UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de un Trivial para Android.

TRABAJO FIN DE GRADO
Grado en Ingeniería Informática

Alumno: Francisco José Abellán Romero
Tutor: Jose Ángel Carsi Cubel
Curso 2016-2017

Resumen

Con el auge de las tecnologías móviles en la actualidad, la portabilidad que las caracteriza e Internet accesible a gran parte del público, se ha propuesto como objetivo de este proyecto desarrollar una aplicación que permite jugar al popular juego de mesa **Trivial Pursuit** a plataformas móviles con el sistema operativo Android que además permitirá jugar partidas multijugador en línea. Con la aplicación cualquier jugador interesado puede jugar una partida en cualquier momento y lugar.

Como la aplicación está dirigida a cualquier tipo de usuario, se ha enfatizado en el propósito que sea una aplicación muy sencilla, intuitiva e interactiva.

En este documento se mostrarán las diferentes fases de desarrollo del proyecto para cumplir el objetivo acordado. Se evaluarán las aplicaciones del mercado con un propósito similar y a continuación se enseñarán todas los pasos necesarios para implementar el producto final.

Palabras clave: Android, juegos, java.

Abstract

Nowadays with the rise of mobile technologies, the portability that defines them and Internet accessible to the most of the population, the development of an application that lets the users play the popular board game ***Trivial Pursuit*** using Android devices has been proposed to be carried out. This application will also allow the players to play online multiplayer matches. With this application any interested player can play a match at any moment and anywhere.
As the application is addressed to any kind of user, it must be very straightforward, intuitive and interactive.
In this document the different stages of development of the project will be shown to fulfill the marked objective. Similar applications in the market will be evaluated and then all the steps done to implement the final product will be explained.

Keywords: Android, games, Java.

1. Introducción

En este apartado se justificará la elección del trabajo y se analizarán las aplicaciones existentes similares en el mercado.

1.1. Motivación

El mercado de aplicaciones para dispositivos de teléfonos móviles ha sufrido un auge en aproximadamente estos últimos diez años en los cuales los dispositivos sólamente se empleaban para realizar llamadas, mensajes cortos de texto (SMS), fotografías, vídeos y juegos muy sencillos.

Tras la llegada de los teléfonos inteligentes (Smartphones) que mejoraron enormemente las prestaciones de los dispositivos facilitaron enormemente la posibilidad de realizar aplicaciones para dichos dispositivos. Entre ellas, juegos con sencillas mecánicas. Asimismo, hubo una variación entre los dispositivos móviles, no existen solamente los teléfonos móviles que ocupan aproximadamente la palma de la mano, sino otros dispositivos como los denominados Tabletas (Tablets) que presentan más tamaño y potencia.

Además con la aparición de los teléfonos inteligentes, se creó la posibilidad de conectarse a Internet tanto de redes wifi como de redes móviles. Además las compañías telefónicas empezaron a ofrecer Internet móvil a precios asequibles, lo que desarrolló el mercado de aplicaciones para dichos dispositivos mediante tiendas virtuales y además produjo que se pudiera jugar multijugador en dichos dispositivos.

Android es el sistema operativo mayoritario en estos dispositivos, concretamente en el 92% de ellos en los vendidos entre diciembre 2012 y febrero 2013 en España [1] con lo que centrarse en este sistema operativo resulta interesante para llegar a un gran número de consumidores.

Teniendo en cuenta lo expuesto anteriormente: dispositivos portables, conexiones a bajo precio a Internet y que la mayoría de población cuenta con estos dispositivos, se consideró interesante llevar el juego de mesa Trivial Pursuit a dispositivos móviles.

Se eligió el Trivial debido a que se trata de un popular juego de mesa muy conocido y con mecánicas sencillas con lo que cualquier persona podría jugar sin dificultad, por tanto, ser un juego accesible a cualquier usuario.

La aplicación ofrecería a los usuarios el poder jugar partidas en el dispositivo y también la posibilidad de jugar partidas contra cualquier jugador que tenga una conexión de Internet, además se debe destacar la portabilidad de los dispositivos móviles que permite jugar partidas en cualquier momento y lugar a diferencia del juego físico que restringe la posibilidad de jugar al juego a menos que se posea la versión física y los jugadores se encuentren la misma localización. También añadir que como aplicación móvil cualquier partida puede ser guardada y retomada casi al instante.

El objetivo de este TFG es construir esta aplicación para jugar al Trivial con los requisitos expuestos en el punto anterior, es decir, permitir jugar partidas en el propio dispositivo y partidas en línea contra otros jugadores y mostrar cómo se fue llevando a cabo.

1.2. Estado del arte

Al ser el trivial un juego bastante popular, existen aplicaciones en el mercado que comparten el mismo propósito. Se analizarán a continuación dos aplicaciones populares que se acercan al máximo al propósito de trasladar el juego de mesa a dispositivos móviles de la manera más fiel posible.

Atríviate:

Este juego consta en una buena adaptación trivial de mesa, sin embargo, no representa fielmente el tablero del trivial, lo que puede no gustar a los usuarios más exigentes: no existen las casillas que se dirigen hacia el centro, con lo que se encuentra limitado en opciones de movimientos, siempre habrá para elegir solamente dos direcciones. Además existen siete “categorías” de quesitos, a diferencia del trivial de mesa original que son seis.
Sin embargo, el mayor problema actual de esta aplicación es que se encuentra “abandonada” según los comentarios de los usuarios.

Figura 1: Imagen del juego Atríviate

Trivial Cuestionados:

Esta aplicación es menos conocida que la anterior, sin embargo se podría destacar que al igual que la anterior aplicación presentada, no existen casillas que se dirijan hacia el centro con lo que no es igual al Trivial Pursuit original. En esta aplicación existen siete “categorías” de juego en lugar de las seis originales.

Figura 2: Imagen promocional del juego Trivial Cuestionados.

1.3. Estructura de esta memoria de trabajo fin de grado

- En el siguiente apartado, especificación de requisitos, se describe cómo se ha planteado inicialmente la aplicación, que componentes contiene, el alcance del proyecto, bajo qué restricciones de hardware o software se encuentra la aplicación...

- El apartado de análisis contiene la descripción de las entidades que forman parte de la aplicación junto su funcionalidad.

- El apartado de diseño explica el trabajo previo que se ha realizado para poder llevar a cabo la fase de implementación con éxito.

- El apartado de implementación detalla qué tecnologías se han empleado para el desarrollo de la aplicación desarrollada y además explica qué secciones del trabajo desarrollado han resultado más destacables.

- En el apartado de pruebas define como estas se han llevado a cabo.

- En el apartado de conclusiones y trabajos futuros se detalla, entre otros, los resultados obtenidos, la satisfacción personal del propio proyecto y posibles mejoras que se podrían realizar.

- Los apartados de apéndices contienen información adicional de la aplicación en sí. El primer apéndice denominado Manual de Usuario explica cuáles son las opciones de juego y cómo se pueden llevar a cabo teniendo en cuenta que el usuario no ha utilizado nunca la aplicación y el segundo apéndice define las vías por las cuales se podría obtener un beneficio económico de la propia aplicación.

2. Metodología empleada.

La metodología de desarrollo que se ha empleado ha sido por prototipado. Esta metodología permite crear prototipos, que se trata de pequeñas partes del desarrollo que no contienen toda la funcionalidad total, pero sí lo necesario para poder evaluar dicha parte. Al tratarse de un entorno sin experiencia previa, esta metodología fue útil para poder realizar continuos cambios conforme se iba aprendiendo y desarrollando el proyecto. También fue útil para cambiar aspectos de la interfaz gráfica si finalmente el resultado obtenido no se consideraba adecuado.

A continuación se mostrará un resumen con la planificación prevista y el resultado real en horas aproximadas. Al tratarse de un entorno nuevo para nosotros, también se tuvo que invertir tiempo para contar con un manejo básico.

Tarea	Horas planificadas	Horas reales
Instalación y manejo básico de Android Studio	20	15
Especificación requisitos	15	15
Análisis y diseño	50	50
Construcción interfaz gráfica básica	30	30
Primera versión del juego, implementación juego por turnos	40	20
Aprendizaje y configuración de Google Play Services	30	50
Segunda versión del juego, funcionalidad en línea	20	40
Mejoras interfaz gráfica	30	30
Tercera versión del juego, modo para un jugador	10	10
Revisión aplicación	30	20
Revisión interfaz gráfica	20	20
Revisión y finalización memoria	35	35
Total	330	335

Como se puede observar, se ahorraron horas principalmente en la primera versión del juego, ya que al tener el análisis y el diseño claros, el modo de proceder fue fluido y las correcciones que se realizaron fueron rápidas. Por contra, donde más tiempo se perdió con diferencia (denotado con un color distinto en la tabla) fue en lo relacionado a los servicios en

línea y Google Play Services. Esto fue en parte debido a un error que surgía al intentar conectarse que más tarde se descubrió se producía porque era necesario contar con una cuenta de desarrolladores de Google para iniciar sesión. También se tuvo que invertir horas para entender el funcionamiento básico de cómo se desarrolla una partida por turnos y su intercambio de información y costó más de lo esperado. Sin embargo, no se han contado horas que se han dedicado a aprendizaje con tareas ajenas al proyecto o mejoras puramente estéticas tanto a nivel de código o a nivel de interfaz.

