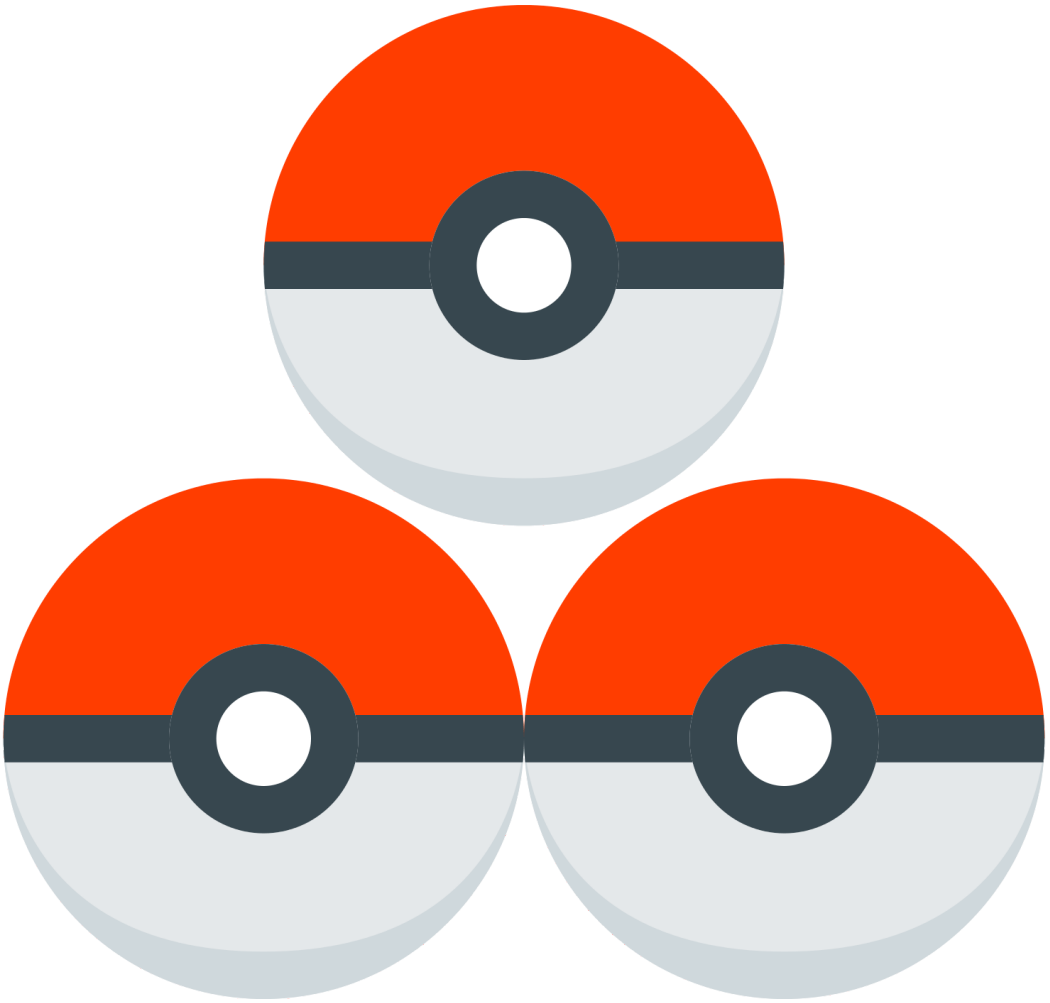
Institut Puig Castellar
Santa Coloma de Gramenet

Pokémon Rumble Arena

CFGS Desenvolupament d'Aplicacions Multiplataforma



Yago Morales Ares

David Tomás Gonzalez

2nDAM B

29/03/2019

Llicències alternatives (triar alguna de les següents i substituir la de la pàgina anterior)

A) Creative Commons:

Resumen del proyecto:

Esta aplicación trata sobre la gestión, organización, lucha y competición contra otros jugadores a través de los Pokémons. La inspiración viene sobretodo de las aplicaciones ya existentes en el mercado como son Biwenger, Comunio o Futmondo combinado con las mecánicas de Pokémon. También se añaden ideas de experiencias propias, de organizaciones realizadas para jugar Pokémon semana a semana y realizar combates. Además, no contamos solo con ideas nuestras, compañeros han ido sugiriendo cambios y mejoras a medida que se desarrolla la idea final.

Esta aplicación consistirá en que los usuarios creen o se unan a una liga, donde competirán con otros usuarios semana a semana. Los jugadores tendrán que comprar, mejorar y organizar su equipo Pokémon para ir ganando cada jornada y volver al ciclo de juego que acabo de mencionar.

Para los usuarios menos expertos de Pokémon, las criaturas que compren los usuarios tendrán unas estadísticas y ataques. Los jugadores podrán mejorar sus propios Pokémon evolucionandolos o cambiando sus ataques o comprar nuevos en el mercado. Luego, los combates se realizarán contra cada jugador automáticamente, el usuario decidirá para cada combate la alienación de salida intentando encontrar la mejor estrategia contra cada jugador.

Palabras Clave :

Pokemon, liga, jugadores, equipo/alienación.

Índice

1. Introducción

1.1 Contexto y justificación del Trabajo

La intención no es cubrir una necesidad primaria o resolver un problema que se puede encontrar actualmente. Lo que se quiere hacer es dar ocio a un público objetivo, tanto de jugadores de Pokémon, como de usuarios de aplicaciones de gestión de equipo/recurso o grupos de amigos que simplemente quieren jugar a algo juntos pero sin estar siempre activos.

1.2 Objetivos del Trabajo

Los objetivos propuestos para el proyecto son los siguientes:

-Aplicación funcional con sus interfaces donde los jugadores pueden moverse con libertad.

-Base de datos en la nube (servidor firebase de google) para almacenar y leer datos.

-Lectura de api para recoger los datos rápidamente de cada Pokémon.

-Ya que los combates serán automáticos, una IA que identificara las debilidades y ataques eficaces y actuará en consecuencia.

-Servidor donde se ejecutarán procesos como resultados de las pujas, nuevos Pokémon en el mercado o el resultado de combates.

-Capacidad de unirse y jugar al mismo tiempo en diferentes competiciones.

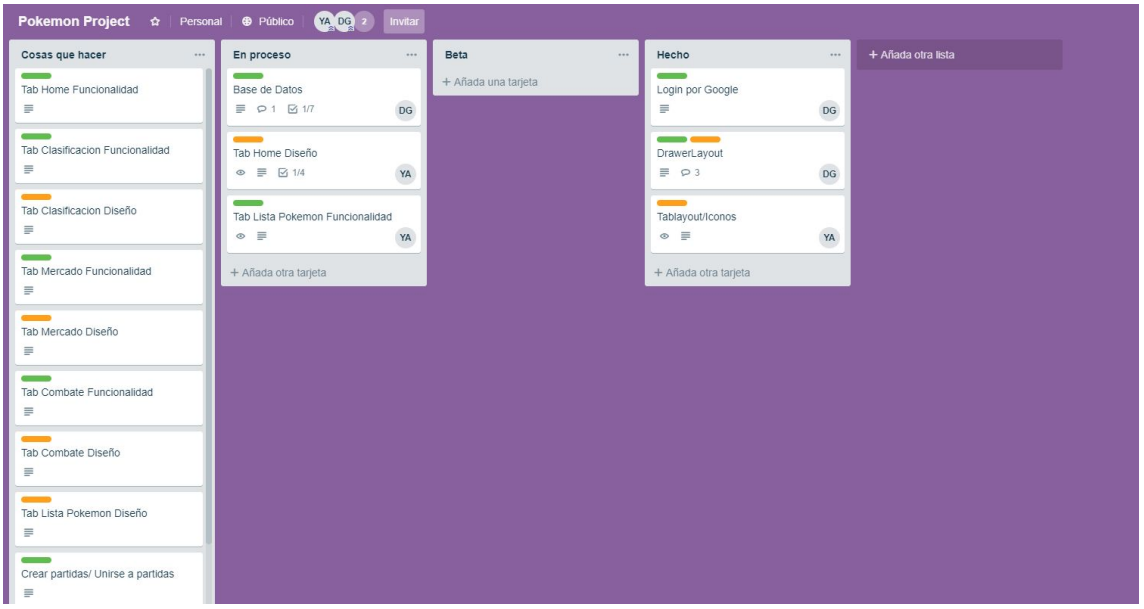
1.3 Enfoque y método seguido

La intención es basarse en interfaces ya existentes que se encuentran en las aplicaciones de gestión ya mencionadas anteriormente para que los usuarios de estas se adapten rápidamente y sean intuitivas desde el principio. En la parte lógica, aunque se aprovecha de trabajos ya realizados durante el curso, se inicia todo de cero con código propio.

Se usa Android Studio para programar el proyecto, el servicio de Firebase de Google para autentificar, registrar y leer datos (Se usará sobretodo la base de datos Firestore). La intención era crear o hostear un servidor propio para que realice las operaciones del juego, pero por tiempo se acabó realizando operaciones que cambian y actualizan los datos de Firebase.

1.4 Planificación del proyecto

Para organizar las tareas, quien lleva cada cosa y saber en que está trabajando cada uno, se utilizara trello, no se utilizara Scrum como tal al pie de la letra, ya que no se encuentran expertos en el equipo como para manejar esta metodología, pero si se usan algunos de sus principios básicos, como la estructura de tareas (To Do, Doing, Done, aunque también se ha añadido una propia que es beta) y aunque no diarias, si se hacen reuniones para ver como avanza el trabajo que se ha repartido y focalizar la siguiente tarea a cumplir.



Tablero de tareas con estructura de Scrum (To Do, Doing Done) en Trello.

1.5 Breve resumen de productos obtenidos

La finalidad es obtener una aplicación plenamente funcional, con posibilidad de jugar varias partidas al mismo tiempo, procesos que se realizarán no en la aplicación, sino un servidor externo que recibirá los datos y los procesara. Además, con la posibilidad de jugar desde cualquier dispositivo Android y si se tuviera tiempo, implementarlo también en página web como aplicaciones ya mencionadas.

