colaboradores de Wikipedia. (2023b, noviembre 14). Modelo-Vista-Controlador.* Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador>
- *Introducción a las actividades.* (s. f.). Android Developers. <https://developer.android.com/guide/components/activities/intro-activities?hl=es-419>
- *Android API reference.* (s. f.). Android Developers. <https://developer.android.com/reference>
- *MIRO | The Visual Workspace for Innovation.* (s. f.). <https://miro.com/>. <https://miro.com/es/diagrama/que-es-diagrama-clases-uml/>