3. Especificación de requisitos

En esta sección se describe el producto en detalle, la funcionalidad necesaria, los tipos de usuarios y los productos necesarios para utilizar el sistema a desarrollar.

3.1. Introducción

A continuación se definitirá los objetivos de esta especificación.

3.1.1. Propósito

El propósito de esta sección es definir los requisitos que debe cumplir este sistema a desarrollar.

3.1.2. Ámbito del sistema

El producto a desarrollar consiste de una aplicación para poder jugar al Trivial en un sistema operativo Android. Además las partidas como mínimo deben tener dos jugadores y como máximo seis.

A diferencia de otras aplicaciones del mercado examinadas en el apartado Estado del arte, dichas aplicaciones ofrecen la posibilidad de jugar al Trivial de mesa, sin embargo se alejan ligeramente del clásico juego de mesa y no se ofrece la opción de jugar al trivial clásico en dispositivos móviles, con lo que los jugadores que no puedan reunirse para jugar o no tengan el juego físico disponible en ese momento, podrán hacerlo con la aplicación en cualquier momento y lugar. Además podrán disponer una partida para retomarla más adelante de forma sencilla. También se ofrece la opción de jugar contra jugadores controlados por el sistema.

3.1.3 Definiciones, acrónimos y abreviaturas

Tablero: Simulación del tablero del trivial, el cual estará formado por una casilla destino, seis casillas quesito, varias casillas relanzar dado y varias casillas base.

Ficha: Representa la situación física en el tablero del jugador. Además contendrá los quesitos obtenidos.

Quesito: Al igual que en el Trivial clásico de mesa, se obtiene al responder. Cuando un jugador tiene todos los quesitos de todas las categorías, y su ficha llega a la casilla final, gana la partida.

Categoría: Definen un ámbito del conocimiento determinado, son de seis tipos con su color asociado, las cuales son: Geografía (Azul), Espectáculos (Rosa), Arte y Literatura (Amarillo), Historia (Marrón), Ciencias y Naturaleza (Verde) y finalmente Deportes (Naranja)

Casillas: Espacio físico del tablero donde las fichas pueden moverse, existen cuatro tipos de casillas, casilla base, casilla quesito, casilla relanzar dados y casilla final. Cada casilla base pertenece a una categoría.

Casilla base: En esta casilla, si se accede la ficha del jugador el sistema lanzará una pregunta al jugador, si la contesta correctamente podrá volver a tirar los dados y mover la ficha de nuevo. Si responde incorrectamente, pasará el turno al siguiente jugador. Estas casillas pertenecen a una sola categoría.

Casilla quesito/premio: Esta casilla se comporta como la casilla base con el comportamiento adicional que al responder correctamente la pregunta, se obtendrá un quesito de la categoría a la que pertenece la pregunta.

Casilla relanzar dados: En esta casilla, al llegar un jugador, debe volver a lanzar los dados.

Casilla final: En esta casilla, al alcanzarse con los quesitos de todas las categorías, se gana la partida. En caso de llegar sin todos los quesitos, se cederá el turno al siguiente jugador.

Pregunta Respuesta: Consiste en una pregunta y cuatro posibles respuestas, de la cual solamente una es correcta. Todas las preguntas son de una sola categoría determinada.

Dado: Dado clásico de juegos de mesa con seis caras, con los números del uno al seis incluidos. Cuando se lanza el jugador tiene que mover su ficha tantas casillas adyacentes y consecutivas como indica el resultado del dado.

3.1.4. Referencias

Para la realización de este documento, se han consultado los siguientes documentos:

- Especificación de Requisitos IEEE 830 de 1998
- Especificación de Requisitos IEEE 830 de 22 de Octubre de 2008

3.2. Descripción General

3.2.1. Perspectiva del producto

El producto consiste en una aplicación para jugar al Trivial uno o varios jugadores para dispositivos con el SO de Android. Dichas partidas serán multijugador, como mínimo dos jugadores, y se podrá jugar a la partida varios usuarios en el mismo dispositivo o varios usuarios en distintos dispositivos mediante una conexión de Internet. También destacar que

las partidas que se realicen en el mismo dispositivo, se podrá jugar además contra jugadores controlados por el sistema, conocidos como CPU.

3.2.2. Funciones del producto

Debido a que es un producto orientado a jugar un juego de mesa, las funciones del producto corresponderá al funcionamiento del propio juego y a los procesos correspondientes a partidas y usuarios.

Gestión de cuentas de usuario: Los usuarios deben poder iniciar sesión con su cuenta Google Mail (gmail) para las funciones en línea. Con lo que incluye resumidamente las siguientes funciones:

- Iniciar sesión en línea

Gestión de partidas: Se debe permitir a los jugadores poder crear partidas y salir de partidas. El sistema debe de comprobar cuando una partida ha finalizado y notificar a los jugadores. Además se debe ofrecer la posibilidad de añadir los jugadores que se deseen o invitarlos a partidas. Incluye:

- Crear partida
- Invitar jugadores
- Aceptar/Rechazar invitaciones
- Cargar partida
- Abandonar partida

Gestión de posiciones del juego: Se debe permitir a los jugadores seleccionar un resultado aleatorio del dado cuando corresponda, mover la ficha cuando corresponda, a las casillas permitidas. El sistema debe calcular las casillas que son permitidas a partir del resultado del dado, con lo que contiene:

- Lanzar dado
- Calcular destinos
- Mover ficha

Gestión preguntas y respuestas: El sistema se debe encargar de cargar las preguntas con sus correspondientes respuestas, además debe presentar las preguntas y posibles respuestas correctas cuando el jugador llegue a una casilla apropiada y debe comprobar si la respuesta proporcionada por el jugador es correcta o no. También debe proporcionar quesitos a los jugadores. Por tanto incluye:

- Cargar preguntas y respuestas
- Mostrar preguntas y respuestas

- Seleccionar respuesta
- Obtener si es correcta o no la respuesta
- Proporcionar quesito

Gestión de turno de los jugadores: El sistema deberá ocuparse que cuando termine el turno de un jugador, juegue el siguiente jugador.

- Rotar turno entre jugadores

3.2.3. Características del usuario

En esta aplicación existirán dos tipos de usuario, los usuarios que tienen una conexión a Internet y que tienen una cuenta de Google Mail válida y los usuarios que no. Los usuarios que no cumplan las condiciones de Internet o que no tengan una cuenta válida no podrán acceder a las funciones en línea del sistema. Los usuarios que sí, podrán acceder a toda la funcionalidad que ofrece el sistema.

3.2.4 Restricciones generales

El sistema requerirá de un dispositivo con el SO (Sistema Operativo) Android para poder ser ejecutado. Además el nivel de API deberá ser igual o superior a 14. Además para acceder a las funciones en línea se debería contar con una cuenta Google Mail y tener instalado en el dispositivo Google Play Games Services.

3.2.5 Dependencias del sistema

El sistema requiere de la API de Google Play Game Services para los servicios en línea.
Al tratarse de una compañía especializada con bastante prestigio es improbable que pueda no estar disponible en un futuro cercano más allá algún fallo puntual.

3.3. Requisitos específicos

El producto deberá gestionar todo el proceso que constituye una partida al Trivial, además de la gestión de partidas y de los jugadores. Se enumerará las funciones a continuación.

3.3.1. Requerimientos funcionales.

3.3.1.1. Gestión de cuentas de usuarios

Iniciar sesión en línea	
Introducción	Su función es permitir que Google compruebe la cuenta del usuario para permitir utilizar sus servicios en línea.

Entrada	La cuenta de usuario Google Mail del usuario. Además requiere que el usuario esté conectado a Internet.
Proceso	Google comprueba si los datos son correctos
Salida	Éxito si los datos son correctos y no ha habido ningún problema o fallo si no lo son o hay un problema de autenticación o de red.

3.3.1.2. Gestión de partidas

Crear partida	
Introducción	Se encarga de crear la partida con todos los componentes necesarios para jugar.
Entrada	La lista de los jugadores que desean jugar al juego.
Proceso	Se asignan fichas a los jugadores, se almacenan los jugadores y se crea el tablero con sus casillas, además se posicionan las fichas en la casilla central.
Salida	La partida creada correctamente.

Invitar jugadores	
Introducción	Se ocupa de enviar invitaciones a los jugadores que deseen jugar una partida.
Entrada	Los jugadores que se desean invitar.
Proceso	Google Play se encarga de enviar las invitaciones a los jugadores
Salida	Los jugadores reciben las invitaciones correctamente.

Aceptar/Rechazar invitaciones	
Introducción	Se ofrece la posibilidad de aceptar o rechazar invitaciones
Entrada	La invitación que se desee aceptar/rechazar
Proceso	Google Play tramita el proceso necesario para aceptar o rechazar solicitudes.
Salida	La solicitud aceptada o rechazada correctamente.

Cargar partida	
Introducción	Se encarga de ofrecer la posibilidad de cargar una partida ya existente.
Entrada	La partida que se desee cargar.
Proceso	A partir de los datos de la partida, se genera una nueva partida cargando los jugadores, las fichas en la casilla que corresponda y los quesitos obtenidos.
Salida	La partida cargada correctamente.