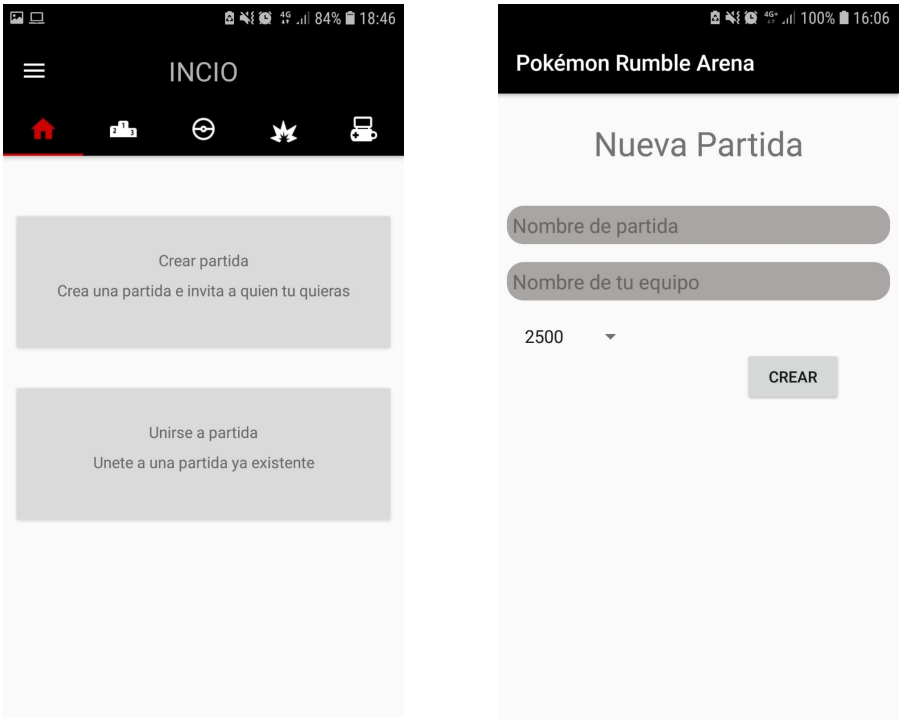
1.6 Breve descripción de los otros capítulos de la memoria

En el siguiente punto, se enumera y se explica elementos externos que se han tomado como referentes o que sirven de base para la aplicación, como las aplicaciones de gestión como Biwenger o Comunio, algunas bases de Pokémon como estadísticas y objetos. También herramientas como firebase que se utilizarán en varios aspectos de la aplicación como por ejemplo la base de datos o la autentificación.

2. Resto de capítulos

2.1 Análisis:

Esta aplicación está orientada a un público objetivo basado en Pokémon. Se busca la esencia de competir con tu propio equipo Pokémon para afrontar una liga en la cual estarán tu amigos. Para ello se ha creado un sistema de gestión de recursos, para poder invertir y mejorar el equipo. Este sistema se basa en las aplicaciones que ya hay en el mercado de gestión de recursos de otros intereses, como Biwenger, que sirve para gestionar un equipo de fútbol, o Fantasy Lcs que se basa en un equipo del deporte electrónico League of Legends.

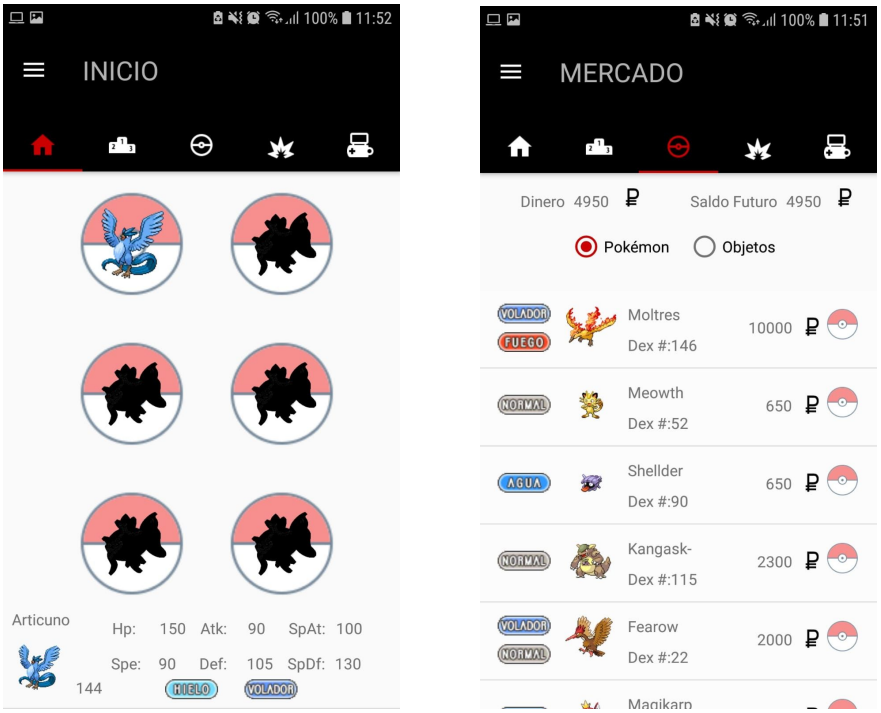


Pantallas de administración. 1. Cuando el usuario entra y no está en ninguna partida. 2. Cuando el usuario va a crear una nueva partida

Las funcionalidades se puede separar en dos ramas, cuando no se está en ninguna liga y cuando se pertenezca a una. Cuando no se está en ninguna liga aparecerán dos opciones, crear una liga de la cual se sera administrador o unirse a una liga. A partir de pertenecer a una liga se abren un gran número de funcionalidades, como comprar pokemons, comprar objetos, establecer un equipo, consultar datos de pokémon, darse de baja de la liga, etc.

Comprar pokemons o objetos se base en la idea de todos los usuarios de la misma partida comparten el mismo mercado, y quien puje mas se llevara el pokemon pujada o el objeto, se sabrá la cantidad de pujas que se han hecho por eso pokémon u objeto pero nunca la puja más alta.

Al establecer un equipo, ese equipo tuyo será el que combata contra los demás equipos cada semana, la recompensa serán puntos en la clasificación y además dinero para poder mejorar tu equipo pokémon.



Pantallas de interfaz principal. 1. Organizar alineación y equipo. 2. Mercado Pokémon.

2.2 Diseño:

Toda la información que se controla, lee y muestra está diseñada utilizando tres bloques en total, una API de la que bajamos la información, la aplicación de intermediario que subirá y modificará datos, y un servidor en Firebase que guardará toda esta información.

Uno de los primeros pasos que se realizó fue la descarga de datos de la api y la subida a la base de datos, ya que es la esencia de lo que se utiliza en la aplicación (Estadisticas,nombres,ataques, etc. de los Pokémon). Se ha utilizado para un par de documentos de datos y modificado de manera que guarde la información que se necesitaba.

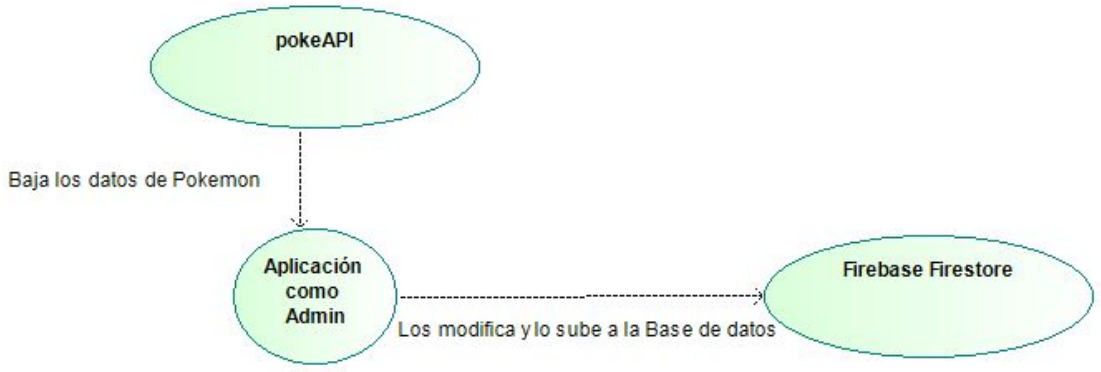


Diagrama de como funciona la conexión y descarga y escritura de datos.

Luego, la aplicación que utiliza el usuario, se conecta para la gran mayoría de operaciones con la base de datos en Firebase para leer, escribir y modificar datos. En esta sección, por ejemplo, encontramos el registro de usuarios, la creación de partidas, el unirse a partidas, comprar Pokémon o objetos, modificar equipos y alineaciones y ver la clasificación y otros equipos.

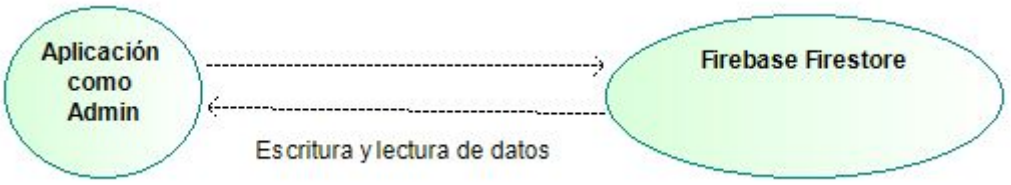


Diagrama de representación de la conexión entre base de datos y aplicación.

2.2.1 Estructura

El diseño de clases que se utilizan en esta aplicación se basan sobretodo en dos bloques, un apartado dedicado a los Pokémon, sus datos, tipos, sprites y otro dedicado a las partidas y sus usuarios. Todo esto subido a Firebase.