Abandonar partida	
Introducción	Su función ofrece la posibilidad de abandonar la partida.
Entrada	La partida que se desee abandonar.
Proceso	Google Play tramita el proceso necesario para abandonar la partida. En caso que sea una partida local, el sistema se encarga de ello.
Salida	La partida abandonada correctamente.

3.3.1.3. Gestión de posiciones del juego

Lanzar dado	
Introducción	Se obtendrá un número al azar del 1 al 6, inclusive, que determinará qué número de casillas consecutivas avanzará la ficha del jugador.
Entrada	Jugador activo.
Proceso	Se obtendrá un número al azar del 1 al 6 que determinará qué número de casillas consecutivas avanzará la ficha del jugador.
Salida	El número aleatorio entre 1 y 6.

Calcular destino

Introducción	Calcula de las casillas destino posibles de la ficha a partir del lanzamiento del dado.
Entrada	Posición de la ficha del jugador activo y número resultante del lanzamiento del dado.
Proceso	Se calculará las casillas consecutivas adyacentes a partir de la posición y el número.
Salida	Las posibles casillas destino, para que el jugador elija una.

Mover ficha	
Introducción	Se encarga de cuando el jugador activo elige, de los posibles destinos para su ficha, uno y se moverá la ficha a dicho destino.
Entrada	Ficha del jugador activo y destino indicado por el jugador.
Proceso	Se mueve la ficha al destino posible elegido por el jugador.
Salida	La ficha en el destino elegido.

3.3.1.4. Gestión de preguntas y respuestas

Cargar preguntas y respuestas	
Introducción	Su función es cargar las preguntas y respuestas que se utilizarán para jugar a a la aplicación.
Entrada	La base de datos que contiene los datos correspondientes a preguntas y respuestas.
Proceso	Se añade la información de la base de datos al sistema.
Salida	Las preguntas y respuestas cargadas en el sistema correctamente en caso de éxito, o en caso de fallo, se informará al usuario y le sugerirá que reinstale la aplicación.

Mostrar preguntas y respuestas	
Introducción	Se encarga de cuando el jugador llega a una casilla base o quesito, aparece en pantalla una pregunta con sus correspondientes respuestas para que el jugador elija una respuesta.

Entrada	El jugador activo y la categoría de la casilla base o quesito.
Proceso	Se elige aleatoriamente una pregunta respuesta que pertenezca a la categoría de la casilla que ha llegado el usuario.
Salida	La pregunta con sus correspondientes respuestas en la pantalla.

Seleccionar respuesta	
Introducción	La función es que a partir de las preguntas con sus correspondientes posibles respuestas correctas, el jugador selecciona una respuesta.
Entrada	El jugador activo, la pregunta respuesta activa y la respuesta seleccionada (En caso que el jugador no haya seleccionado ninguna en el límite de tiempo, se considerará que la pregunta ha sido respondida incorrectamente)
Proceso	Se comprueba si la respuesta indicada es correcta o no.
Salida	Dependiendo si el jugador ha contestado correctamente o no, en el juego se producirá una acción u otra.

Proporcionar quesito	
Introducción	Consiste en asignar el quesito al jugador activo.
Entrada	El jugador activo y la categoría a la cual pertenece la casilla quesito en la cual se encuentra la ficha del jugador.
Proceso	Se asigna el quesito de la categoría al jugador
Salida	La ficha del jugador activo con el quesito obtenido

3.3.2.5. Gestión de turno de los jugadores

Rotar turno entre jugadores	
Introducción	Se encarga en ceder el turno al siguiente jugador.
Entrada	El jugador activo y la lista ordenada de jugadores
Proceso	Se avanza al jugador siguiente de la lista.
Salida	El siguiente jugador.

3.3.2. Requisitos de la interfaz externa

Con requisitos de interfaz externa, se refiere a las características visuales que deberá cumplir el sistema a desarrollar.

Interfaz de usuario

Al iniciar el sistema, se debe proporcionar un menú para elegir que modalidad se desea jugar, en las pantallas a continuación se debe proporcionar las herramientas adecuadas para seleccionar una partida o crear una nueva añadiendo o invitando los jugadores a participar. En un juego en línea además se debe permitir poder rechazar invitaciones.

En una partida tiene que aparecer el tablero de juego, las fichas en el tablero y el jugador que se encuentre realizando su turno y los quesitos que tiene cada jugador. Por razones de espacio y para clarificar, los quesitos pueden estar en otra pantalla accesible de manera instantánea.

Interfaz hardware

Cualquier dispositivo que cumpla con las especificaciones software mencionadas en el apartado 2.2.4. Importante resaltar que la interfaz gráfica sólamente se encontrará optimizada para el tamaño de pantalla de teléfonos inteligentes y *Tablets*.
Se requerirá conexión a Internet para los servicios en línea.

Interfaz software

Cualquier dispositivo con sistema operativo Android en el cual se pueda instalar Google Play Game Services para disfrutar de los servicios en línea.

3.3.3. Requisitos de rendimiento

El único requisito de rendimiento a tener en cuenta es, que para mejorar la experiencia de los usuarios, las interacciones entre el usuario del sistema deben ser prácticamente instantáneas, como máximo tomar 500 milisegundos en mostrar el resultado esperado.

3.4. Restricciones de diseño

3.4.1. Estándares cumplidos

Al tratarse de una adaptación de un juego de mesa, se han incluido los conceptos que se utilizan para que el usuario familiarizado con el juego no tenga problemas a la hora de jugar.

Asimismo destacar que la versión de API de Android que se utiliza es la 14 debido a que según datos de Android Studio es compatible con el 97,4% (Figura 3) de los dispositivos Android del mercado lo cual facilita que cualquier usuario pueda jugar con la aplicación. Además la aplicación no requiere de forma absoluta utilizar una versión mayor (la cual sí sería necesario en el caso se requiriera una funcionalidad no disponible en versiones inferiores)

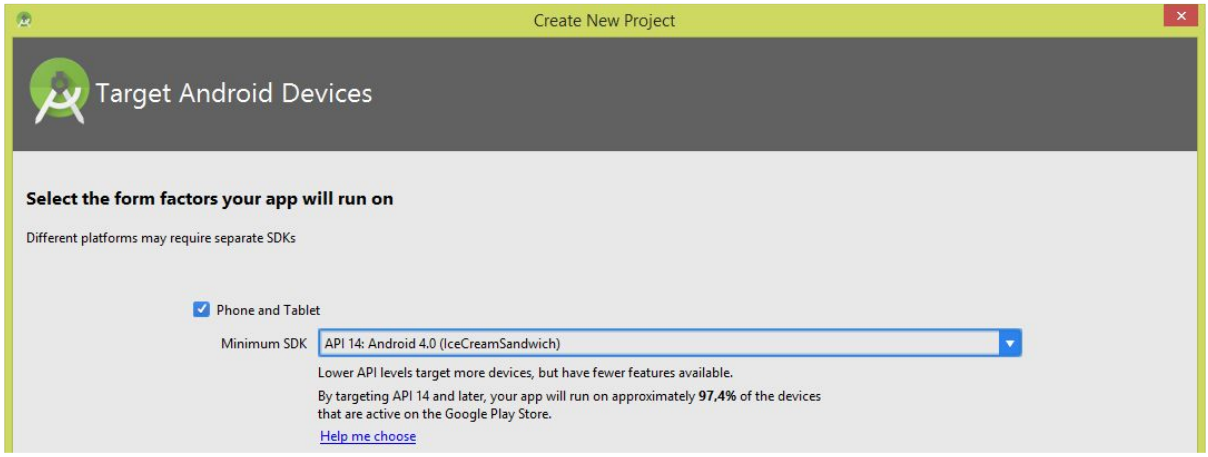


Figura 3: Datos de Android Studio que indican el tanto por ciento de compatibilidad de dispositivos en la tienda virtual de Google.

3.4.2. Limitaciones hardware

Dado que es requisito imprescindible que la aplicación se ejecute en un sistema Android, las limitaciones hardware serán las que impongan dichos dispositivos.

4. Análisis

En el ciclo del desarrollo del software, la fase de análisis es imprescindible con objetivo de reunir y estructurar los componentes y funcionalidad que presenta el sistema a desarrollar.

4.1. Diagrama de clases

En dicho diagrama se cuenta con los siguientes conceptos: (clases):

Juego: La clase principal que gestionará las acciones a realizar en cada turno. Contiene a los jugadores, el tablero y el juego. Ya que la aplicación es un juego por turnos, se cuenta con el atributo JugadorActivo que representa el jugador que tiene el turno en ese momento.

Tablero: Representa el tablero del juego y construye y contiene las casillas del juego.

Dado: Se encargará de proveer cuando sea necesario un resultado que será un número con rango del 1 al 6.

Turno: Se encarga de gestionar las diferentes fases por las cuales pasa el jugador que tiene el turno, por ejemplo, lanzar dado, mover ficha...

Casilla: Consta de las casillas del juego. Además ejecutarán una acción cuando una ficha de un jugador llegue a una dependiendo del tipo de casilla. Se dividen en tres tipos y realizarán una acción u otra al llegar una ficha dependiendo del tipo de casilla:

- Casilla final: Cuando una ficha llega y su jugador tiene todos los quesitos, gana la partida. Si se falta al menos uno, cede turno al siguiente jugador.