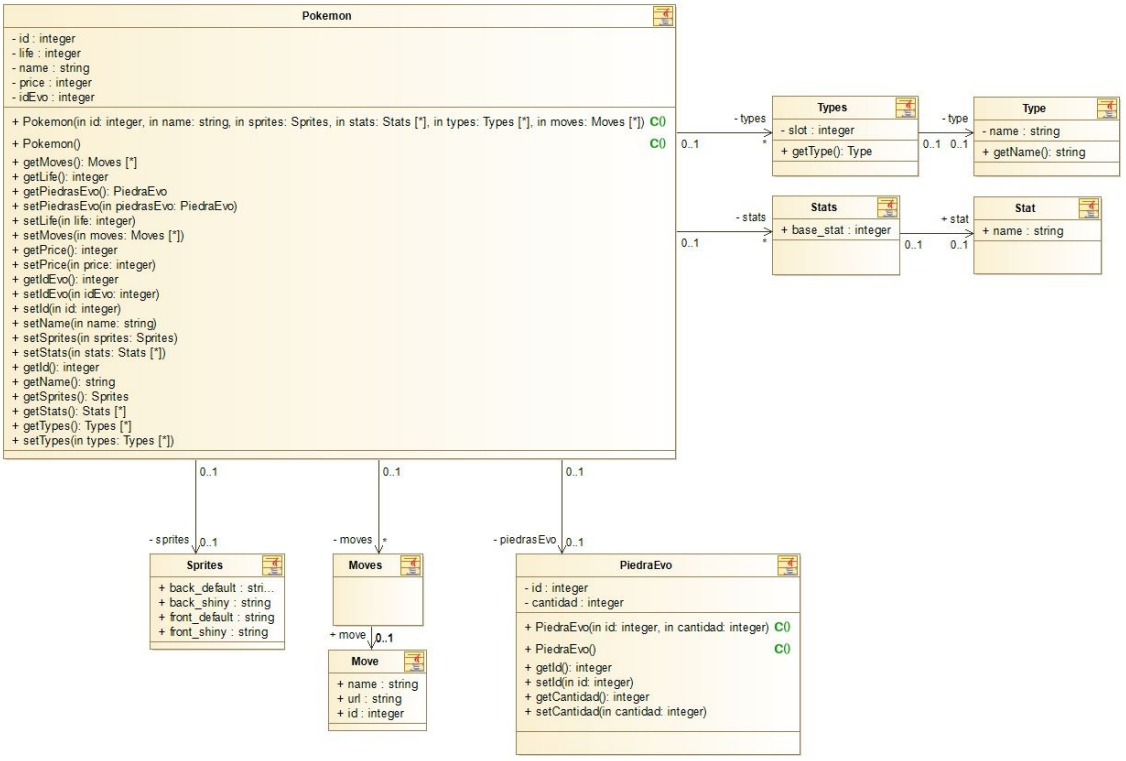
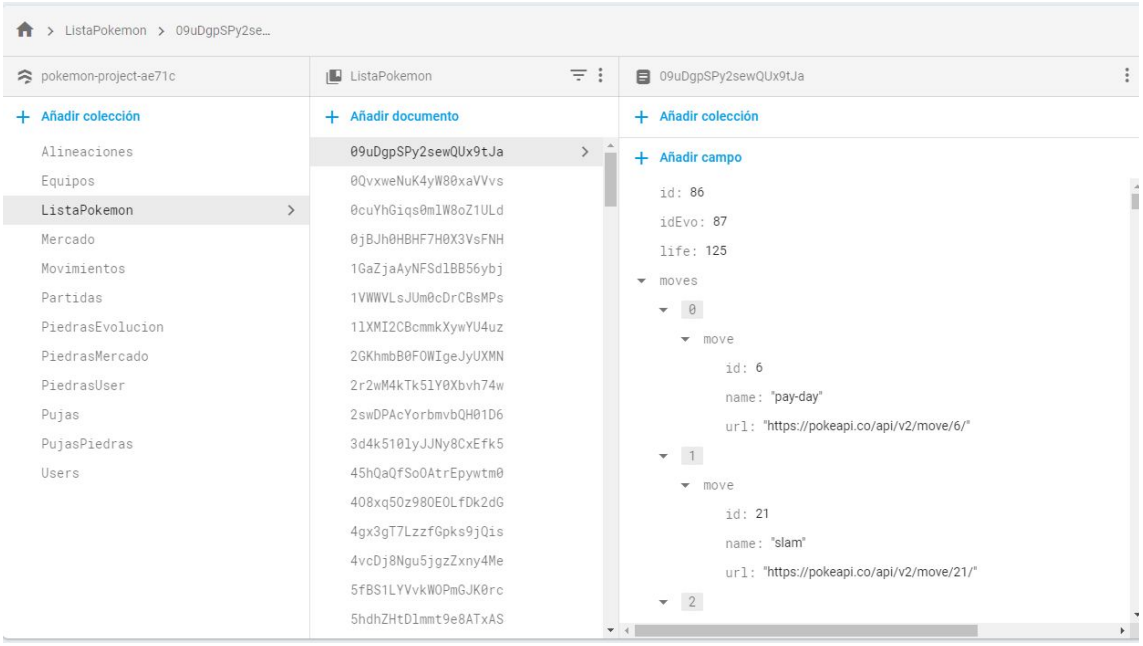


Diagrama de clases de la estructura de los pokemon. Tamaño total en anexos*.

Hay clases con poca información debido a que se sigue la estructura de la api que se utiliza, donde la información está guardada en cada Pokemon, con arrays de estadísticas, tipos y sprites, esto se hizo con la intención de conseguir la mayor cantidad de información posible automáticamente.

Con esto, obtenemos esta estructura en firebase, un documento con una lista de Pokémon, donde cada uno tiene los datos representados arriba.



Estructura de información de los Pokémon en la base de datos Firestore de Firebase.

El bloque de partidas y usuarios funciona de manera diferente, cuando se crea una partida, esta se inicializa con todos los datos necesarios que se piden, un array de usuarios a la que se añade el creador, pero para hacerlo de manera más eficaz, en vez de añadir al usuario solo, lo añadimos con un nombre de equipo a parte.

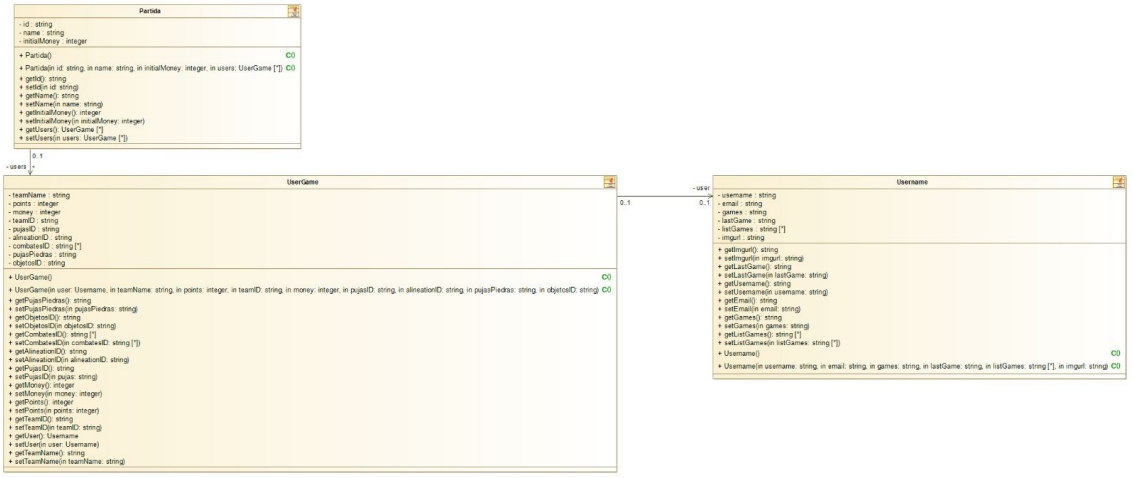
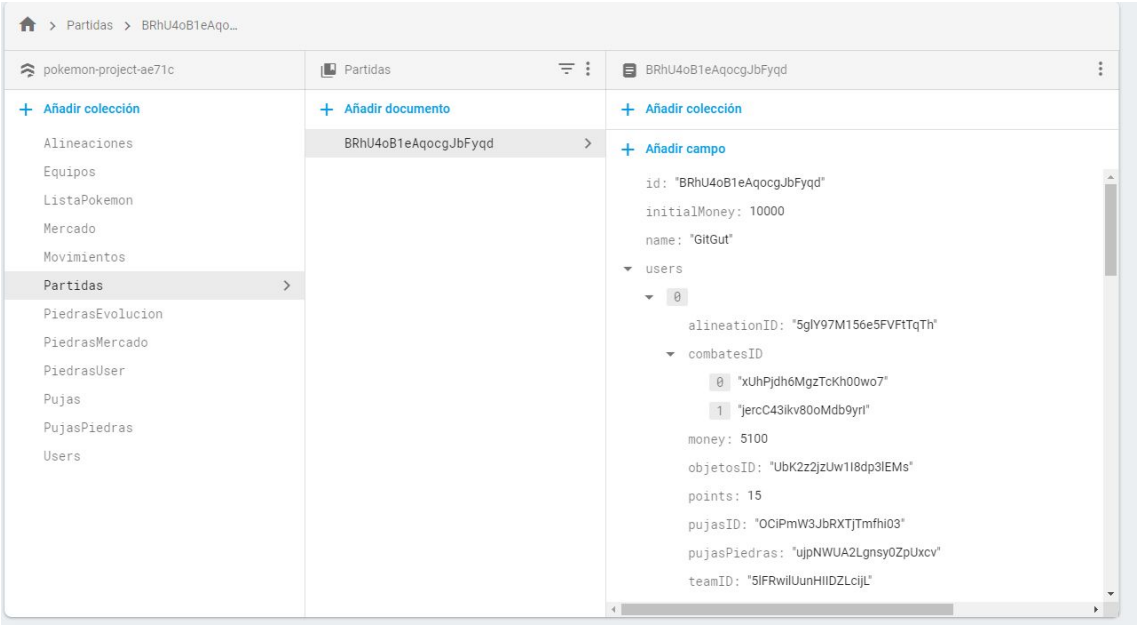
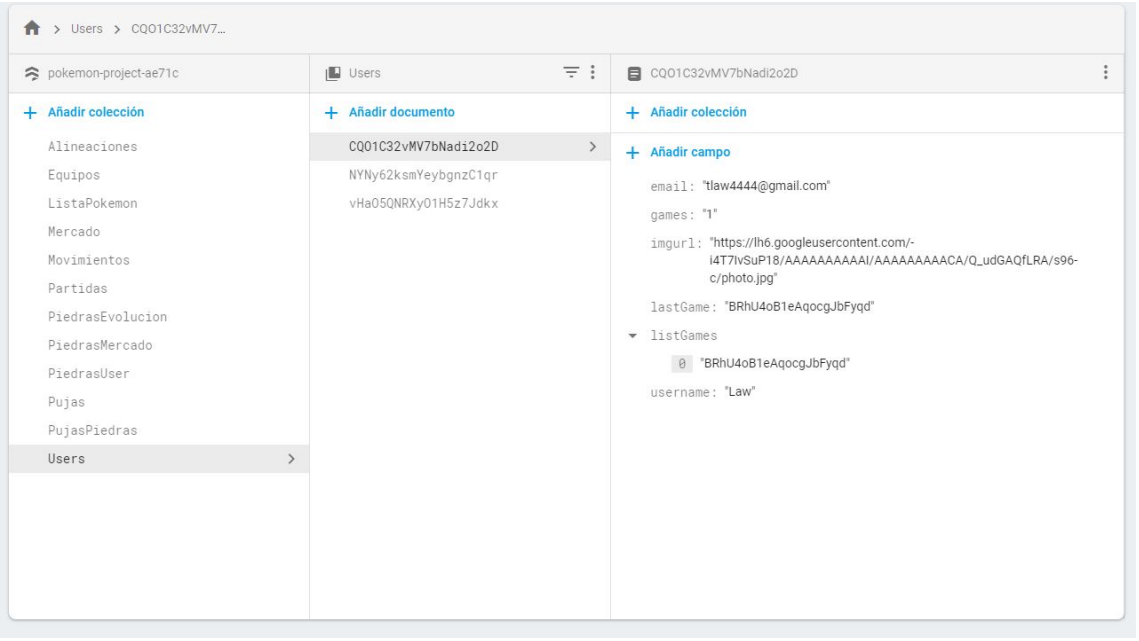