- Casilla base: Cuando una ficha llega, lanza una pregunta aleatoria de la categoría a la que corresponde. Si el jugador que controla la ficha la acierta, vuelve a tirar los dados. Si falla, cede turno al siguiente jugador. En caso que esta casilla sea casilla quesito, además otorgará un quesito de la categoría a la que pertenece al jugador si el jugador no había obtenido el quesito de la categoría anteriormente.

- Casilla relanzar dados: En esta casilla, cuando llega una ficha, el jugador vuelve a lanzar los dados.

Jugador: Representa los jugadores del sistema. Se distingue entre jugadores humanos y jugadores CPU. Además, contiene el atributo nombre, que es el apodo del jugador en el sistema.

Ficha: Representa la ficha de cada jugador que además almacenará las categorías de los quesitos obtenidos.

Pregunta Respuesta: Entidad que contiene una pregunta y cuatro posible respuestas con sólamente una como correcta.

Categoría: Representa las categorías del juego, también contará con un atributo del color que representa a la categoría. Las categorías serán Ciencias y Naturaleza, Deportes, Historia, Geografía, Deportes, Espectáculos y Arte y Literatura.

Importante resaltar que el concepto *quesito* no se encuentra presente. Cada quesito pertenece a una categoría determinada, por lo que la relación de una ficha tiene un quesito de una categoría se representa mediante la expresión de que la ficha tiene dicha categoría.

Respecto a las relaciones entre entidades en el diagrama, destacar que como restricción no mostrada, existe solamente una casilla final en el tablero y el tablero contiene seis casillas quesito, una por categoría.

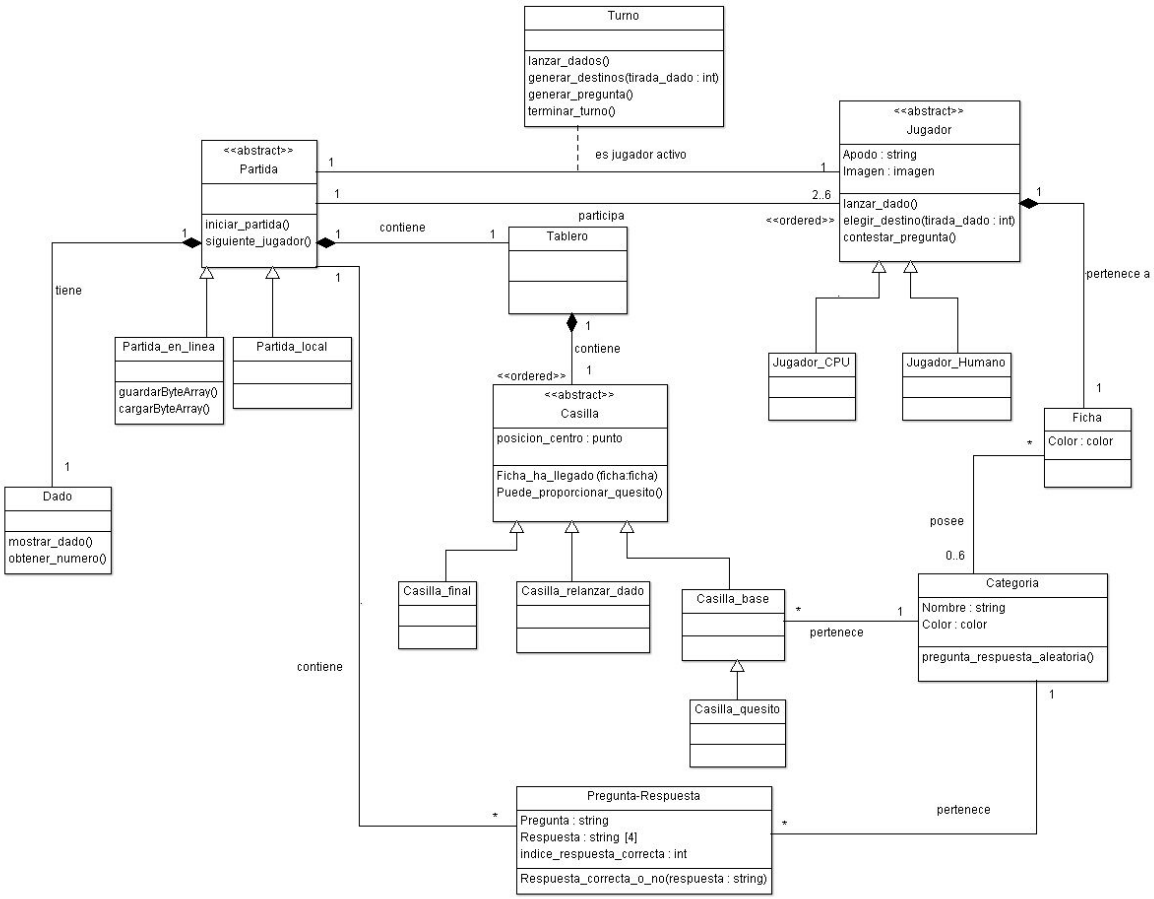


Figura 4. Diagrama de clases

5. Diseño

Respecto a la arquitectura de la aplicación, se empleó una arquitectura por capas distinguiendo la capa de presentación, la cual se encargará de mostrar datos e interactuar con el usuario, la capa de lógica de negocio, en la cual se llevará a cabo los procesos de los datos de la aplicación y por último, la capa de persistencia, que se emplea para interactuar con los datos contenidos en un almacén de datos.

5.1. Capa de presentación

En primer lugar, se mostrará el esquema de las interfaces gráficas que contiene la aplicación y su conexión entre ellas.

Resaltar que, al principio en la interfaz gráfica se deseaba que los quesitos estuviesen dentro de las fichas como ocurre con el juego de mesa. Sin embargo, al realizar pruebas, se comprobó que en dispositivos que no tuviesen grandes dimensiones, los quesitos en las fichas aparecían muy pequeños y podía forzar la vista, con lo que se decidió que los quesitos serían mostrados en una pantalla aparte.

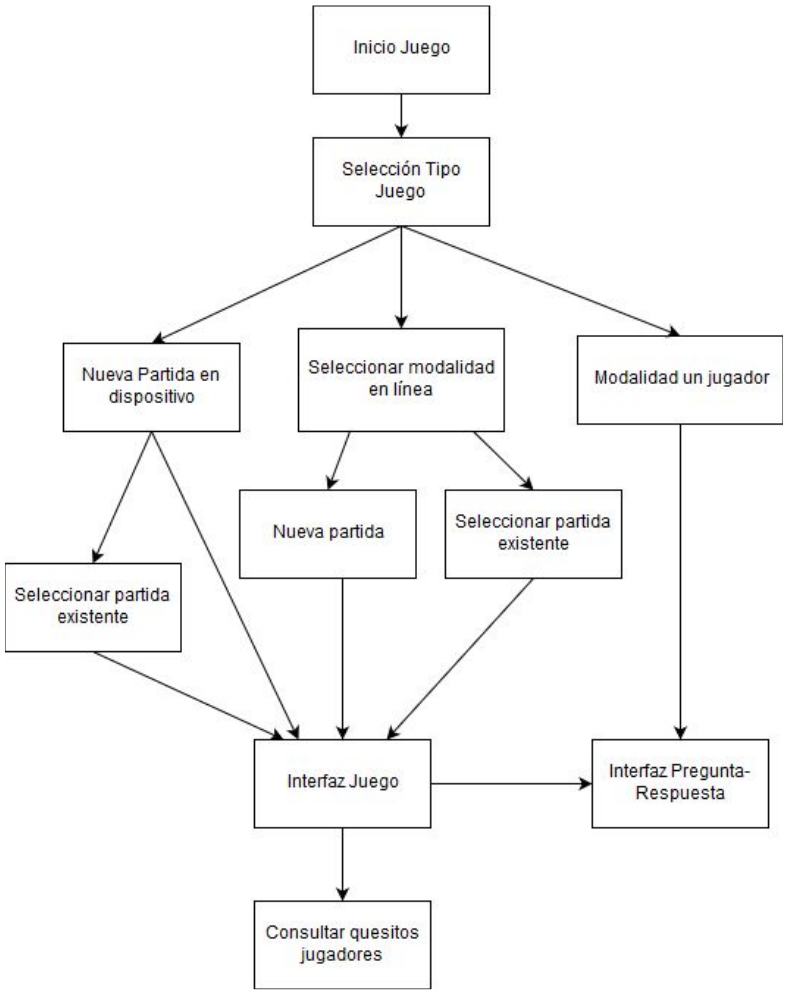
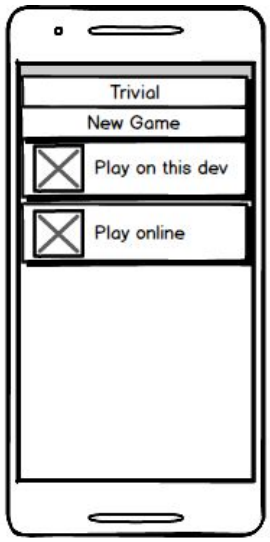
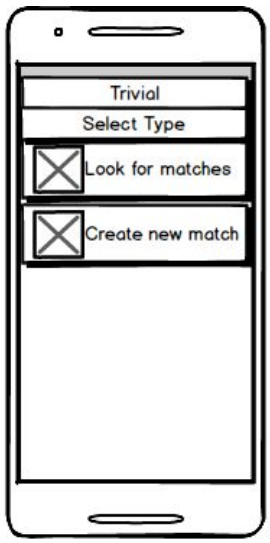


Figura 5: Esquema de las interfaces gráficas de la aplicación con sus conexiones.

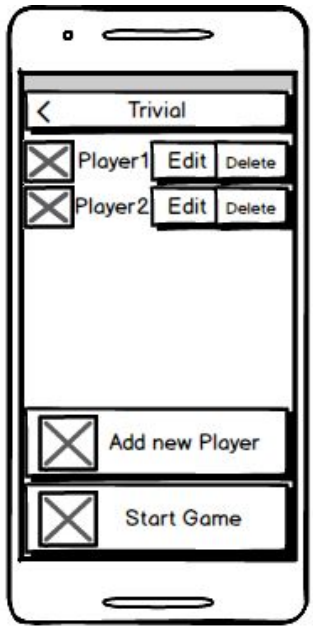
En primer lugar se diseñaron unos mockups preliminares sencillos que se fueron perfeccionando conforme se avanzaba con el proyecto. Los mockups se realizaron con la herramienta web gratuita en versión de prueba Balsamiq (www.balsamiq.com)



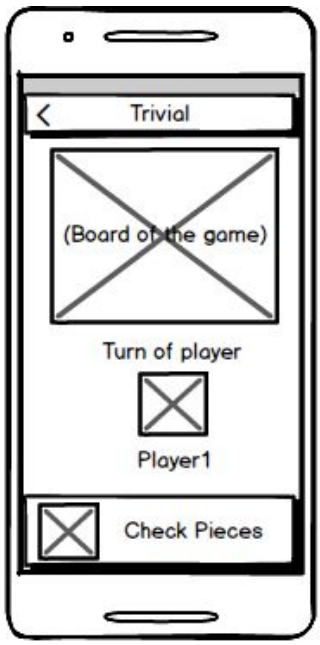
Mockup 1: Selección tipo de juego.



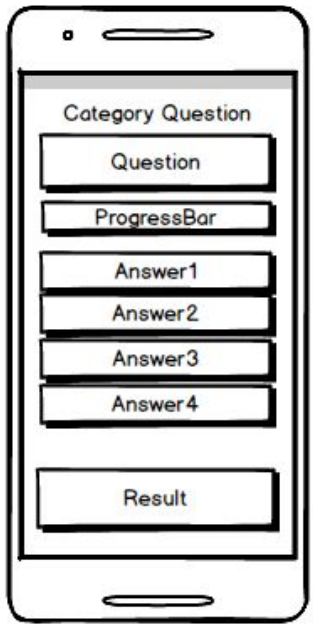
Mockup 2: Seleccionar tipo de juego en línea.



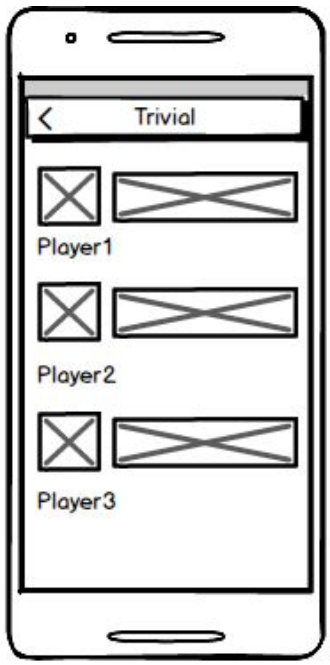
Mockup 3: Gestión de jugadores en partida en dispositivo.



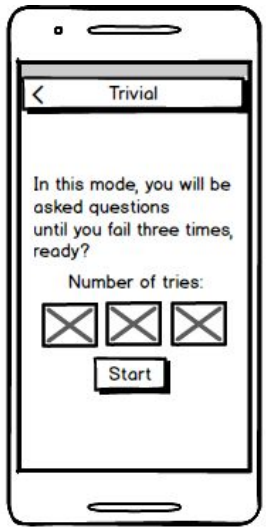
Mockup 4: Interfaz de la partida



Mockup 5: Interfaz gráfica de preguntas y respuestas.

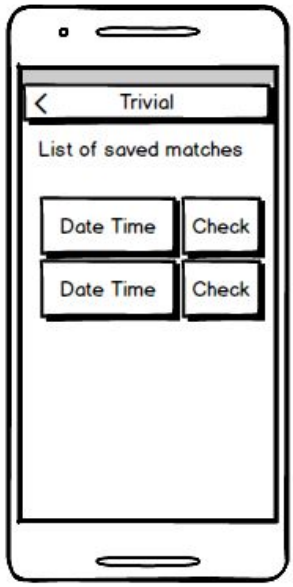


Mockup 6: Interfaz consulta quesitos de jugadores.



Mockup 7: Interfaz modo para un jugador

Respecto a las interfaces de buscar y añadir jugadores en línea y seleccionar partidas en línea, se emplearon las interfaces que ofrece Google Play para ello ya que se ajustaban a la funcionalidad buscada.



Mockup 8: Interfaz de gestión de partidas locales guardadas

5.2. Capa de lógica

5.2.1. Flujo del turno

Mediante el uso del diagrama de transición de estados mostrado en la Figura 5 se distinguen todas las fases del turno de un jugador. En todos los casos se distingue entre el

caso que esté jugando un jugador real y que esté jugando un jugador controlado por el sistema, a este jugador se le denominará CPU por ser el nombre estándar en los videojuegos para este tipo de jugador.

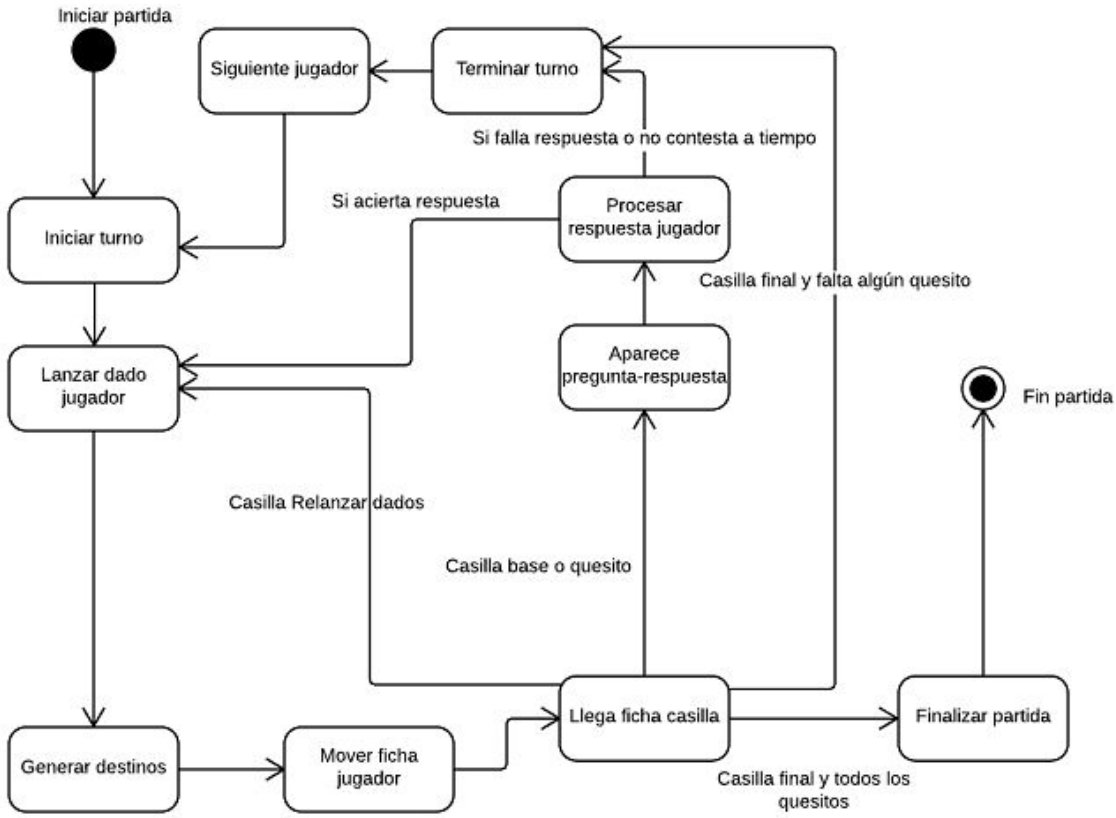


Figura 6: Diagrama de transición de las fases del juego

A continuación se describe la funcionalidad de cada una de las fases:

- **Iniciar juego**

A partir de la lista de jugadores, se ordenan aleatoriamente y se selecciona al primer jugador, además se construye el tablero y las casillas.