Diagrama de clases de la estructura de las partidas. Tamaño total en anexos*.

Esto está representado en dos documentos diferentes en Firebase, uno para los Usuarios, que sirve como control de registro y base de datos y otro con las partidas, donde encontramos también los usuarios que hay.



Estructura de la información de las partidas en la base de datos Firestore de Firebase.



Estructura de la información de los usuarios en la base de datos Firestore de Firebase.

Es cierto que encontramos otras clases en la aplicación, pero la mayoría no tienen relación y son solo para subir y leer los datos correspondientes de la base de datos de Firebase.

2.2.2 Funcionalidades

En cuanto a lo que el usuario podrá realizar en la aplicación, tenemos varias funciones, diferenciado en usuario y administrador, la parte de conexión/registro y crear/unirse a partidas y la parte de dentro de la aplicación, que serán al final, las funciones reales de esta misma, pertenecen tanto a usuario como a administrador. Luego el administrador podrá realizar las funciones relacionadas con la base de datos, como crearla de 0, actualizar pujas o realizar combates.

2.2.2.1 Funcionalidades Usuario

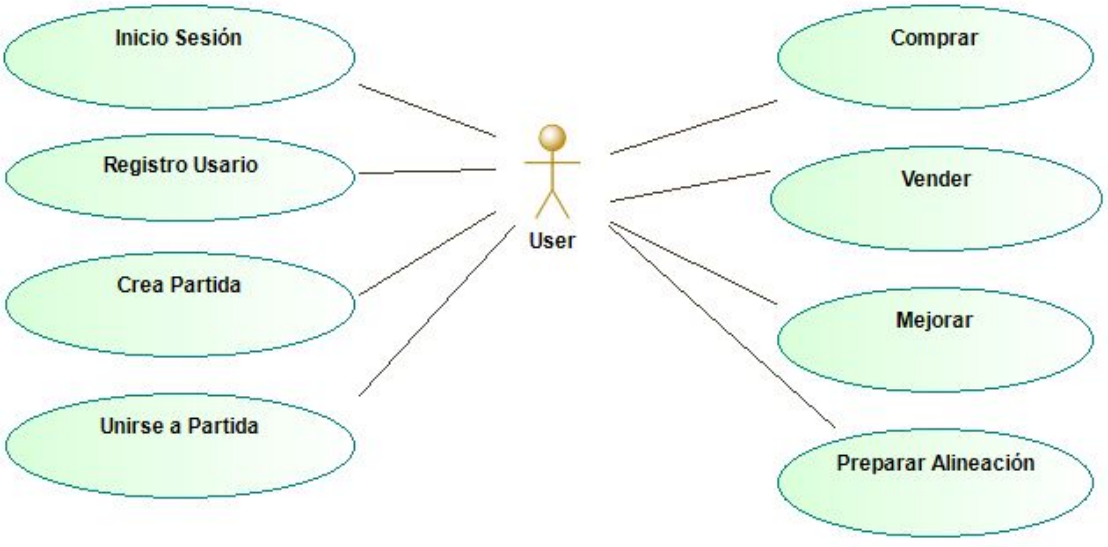
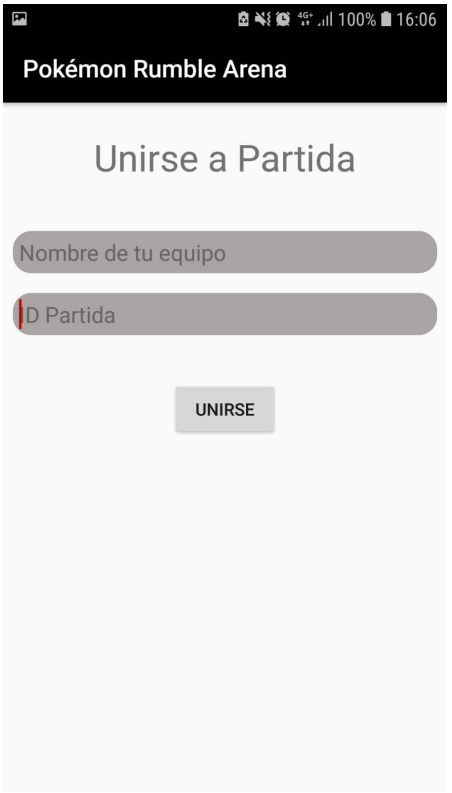


Diagrama de casos de usos de los usuarios separado en dos bloques (conexión y jugabilidad).

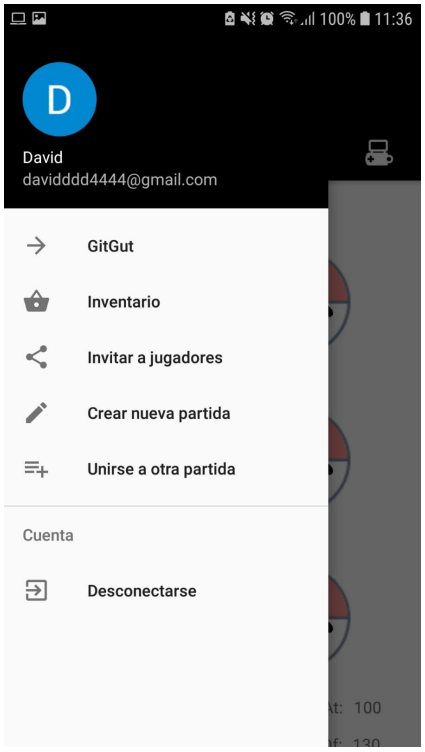
El bloque de conexión/registro funciona a través de una combinación de Firebase y su propia base de datos Firestore. Los usuarios al iniciar la aplicación podrán elegir una de sus cuentas de Google para conectarse, si esta cuenta está registrada en la base de datos, el usuario entrará directamente a la interfaz principal, donde podrá utilizar las funcionalidades principales (si está en una partida) o crear o unirse a una partida. En caso de no estar registrado, se le pedirá un nombre de usuario para hacer este registro y entonces será enviado a la interfaz principal.

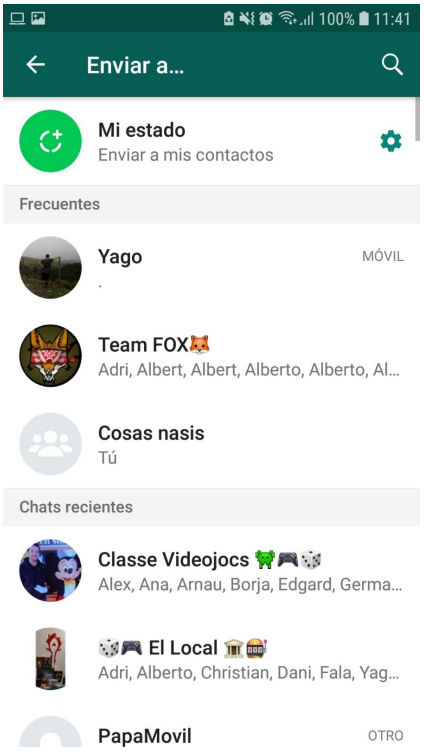
Crear partidas es sencillo, basta con poner un nombre de partida, un nombre de equipo y el presupuesto inicial con el que empezará cada jugador. Una vez creada, podremos invitar fácilmente a otros jugadores, enviando el codigo a traves de whatsapp con un solo clic.

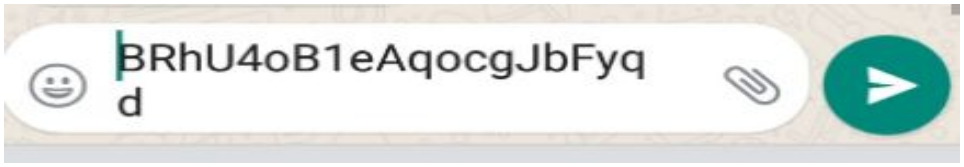


1. Pantalla de unirse a partida.

Para unirse a partidas, bastará con introducir este código que nos compartan, elegir un nombre de equipo y directamente entraremos en la partida. Las partidas están limitadas a 8 jugadores.







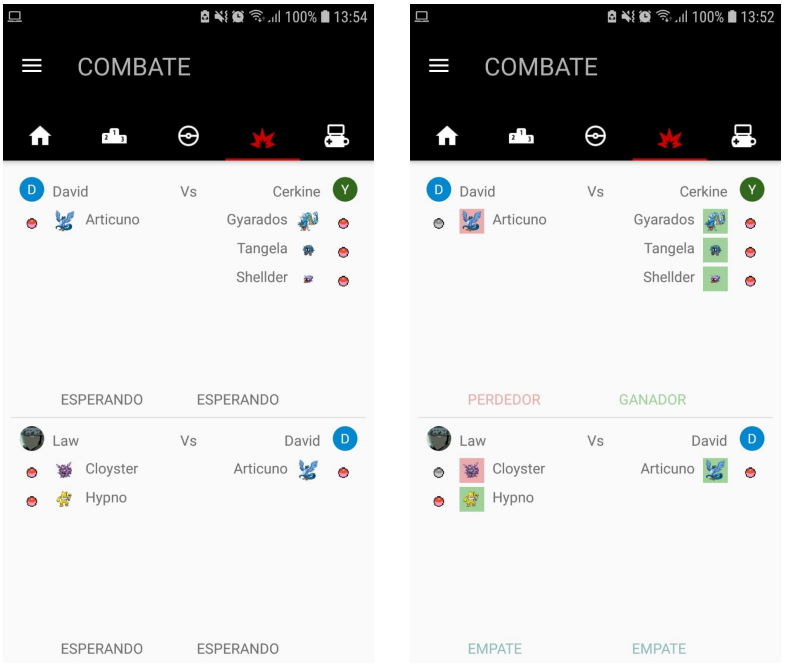
1. Drawer desde el que podremos invitar a jugadores en la partida que tenemos activa, crear nuevas partidas o unirse a unas ya existentes. 2 y 3. Al clicar invitar a jugadores nos redirigirá a whatsapp para mandar el codigo a quien queramos.

El otro bloque, que consiste en las funcionalidades principales, encontramos la compra y la venta de los Pokemon, los usuarios podrán pujar entre 10 Pokemon cada día, compitiendo contra otros usuarios para ver quien puja más y llevarse el Pokemon para añadirlo a su equipo, y si, se necesita dinero o un Pokemon ya no entra en sus filas, ponerlo a la venta para que otros usuarios puedan venderlo.

Luego, con estos Pokémon que se adquieren, se organizara la alineación para cada semana, ya sea poniendo nuevos Pokémon adquiridos, cambiando el orden o mejorandolos (evolucionandolos o cambiando sus ataques).

Si bien es cierto que estas son las funcionalidades que el usuario puede realizar, todo esto tiene como objetivo el luchar contra los equipos de otros jugadores de la partida para ver quien es el mejor. Semana a semana, cada equipo de cada jugador se enfrentará al resto.

Estos combates se realizarán automáticamente, controlados por una IA para escoger cada ataque. El primer Pokémon del equipo se enfrentará al primero del equipo contrario y así sucesivamente. Por cada combate ganado en el enfrentamiento el jugador ganará puntos y recibirá una recompensa en dinero en función al total de puntos que haga en todos los combates. Con esto, obtendremos de nuevo dinero para poder comprar y mejorar, para así preparar para la siguiente semana. Estas partidas serán jornadas de 8 semanas.



1. Interfaz de la jornada de combates antes de realizarse y 2. Después de que se hayan realizado los combates

2.2.2.2 Funcionalidades Administrador

Como se ha mencionado anteriormente, además de sus funciones, el administrador podrá realizar las mismas funciones que el usuario como si fuera un jugador más.

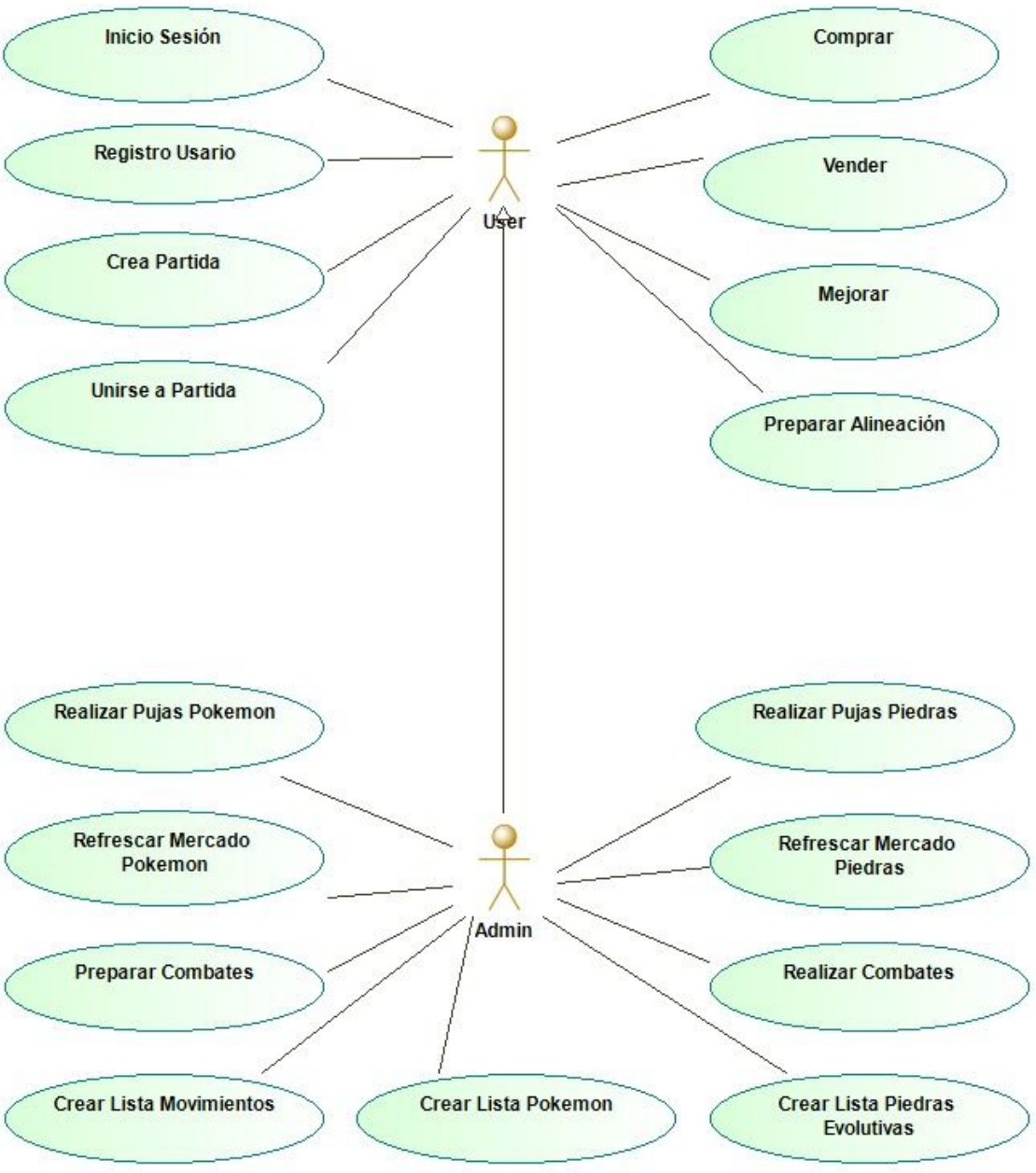


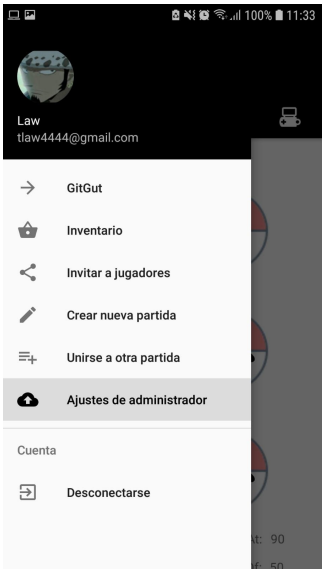
Diagrama de casos de usos de la parte de administrador, heredando las funciones de usuario y teniendo las suyas propias.

Los administradores podrán realizar todas las funciones correspondientes y relacionadas con el servidor que almacena la base de datos. Estas funciones se pueden dividir en 3 bloques:

·**Mercado:** Consultara las pujas de cada usuario en cada mercado (Mercado de Pokémon y de Objetos), mirara cual es el usuario que ha realizado la puja más alta y le añadirá el Pokémon o el objeto a su cuenta. Luego el administrador refrescara las dos listas para que aparezcan nuevos Pokémon y objetos diferentes para pujar.

·**Combates:** Separado en dos, preparar los combates creará los enfrentamientos de cada usuario, todos lucharán contra todos, leerá las alineaciones de cada uno y creará la información correspondiente. Luego, realizar combates, usará la información de estos enfrentamientos que se han creado y una pequeña IA realizará los ataques combate a combate para ver quien gana. Guardará la información de cada enfrentamiento para que el usuario pueda ver cuales han sido los resultados.

·**Crear Base de Datos:** Finalmente, podrá crear las listas principales de las que la aplicación saca la información (Lista de Pokémon, lista de movimientos y lista de piedras evolutivas).

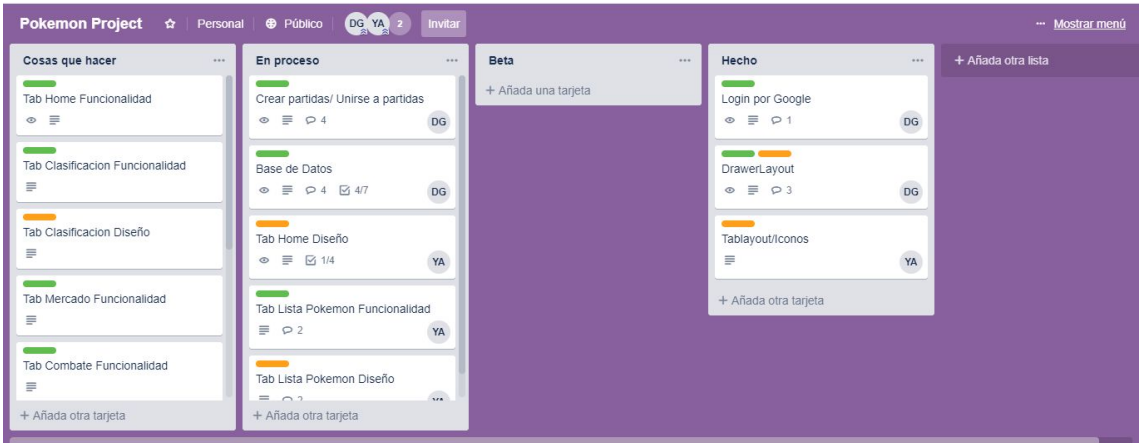


Funcionalidades administrador. 1. Ajustes de administrador solo saldrá si el usuario actual lo es. 2. Pantalla a la que se accedera al clicar en ese botón.