- **Lanzar dado jugador**

En esta fase se muestra un dado en el centro de la pantalla en cual se van sucediendo de forma aleatoria y rápida los números del 1 al 6. En el caso de un jugador humano tiene que pulsar en el dado y se detendrá. El último número mostrado por el dado será el valor obtenido en la tirada.
En el caso de un jugador CPU, el dado se detendrá transcurridos unos instantes. Una vez detenido el dado, desaparece de la pantalla y se comienza la siguiente fase (generar destinos) enviando el resultado numérico del dado.

- **Generar destinos**

A partir del resultado de la fase lanzar dado, se lanzará el algoritmo para seleccionar a cuales casillas adyacentes se puede acceder. Estas se marcarán de un color especial. En caso de la CPU calcula la casilla que más le conviene a la hora de recoger los quesitos restantes o en caso de tener todos, a la casilla central. Una vez seleccionada la casilla, se procede a la siguiente fase. En caso de jugar un jugador real, selecciona una casilla destino y a continuación se procede a la siguiente fase (Mover Ficha). En ambos casos se envía la casilla seleccionada.

- **Mover Ficha jugador**

Fase muy sencilla, se elimina el color especial de las casillas a las que se puede acceder y se mueve la ficha a la nueva casilla y se prosigue a la fase siguiente, Llega Ficha.

- **Llega Ficha casilla**

En esta fase se distingue el tipo de casilla en la cual la ficha se encuentra:

- Si es la casilla final comprueba si el jugador tiene todos los quesitos, si los tiene se avanza a la fase Finalizar Partida. Si no, se continúa a la fase Terminar Turno.

- Si la casilla es una casilla base o una casilla quesito se continúa en la fase Pregunta Respuesta.

- Si es una casilla relanzar dado, se vuelve a la fase Lanzar Dado.

- **Lanzar Pregunta Respuesta**

En esta fase se selecciona una pregunta al azar de las disponibles en el juego que pertenezcan a la categoría de la casilla. A continuación se muestra la pregunta junto a las cuatro posibles respuestas y se ofrece un tiempo limitado para contestar. Si el tiempo finaliza sin proveer una respuesta, la pregunta se contará como contestada erróneamente. El jugador real selecciona una respuesta de las disponibles, o deja que se termine el tiempo. En caso de la CPU, seleccionará una respuesta a base de un coeficiente de acierto (por defecto es 60%) que determinará si contestará correctamente o no y siempre responderá antes que acabe el tiempo. A continuación se pasa a la fase de Procesar Pregunta Respuesta.

- **Procesar Pregunta Respuesta**

En esta fase se procesa si la pregunta se ha contestado correctamente o no. Si se ha contestado correctamente y la casilla es quesito y el jugador no tiene el quesito de la categoría de la casilla, le es otorgado el quesito de la categoría. A continuación se pasa a la

fase de Lanzar Dado. En caso de haber contestado incorrectamente, se pasa a la fase de Terminar Turno.

- **Siguiente Turno**

Se cede el turno al siguiente jugador de la lista de jugadores y se pasa al estado Lanzar Dado del nuevo jugador.

- **Finalizar Partida**

Se notifica a los jugadores que el jugador que se encuentra realizando su turno ha ganado la partida y se notifica que la partida ha terminado.

Modo un jugador

Se mostrará a continuación el diagrama de transición de estados en el modo de un jugador. Este modo consiste en preguntas seguidas hasta que el jugador falla tres veces y termina. Cada vez que falla el jugador pierde una "vida". Se muestra el diagrama de transición de estados a continuación:

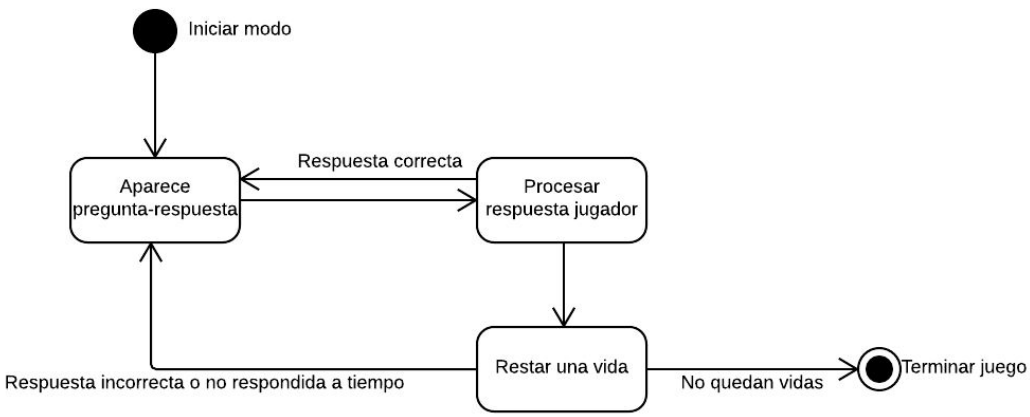


Figura 7: Diagrama de transición de estados en el modo un jugador.

5.3. Capa de persistencia

La base de datos que se ha empleado ha sido Sqlite ya que Android incluye soporte para esta base de datos.

Uno de los datos persistentes de interés son las preguntas y las respuestas de la aplicación. Con dicho objetivo se definió la base de datos que se presenta a continuación:

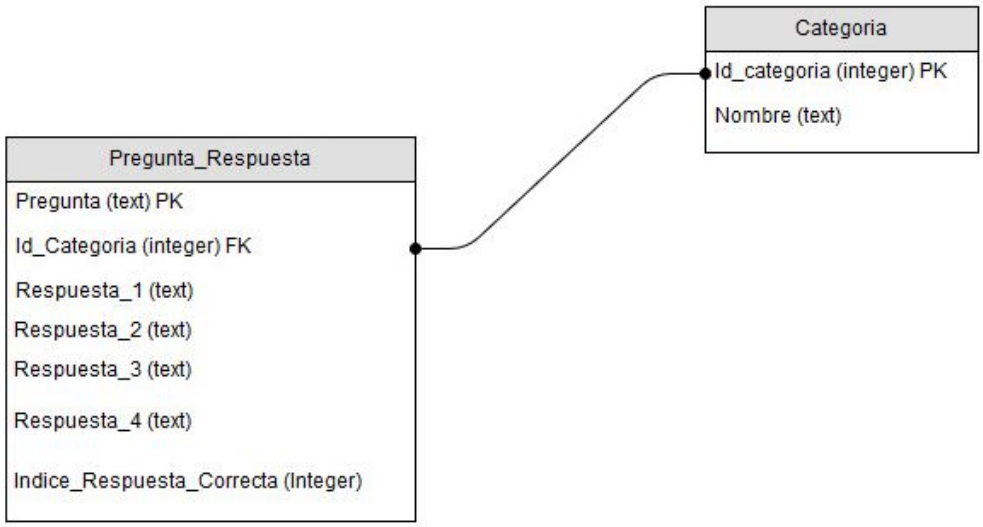


Figura 8 : Base de datos del conjunto de preguntas/respuesta

El procedimiento para mantener preguntas con sus respuestas es emplear un archivo Excel con esta información, el cual se exporta a CSV y se importa a la tabla pregunta respuesta base de datos. Esto permite una gran flexibilidad a la hora de gestionar los datos (es decir facilita añadir preguntas y respuestas, corregir preguntas/respuestas incorrectas o no bien formuladas, eliminar preguntas respuesta…).

Para guardar los datos de las partidas que se jueguen en el dispositivo, es decir, las que no se juega en línea, se utilizó el archivo sharedpreferences del dispositivo. La manera de leer y escribir en el dispositivo es muy sencilla. Emplea el sistema key-value.
Como valor de *key* se utiliza la fecha de guardado del juego y cómo *value* se utiliza un string con los datos de la partida (apodo de jugadores, posiciones de las fichas, quesitos de los jugadores, tipo de jugadores (controlados por el sistema o humanos), jugador activo del juego y fase actual del jugador activo) Existe un método que permite obtener todas las *key* guardadas.

Los datos de las partidas en línea se guardan en el servidor de Google Play Services. Se explicará con más detalle en el capítulo 6.6.1. como se realiza.

6. Implementación

En este apartado se mostrará cómo se ha construido el software de la aplicación, tanto en las herramientas utilizadas, como la estructura de la aplicación y los detalles que se consideran más importantes a la hora de llevar a cabo esta fase.

6.1. Lenguajes empleados

A continuación se presentarán todas las tecnologías utilizadas en este proyecto. Todas las que se han empleado son gratuitas, a excepción de Google Play Services, la cual hay que pagar una pequeña tasa de aproximadamente 23,3€.

Java

Java es el lenguaje de programación empleado para la implementación del proyecto. Java se trata de un lenguaje de programación multiplataforma, de propósito general, concurrente y orientado a objetos. Fue desarrollado por Sun Microsystems, la cual fue adquirida por Oracle Corporation.

XML

XML se define como las siglas de *eXtensible Markup Language*, se trata de un meta-lenguaje de marcas extensible. Fue desarrollado por el *World Wide Web Consortium* (W3C) y se emplea para almacenar datos de forma legible.

En el caso de Android, para el diseño de las interfaces gráficas, los mantiene separados de los ficheros Java y estas interfaces las almacena y gestiona en ficheros XML.