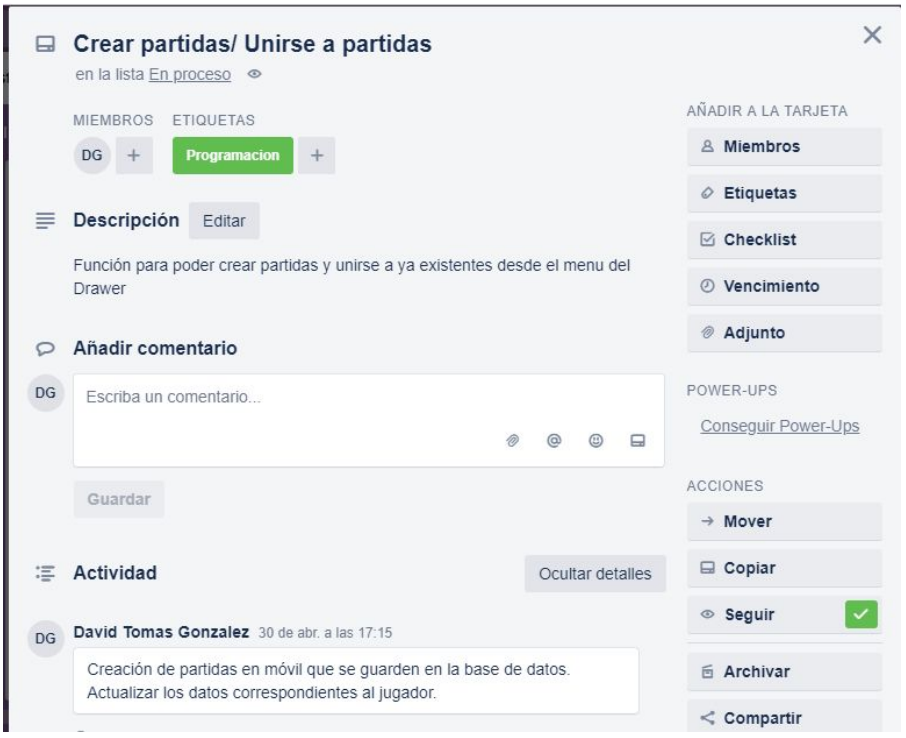
2.3 Desarrollo:

La aplicación ha sido enfocada primero en el aspecto lógico para poder tener toda la información que se necesita y luego la parte gráfica. Primero se ha confeccionado la estructura de la base de datos que se está usando Firestore, una base de datos gratuita de google basada en colección de documentos, y a partir de ello se han creado las interfaces en las cuales se van a manejar los datos. A la vez a través del Modelio, programa para diseñar diagramas UML, se han diseñado en concepto todas las funcionalidades que tendrá la aplicación, tanto a nivel de participante de liga como a nivel de administrador. Todo el desarrollo de código se usa Android Studio, además también se han utilizado Apis ya creadas de pokemon para consultar datos y guardarlos en la base de datos de Firestore.

Para la gestión de reparto de tareas se está utilizando la herramienta Trello, una herramienta que se basa en la idea del Scrum, aunque no se está utilizando un Scrum como tal ya que en el equipo no hay ningún experto, si que se está intentando hacer una separación equitativa de tareas, ir completando y marcando las que ya están hechas y las que se están haciendo, y a la vez se reúne el equipo para comentar cómo está yendo el ritmo de desarrollo y en que se tiene que focalizar.



Tablero de tareas con estructura de Scrum (To Do, Doing Done).



Detalles de una de las tareas con comentarios.

Detalles de una de las tareas con lista de cosas por hacer.

2.3.1 Base de datos

Para escoger la base de datos que se está utilizando se barajaron varias opciones diferentes o combinaciones de ellas con diferentes pros y contras:

-Firebase Realtime Database

-Firebase Cloud Firestore

-Room Database Android

-Realm

-SQL

La primera elección fue fácil, se necesitaba una base de datos en servidor o nube accesible por los usuarios para que se actualicen los datos. Con esto quedaba descartado Room Database y Realm en solitario, ya que son bases de datos que guardan sus datos solo localmente. Se planteó la combinación de estas con Firebase, pero ya que la mayoría de datos se consulta una vez cuando se abre la aplicación se vio innecesario mantenerla en el teléfono gastando memoria.

Con esto quedaban los dos tipos de Firebase y SQL. Se planteó la siguiente pregunta, necesitamos que estos datos estén siempre accesibles para los usuarios, la elección venía entre un servidor propio o el servidor de google siempre activo. Ya que no había gran cantidad de datos a tener en cuenta, se optó por Firebase con su tarifa gratis ya que era suficiente. También es la herramienta con la que había más experiencia.

Entre los dos tipos finales que vemos, se escogió Firestore por comodidad de consulta y lectura, ya que aun estando en beta, funciona

con una estructura mucho más ordenada, por documentos y consultas más fáciles y accesibles.

	Cantidad Almacenamiento	Acceso rápido y simple a Datos	Integridad de datos	Consultas simples y automáticas
Firebase	Limitado por licencia	Si, directo a campos por ruta	Se han de repetir los datos para agilizar las consultas	No
Firestore	Limitado por licencia	No, a través de documentos	Es difícil mantener la integridad de datos	Si
Sql	Ilimitado	Compleja al tener que relacionar tablas	Es una virtud en la integridad de datos	Si

Se ha completado una parte de la base de datos con datos descargado directamente de una api. De esa api se han descargado la lista de todos los pokemons con sus respectivos movimientos, se han modificado esos datos y se han incluido en la base de datos. La api usada ha sido pokéApi, se ha usado esta api porque es la mejor estructura y la que tiene una gran cantidad de información y una accesibilidad muy sencilla.

2.3.2 Servidor:

Para complementar el funcionamiento de la aplicación se tenía la idea de hacer un servidor. La idea principal era que estuviera corriendo todo el tiempo y que cada 24 horas se leyera la base de datos y que se actualizarán los datos según la información recogida. Se encontraron problemas para implementar la lectura de datos a firestore desde una programa java básico. Se estudió hacerlo en javascript pero se desechó por falta de tiempo para aprender un lenguaje nuevo así que se decidió hacer una aplicación complementaria que simulara un servidor ajeno. Esta aplicación se compone de una sola pantalla y varios botones. Con esto se ha ganado en tiempo y además se ha separado las cosas que debería hacer un servidor para así poder implementarlo en un futuro.

Al hacer el servidor en una aplicación de android, se tuvo problemas con la integridad de datos cuando se tenía que modificar varios documentos. Ya que android no permite hacer consultas en el proceso principal de la aplicación que puedan parar o ralentizar el aplicación, te obliga ha hacer procesos en segundo plano, esto daba problemas porque se leían algunos documentos antes que otros y se perdía la integridad de la información. Se consultó este tema con un tercero más experto en el temario, el cual sugirió ayudó a hacer una estructura recursiva para poder así leer los documentos por orden y modificar la información cuando interesaba para así no machacar datos.

Esta aplicación externa, por temas de sencillez y de sincronización ha sido añadida a la app principal pero en una sección solo visible para los creadores de la aplicación. Aunque sigue manteniendo su independencia.

2.4 Pruebas:

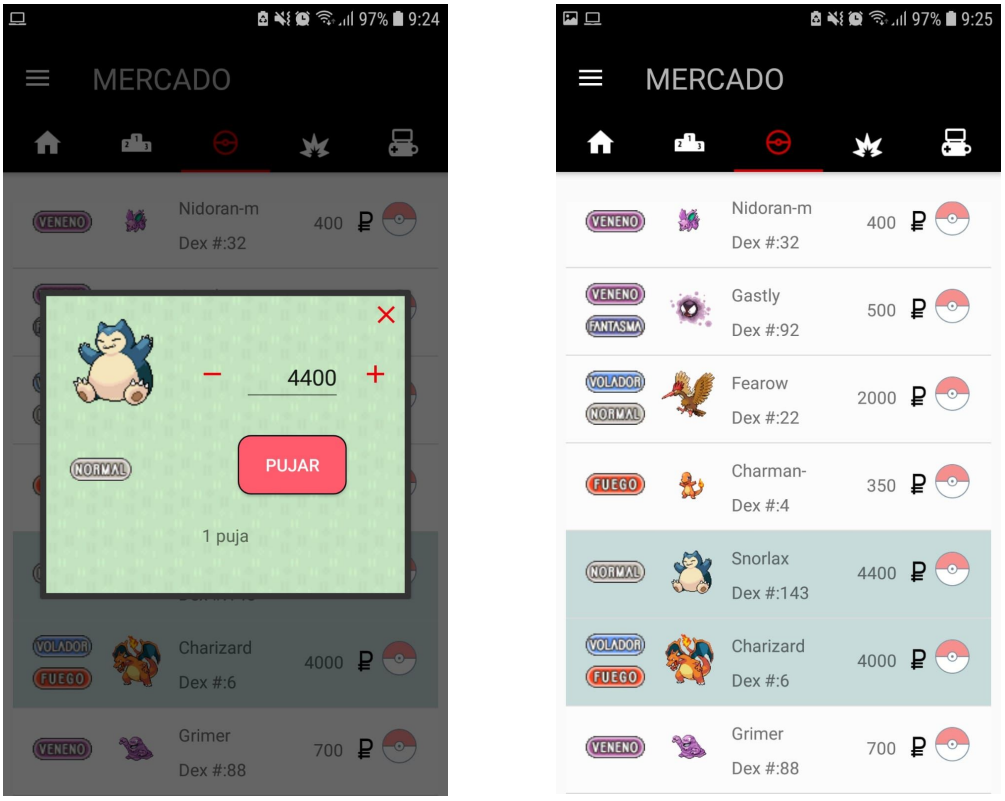
Las únicas pruebas que se han realizado como tal han sido durante el trabajo de la misma aplicación, debugando y viendo si se ejecuta lo que tiene que ejecutarse. Una vez se tenga la estructura funcional básica, se realizarán pruebas simultáneas en diferentes dispositivos para ver si todos se conectan y muestran la información que se prevé.

La pruebas más importantes se realizaron a través de Beta Testers, usuarios que ayudaron a probar la funcionalidad y sincronización de datos en tiempo real.

También se han realizado un par de pruebas de diseño, para ver si el usuario entiende la estructura y es capaz de moverse sin mucha ayuda. La gran mayoría de estas comprobaciones han sido satisfactorias, en el caso contrario, se cambiaba el diseño a un estilo más satisfactorio para el usuario que nos ha aconsejado.

2.4.1 Beta Testers:

Se han realizado varias pruebas con diferentes usuarios y dispositivos para comprobar la eficiencia y sincronización de los datos (Pujas, equipos, alineaciones, usuarios). Con esto se han detectado pequeños bugs de diseño, cosa que ha resultado útil para corregirlos (Dinero infinito, evitar la compra del mismo pokemon, evitar el uso del mismo pokémon), además de encontrar posibles mejoras que se implementaron posteriormente (Detección de puja propia o ver pujas totales).



Imágenes de las mejoras explicadas. 1. Número de pujas del pokemon. 2. Pujas que has realizado se muestran visualmente.

2.5 Lanzamiento:

Esta aplicación está hecha para aprender y en todo caso jugar con amigos, ya que en ningún momento puede ser lanzada porque usa imágenes y datos de una propiedad intelectual con derechos reservados (Pokémon) y ya se ha visto anteriormente otras empresas o grupos de fans crear juegos sin ánimo de lucro y ser denunciados y clausurados por Pokémon Company, Nintendo o similar.

En caso de quererse publicar habria que sustituir todos los datos referentes a Pokémon (Pokémon, movimientos, piedras, estadísticas, evoluciones, etc.), lo cual resultaría en casi un cambio de aplicación completa, ya que muchas funciones y cálculos están basados en esos datos.

Así que el “lanzamiento” será vía Whatssap a usuarios conocidos que quieran jugar que a la vez servirán de testers continuos para comprobar su funcionamiento y rendimiento.

3. Conclusiones

Durante la realización de este proyecto se han aprendido muchísimos aspectos sobre la programación de una aplicación, entre ellas, diseño, base de datos, programación etc.

Es cierto que se planteó una gran cantidad de trabajo al empezar y no se contaba con lo que había que aprender para realizar todos los puntos pensados, entre ellos, utilizar recursividad de manera compleja al leer y escribir datos, el entender el funcionamiento y sincronización de información en la base de datos utilizada o simplemente temas de diseño y estructura.

Aun contando todo esto, se ha realizado la gran mayoria de trabajo planteado, con todas sus funciones. Es verdad que faltan detalles que se plantearon al principio (Como vender Pokémon, mercado para comprar ataques y cambiarlos a los Pokémon o modificar ajustes de una partida, ya sea cambiando el nombre, el dinero inicial o simplemente eliminarla), pero a medida que avanzaba el tiempo, parecía que se iba a llegar a mucho menos del producto final con el que ha finalizado el proyecto.

Después de realizar el proyecto, seguramente utilizar otra base de datos, aunque se hubiera tenido que aprender como usarla, hubiera sido más sencillo, ya que Firestore, aunque ha sido muy útil y era la base de datos en la que el equipo tenía más experiencia, se han encontrado problemas que quizá no hubieran pasado con otra, entre ellos, un límite por Google de consultas diarias y el tener que cambiar los modelos cada vez que se cambiaba un poco la estructura de la clase en la aplicación.

A pesar de ser la primera aplicación seria a la que se le dedica tanto tiempo, el equipo ha organizado y repartido el trabajo satisfactoriamente. Cada punto y funcionalidad que se planteaba para introducir en el proyecto, se discutió previamente como realizarlo, quien lo hacía y cuánto era la carga de trabajo para ver si otro miembro hacia el equivalente.

A pesar de no coincidir en algunos puntos, se debatía para ver que se hacía finalmente, y siempre uno cedía ya fuera convenciendo al otro, asumiendolo para empezar a trabajar y si salia algun problema resolverlo a su manera o porque después de modificar la propuesta, ambos estaban de acuerdo.

Se siguió un seguimiento de trabajo cada dia para ver en qué tarea estaba trabajando cada uno, si alguien necesitaba ayuda por estar atascado o que no había manera de realizarlo, o por si se dedicaba demasiado tiempo, enfatizar ambos en la misma tarea para acabarla antes.

Como resultado, se acabó trabajando conjuntamente de manera muy satisfactoria y compenetrada, puliendose mutuamente y supliendo los fallos o errores de cada uno, para llegar al proyecto final.

4. Glossario

Android Studio: Entorno de desarrollo integrado oficial para la plataforma Android.

Firebase: Plataforma de Google para el desarrollo y mantenimiento de aplicaciones web y bases de datos.

Firestore: Base de datos flexible y escalable para la programación en servidores y móviles. Opción de Firebase.

Pokémon: Es una palabra que su origen es la fusión de Pocket Monster, que significa monstruos de bolsillo.

Biwenger, Comunio, Futmondo, Fantasy LCS: Aplicaciones de referencia, gestores de ligas de fútbol y de League of Legends. Consiste en organizar y controlar un equipo para ganar dinero y mejorar.

League of Legends: Videojuego de género MOBA.

MOBA: Género de videojuego, multijugador de arena de batalla en línea.

Room Database: Base de datos local con funcionamiento basado en SQL.

SQL: (or sus siglas en inglés Structured Query Language; en español lenguaje de consulta estructurada) Lenguaje de específico del dominio utilizado en programación diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.

JavaScript: Es un lenguaje de programación, al igual **que** PHP, si bien tiene diferencias importantes con éste. **JavaScript** se utiliza principalmente del lado del cliente (es decir, se ejecuta en nuestro ordenador, no en el servidor) permitiendo crear efectos atractivos y dinámicos en las páginas web.

API: (siglas de 'Application Programming Interface') es un conjunto de reglas y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas y recoger y mostrar información de un tercero sin tener que programarlo en el móvil.

IA: (siglas de Inteligencia Artificial), conjunto de directrices o reglas que sigue una máquina o programa para ejecutar sus funciones.

Modelio: Programa con el que se han diseñado los diagramas UML de clases y casos de uso.

UML: (por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para construir un plano de funcionamiento de un software o sistema.

Trello: Página web para organizar y preparar una estructura de trabajo por tareas. Muy similar a una pizarra de tareas del método Scrum.

Scrum: Metodología de trabajo ágil en el que se aplican de manera regular una serie de prácticas y directrices para trabajar de manera organizada y colaborativa.

Drawer: Menu lateral deslizante, usado normalmente como sistema de navegación, para acceder a otras pantallas desde una interfaz que se abre a parte de la principal.

Beta Tester: Usuario que prueba programas o similar cuyos ejecutables están pendientes de terminar su fase de desarrollo, que tienen un funcionamiento completo, pero que aún no están totalmente terminados presentando fallos de diversos tipos o características pendientes de implementar.

5. Bibliografía

Trello, herramienta para organizar las tareas, división de trabajo y llevar un control de lo que se va realizando: https://trello.com/b/OkNWR99E/pokemon-project

PokéApi, api utilizada para descargar y guardar la información correspondiente a cada Pokémon (estadísticas, imagenes, evoluciones, movimientos): https://pokeapi.co/

Firebase, base de datos utilizada durante todo el proyecto, consultas de documentos, usuarios y conexiones: https://firebase.google.com/

Documentación Firebase, página utilizada para consultar tutoriales, guías y maneras de realizar las operaciones o funciones que se querian implementar: https://firebase.google.com/docs/guides?hl=es-419

StackOverflow, consultada a causa de errores y problemas que aparecían durante el desarrollo: https://stackoverflow.com/

Documentación Android, documentación oficial de Android Studio, para buscar guías, ejemplos o maneras de implementar funciones de Android: https://developer.android.com/docs

Github, sistema de control de versiones utilizado como copia de seguridad y con el objetivo de compartir el código dividido entre los trabajadores: https://github.com/Law44/PokemonProject

Github.io de Gerard Falcó, pagina utilizada durante el año como enseñanza por parte del profesor, utilizada como referencia para implementar funciones de cosas explicadas: https://gerardfp.github.io/#0

Icono Aplicación, icono utilizado para la aplicación y en algunas pantallas, libre de copyright: https://www.freeiconspng.com/img/45338

Imagen piedras evolutivas, imagen para las piedras evolutivas, libre de copyright:https://www.pngfind.com/mpng/ibJwRTm_oras-all-mega-stones-code-pidgeot-beedrill-shiny/

Logo Firebase y Logo Android, utilizados en esta memoria:
https://firebase.google.com/images/brand-guidelines/logo-standard.png?hl=es-419
https://i0.wp.com/www.arcanstudios.com/wp-content/uploads/2017/10/android-studio-logo.png?ssl=1

Alertas personalizadas, tutorial de como personalizar alertas, utilizado para la alerta de carga:
https://github.com/ttasterisco/CustomProgressDialogExample

Gif de carga, gif utilizado para mostrar en la la alerta de carga de datos:
https://thenewbarkcodex.files.wordpress.com/2018/04/tumblr_nu4n8oxf6b1r7tm2fo1_500.gif

Fondo Alertas, imágenes utilizadas como fondo en las alertas o modals que aparecen en la aplicación:
https://bulbapedia.bulbagarden.net/wiki/Pok%C3%A9mon_Storage_System

Librerías Externas:

-Libreria utilizada para añadir una barra de busqueda:
https://github.com/MiguelCatalan/MaterialSearchView

-Pasos seguidos para añadir las librerías de Firebase y Firestore:
https://firebase.google.com/docs/firestore/quickstart?hl=es-419

-Pasos seguidos para añadir la librería del RecyclerView:
https://developer.android.com/guide/topics/ui/layout/recyclerview

6. Agradecimientos

Esta aplicación y todas sus funciones no habían llegado al punto del proyecto que encontramos sino llega a ser por ciertas personas que han ayudado a su desarrollo, ya sea en cualquier campo, como diseño, arte o programación. En esta lista encontramos a:

·**Gerard Falcó,** profesor del curso, sin su ayuda y experiencia varias funciones y operaciones no se habrían realizado o se habrían enfocado de una manera mucho más compleja e ineficiente.