6.2. Tecnologías empleadas

Android

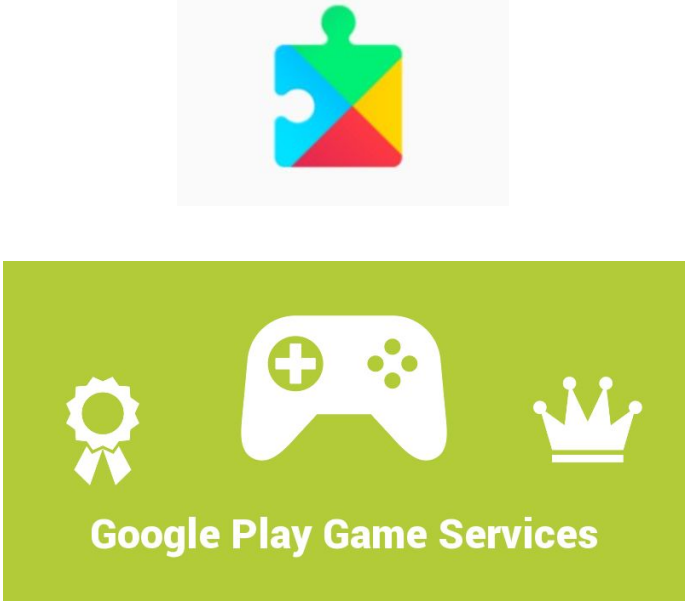
Android es un SO propiedad de Google el cual está presente en gran parte de dispositivos móviles (Acerca de 94% de dichos dispositivos).

Para este proyecto, se utilizó la versión 14, la cual según datos proporcionados por Android Studio es compatible con el 97,4% de dispositivos en Google Play Store con el objetivo de asegurar la compatibilidad con la mayoría de dispositivos.

Android Studio

Android Studio ha sido el IDE (Integrated Development Environment) escogido para realizar la implementación. Se trata de un IDE expresamente diseñado para desarrollar aplicaciones Android. Además ofrece un emulador de dispositivos Android, denominado AVD Emulator, para poder probar las aplicaciones sin necesitar un dispositivo físico. Sin embargo, el emulador no es compatible con las funciones de Google Play Game Services con lo que obliga a probarlo en un dispositivo, lo cual lo priva de funciones como la consola de ejecución para obtener información de cómo sucede.

6.3. Integración con Google Play Game Services



Con objetivo de permitir el juego multijugador, se decidió emplear Google Play Game Services ya que se encargaría de las tareas de conectar a los usuarios y gestionar la funcionalidad multijugador de forma sencilla que además permite la persistencia de datos de las partidas en sus servidores. El único requisito que se requiere a los usuarios para jugar en línea es una cuenta de Google Mail y su uso es completamente gratuito. Sin embargo, para emplear la funcionalidad de Google Play Game Services en una aplicación software es necesario contar con una cuenta de Google Developers, y crear la cuenta exige un pago de 25$, que es aproximadamente 23,3€.

6.4. Organización del proyecto

La estructura del proyecto se puede observar en la Figura 8. Android es poco flexible e impone una estructura de directorios concreta en los cuales cada elemento debe ir en un directorio determinado.

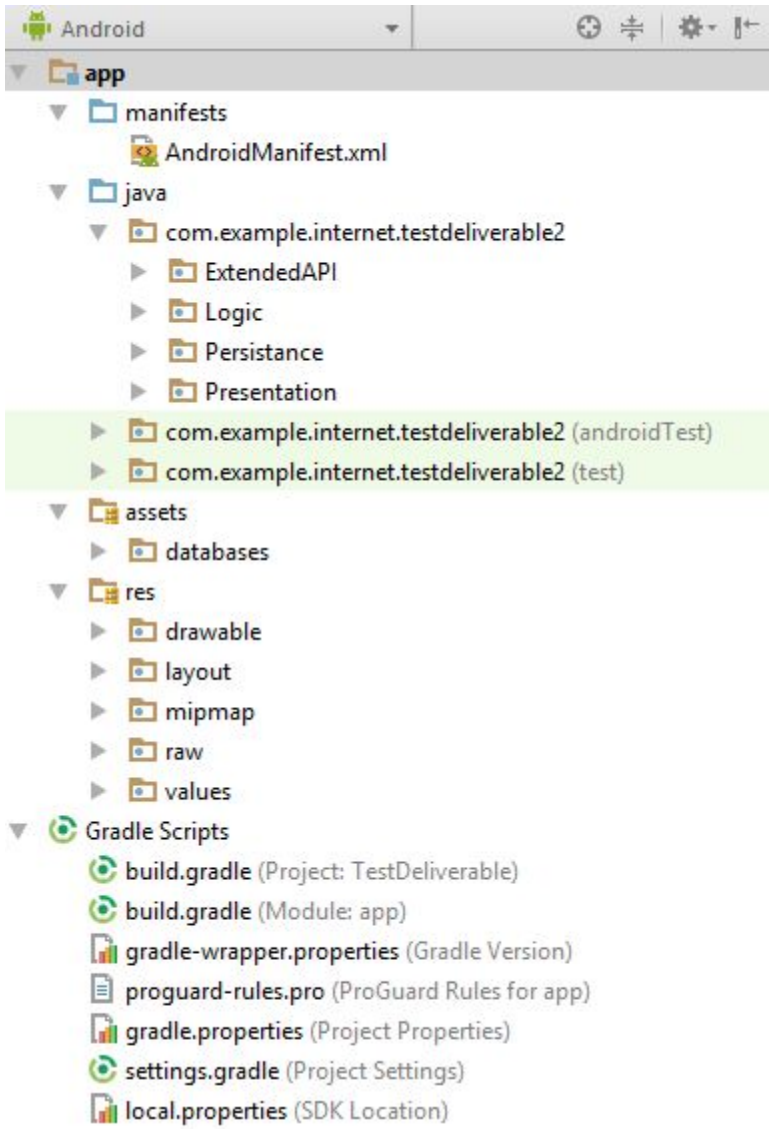


Figura 9: Estructura de ficheros en la aplicación.

- **manifest/ :** Contiene datos de la configuración del proyecto. Android se encarga del contenido de este directorio.

- **java/ :** Contiene los archivos java que han implementado el proyecto. Está dividido en los tres directorios correspondientes a las tres capas que componen el proyecto y un directorio auxiliar que contiene funciones de apoyo. El directorio de presentación contiene clases que heredan de *Activity* y cada clase carga el *layout* correspondiente de su interfaz gráfica contenida en *res/layouts*.

- **assets/ :** En este directorio, se almacena la base de datos del proyecto.

- **res/ :** Contiene los recursos de la aplicación, drawable contiene las imágenes empleadas en el proyecto, *layout*, como su nombre indica, contiene los l*ayouts* de

las interfaces gráficas, raw contiene los archivos de sonido que empleará la aplicación y values las constantes como el tamaño de letra, colores….

- **gradle scripts/ :** Se trata de archivos de configuración del proyecto, Android se encarga del contenido del proyecto.

6.5. Gestión de dimensiones de dispositivos

Con el objetivo de integrar la interfaz gráfica a distintos dispositivos, tanto dispositivos móviles como Tablets, se siguieron las siguientes metodologías:

- El uso de los atributos *weight y weight_sum,* que permiten que las *view* contenidas en un *viewgroup* ocupen un ratio de tamaño determinado por el espacio disponible. Por ejemplo, si un *viewgroup* tiene un weight sum de 3 y tiene disponible 600 dp de ancho para tres botones y cada uno tiene un *weight* de 1, cada uno ocupará 200 dp de ancho. Si en otro dispositivo se cuenta con 900 dp de ancho, cada botón ocupará 300 dp de ancho.

- Empleando un *layout* distinto por cada conjunto de tamaños de pantalla. En Android se consigue creando en el directorio *res* un directorio con nombre *layout*, junto al identificador que utiliza Android para el tamaño de pantalla, por ejemplo, *layout-large.* Android ya se encarga de emplear uno u otro *layout* dependiendo de las dimensiones del dispositivo.

- Los tamaños de letra se procede de la siguiente manera: se crea constantes con un valor y mismo nombre para distintas dimensiones y según el tamaño del dispositivo obtiene el valor correspondiente a sus dimensiones para la constante requerida. Estas constantes se almacenan en *res/values*.

6.6. Detalles de la implementación

De la implementación se profundizará en los siguientes apartados:

6.6.1. Implementación del multijugador mediante Google Play Services

Para integrar Google Play Game Services al proyecto en primer lugar hay que realizar unas modificaciones en AndroidManifest.xml y en el build.gradle, además de incorporar las API adecuadas [2].

A la hora de implementar el multijugador mediante Google Play Game Services (a partir de ahora, GPGS) se tomó como modelo el código proporcionado en [3], destacando los detalles explicados a continuación.

GPGS permite que las partidas sean o en tiempo real o por turnos. Por turnos para Google Play Services implica que los jugadores no tienen porque jugar en el mismo momento en el cual el jugador previo a su turno finalice el turno.

Se podría haber optado a jugar en tiempo real para añadir dinamismo a las partidas y que terminen momentos después de iniciarlas. Sin embargo, al ser un juego con larga duración (las partidas pueden durar como mucho unas pocas horas) se optó a que los jugadores puedan continuar su turno en cualquier momento, especialmente en pequeños instantes cortos. Por tanto se escogió la modalidad por turnos.