·**Fernando Porrino**, profesor también del curso, gracias a sus consejos y explicaciones, toda esta memoria y parte del trabajo se estructuró de una manera útil.

·**David Delgado**, amigo y compañero del curso, ha intervenido varias veces en el diseño del funcionamiento de la aplicación, dando consejos de como hacerlo y mejorar. Además, fue el que nos sugirió una idea de este proyecto.

·**Adrià Gregoriano**, también amigo y compañero del curso, ha ayudado a realizar y editar imágenes que se utilizan en la aplicación.

Y a las siguiente personas por ayudarnos como Beta Testers:

-Ales de Toledo

-Ricard Martín

-Daniel Ferrero

-Albert Fernández

-Alberto Garcia

-Alberto Gutierro

-Catalin Trandafir

-David Delgado

-Adrià Gregoriano

7. Anexos

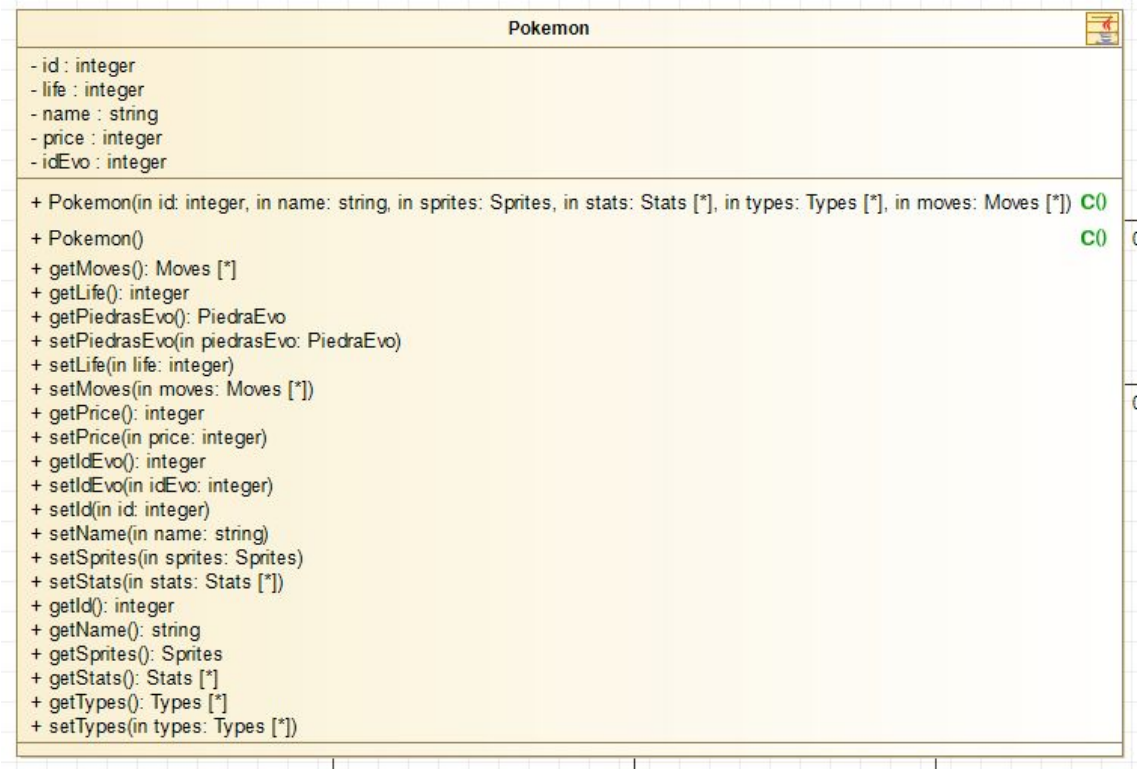


Diagrama de clases, bloque pokemon, 1/3

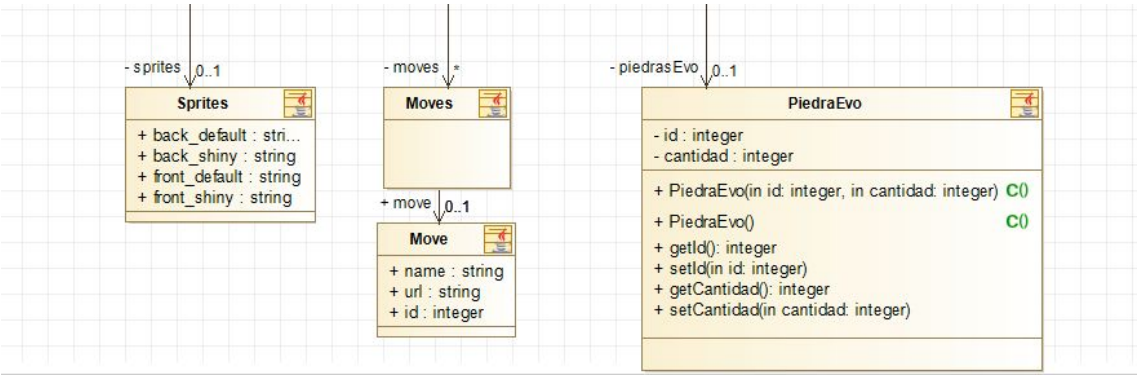


Diagrama de clases, bloque pokemon, 2/3

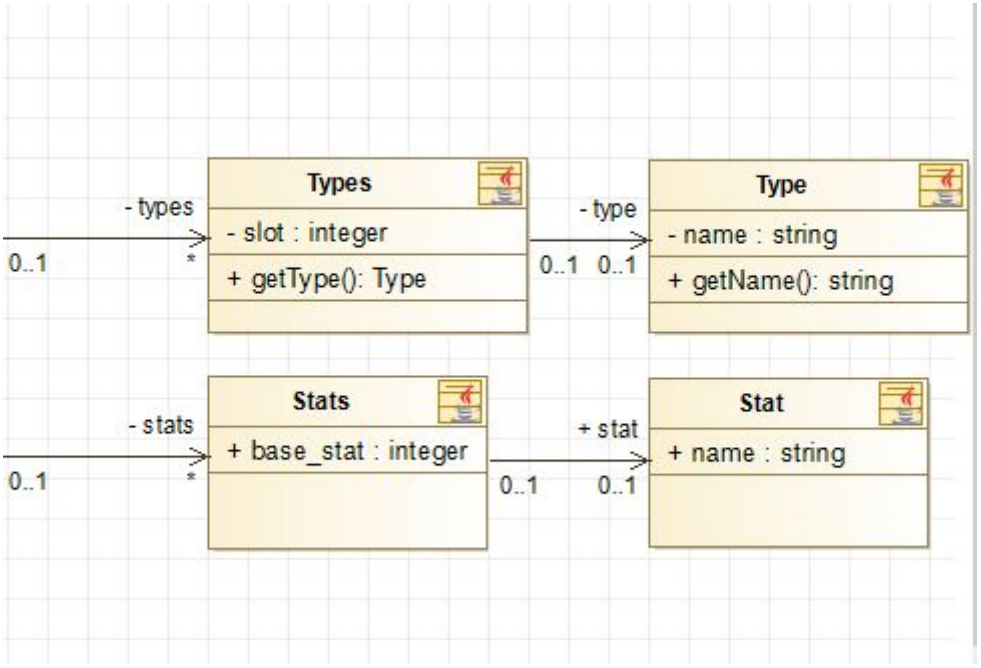


Diagrama de clases, bloque pokemon, 3/3

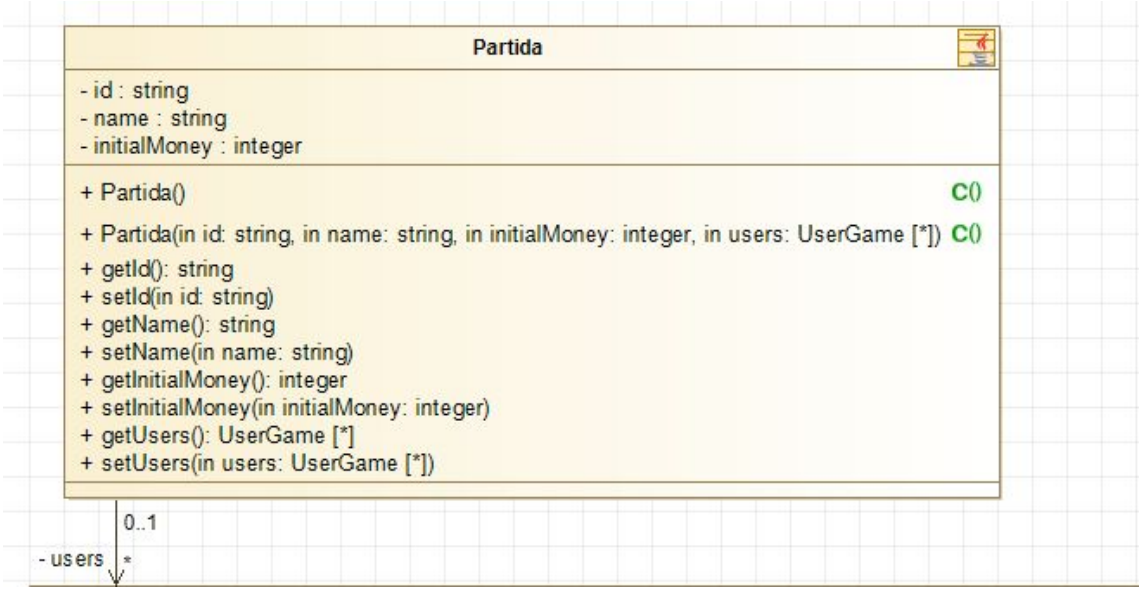


Diagrama de clases, bloque Usuarios y partida, 1/3

Diagrama de clases, bloque Usuarios y partida, 2/3

Diagrama de clases, bloque Usuarios y partida, 3/3