GPGS sólo permite que el intercambio de información del juego sea mediante un array de tipo byte. Este array almacenará toda la información del juego exceptuando los apodos de los jugadores ya que esa información se puede obtener directamente de GPGS .

La manera de obtener este byte array es utilizando el método getDataByte de la clase TrivialOnlineGame. La manera de actualizar el juego de acuerdo con los datos obtenidos con el byte array sería con el método setDataByte.

Se añadió la función *updateData*, la cual se va ejecutando periódicamente cuando el usuario se encuentra en línea mediante *Handler* y su método *postDelayed* y funciona de la siguiente manera: En el caso que el usuario haya cargado una partida, se comprueba si los datos de la partida (el array byte explicado en la sección anterior) son distintos a los últimos datos que ha recibido. Si son distintos, llama a la función *updateMatch* para que actualice los datos de la partida y permite al jugador jugar su turno si otro jugador ha terminado su turno y le toca jugar. Además *updateMatch* ya contiene el código que se encargará de actualizar la interfaz gráfica con los últimos datos recibidos. Se muestra el código a continuación:

```
public void updateData() {

    if (mMatch != null) {

        String string = mMatch.getMatchId();

        Games.TurnBasedMultiplayer.loadMatch(mGoogleApiClient, string).setResultCallback(
                new ResultCallback<TurnBasedMultiplayer.LoadMatchResult>() {
                    @Override
                    public void onResult(TurnBasedMultiplayer.LoadMatchResult loadMatchResult) {
                        mMatch = loadMatchResult.getMatch();
                        if (!Arrays.equals(data, mMatch.getData())) {
                            updateMatch(mMatch);
                        }
                    }
                }
        );
    }

}
```

Figura 10: Código para la actualización periódica

Además, el switch que contiene el método updateMatch visto justo antes se le añadió una variable y un método auxiliar, isDoingTurn y setGamePlayUI respectivamente. Este método, en caso que se trate del jugador actual, permitirá jugar su turno y además, si carga una partida en la cual ya estaba realizando su turno seguirá desde el último punto salvado, por ejemplo, si lanza el dado y deja de jugar, seguirá por la fase que elige hacia donde mueve la ficha con el resultado del dado y continuará con la partida. Además en cualquier caso se encargará de mostrar en pantalla el estado actual de la partida (fichas en la casilla correspondiente, quesitos obtenidos…)
La variable isDoingTurn se utiliza para almacenar el estado de turno actual, del jugador o de un oponente.

```
switch (turnStatus) {

    case TurnBasedMatch.MATCH_TURN_STATUS_MY_TURN:
        data = mMatch.getData();
        isDoingTurn = true;
        setGameplayUI(true);
        return;
    case MATCH_TURN_STATUS_THEIR_TURN:
        isDoingTurn = false;
        data = mMatch.getData();
        setGameplayUI(false);
        break;
```

Figura 10: Implementación de la gestión del turno de los jugadores.

Finalmente, para guardar los datos en el servidor o terminar el turno se llama al método takeTurn. Este método incluido en GPGS se utiliza para dos cosas. Una es almacenar y recuperar datos en un array de tipo byte como se ha visto anteriormente y otra es pasar el turno a otro jugador. Si se llama al método pasándole el mismo jugador que está jugando, solamente almacena datos. Por tanto, en caso que el jugador continúe su turno, por ejemplo, al lanzar el dado, guardar la tirada del dado en caso que el jugador decida seguir jugando en otro momento. En esa situación se pasa al método takeTurn como jugador al mismo jugador. Si el jugador termina el turno, se pasa al método el string del siguiente jugador para que GPGS le ceda el turno. Los otros dos datos que pide takeTurn son la instancia de GoogleApiClient y el string de la partida (match) correspondiente. Se siguió el código indicado al principio del capítulo y se pasan mGoogleApiClient y mMatch.getMatch().

```
Games.TurnBasedMultiplayer.takeTurn(mGoogleApiClient, mMatch.getMatchId(),
        data, getCurrentParticipant()).setResultCallback(
```

Figura 11: Método takeTurn para guardar datos en los servidores de Google Play Game Services y para ceder el turno al jugador correspondiente si es el caso.

6.6.2. Implementación de las listas en pantalla

Para la implementación de las listas en pantalla y poder interactuar con ellas, se utilizó Listview y BaseAdapters para ello [3]

Se creó tres clases que hereden de BaseAdapter para implementar los listados, los datos que contienen y por tanto, muestran se indica a continuación.

- BaseAdapterMatch, se utiliza para gestionar las partidas locales del dispositivo y muestra la fecha del guardado de la partida y contiene además de la fecha de guardado, los datos que contiene la partida.

- BaseAdapterPlayers, se utiliza para gestionar los jugadores que se pueden añadir a una partida local, es decir, en el mismo dispositivo, contiene y muestra en pantalla el apodo de cada jugador y el tipo de jugador (humano o controlado por el sistema)

- BaseAdapterStatusPlayers, se utiliza para visualizar los quesitos que tiene cada jugador, esta clase muestra los apodos de los jugadores, el tipo de jugador y los quesitos de los jugadores.

6.6.3. Calculando puntos del tablero

Con el objetivo de poder ubicar las fichas en el tablero, se siguió el siguiente procedimiento:

- En primer lugar, se obtiene la imagen contenida en el *layout* XML, al conocer el *context* de la pantalla en la cual se ejecuta habría que hacer, suponiendo que la imagen tenga como *id* en el *layout* '*image*':

  (ImageView) image = context.findViewById(R.id.image)

- Seguidamente, se calcula las dimensiones del propio tablero, no son fijas ya que las dimensiones varían según el tamaño de la pantalla. Se calculan mediante el código a continuación:

```
getImage().measure(0,0);

array[0] = getImage().getMeasuredWidth();

array[1] = getImage().getMeasuredHeight();
```

- Después, se calcula el centro de la imagen que contiene del tablero. Como se encuentra en la parte superior de la pantalla, el desplazamiento de su componente 'y' será cero. Como el tablero se encuentra centrado en pantalla, se puede calcular su desplazamiento con la fórmula: desplazamiento = (x'-x) / 2. Observar Figura 11.

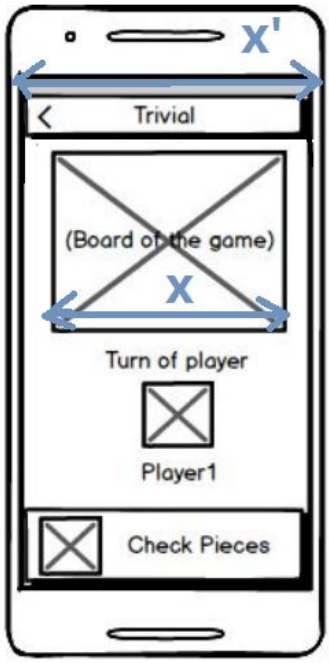


Figura 12. Valores de x' y x.

Para calcular el centro del tablero, habrá que sumar los desplazamientos al centro, que se calcula dividiendo entre dos las dimensiones del tablero.

centro x = desplazamiento x + dimensión x/2

centro y = desplazamiento y + dimensión y/2

- A continuación, se calculan los puntos que corresponden a los centros de las casillas del tablero. Para ello se crean circunferencias cuyos puntos se encuentran en los centros de las casillas. Para calcular un punto concreto de la circunferencia que sea punto central de una casilla, se hará mediante el ángulo que forma respecto al centro del tablero. [4].

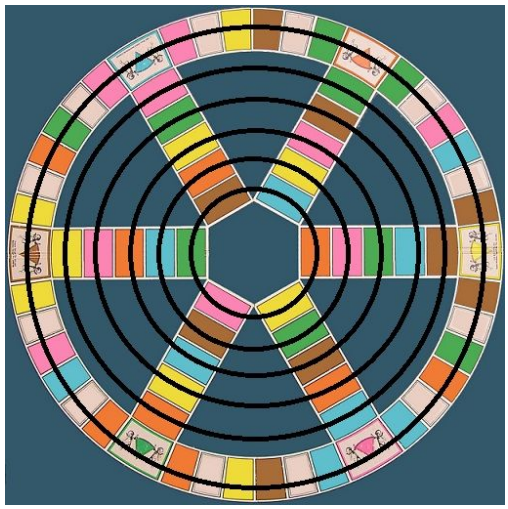


Figura 12: Circunferencias que pasan por los centros de las casillas.

Figura 13: Se muestran algunos puntos que entran en el centro de las casillas. Estos puntos se calculan mediante la circunferencia y el ángulo que forma con el tablero.

- Finalmente se añade cada punto calculado a la variable centro de cada casilla.

Al tener cada casilla su punto central, se puede añadir círculos que representan las fichas de cada jugador mediante el procedimiento que se explicará en el apartado siguiente, apartado 6.6.4.

Asimismo, también se utilizan los puntos explicados justo anteriormente para crear círculos que indican al jugador las casillas destino que accesibles con la tirada del dado. El jugador puede seleccionar una casilla destino pulsando el círculo que rodea esa casilla destino (Figura 14)

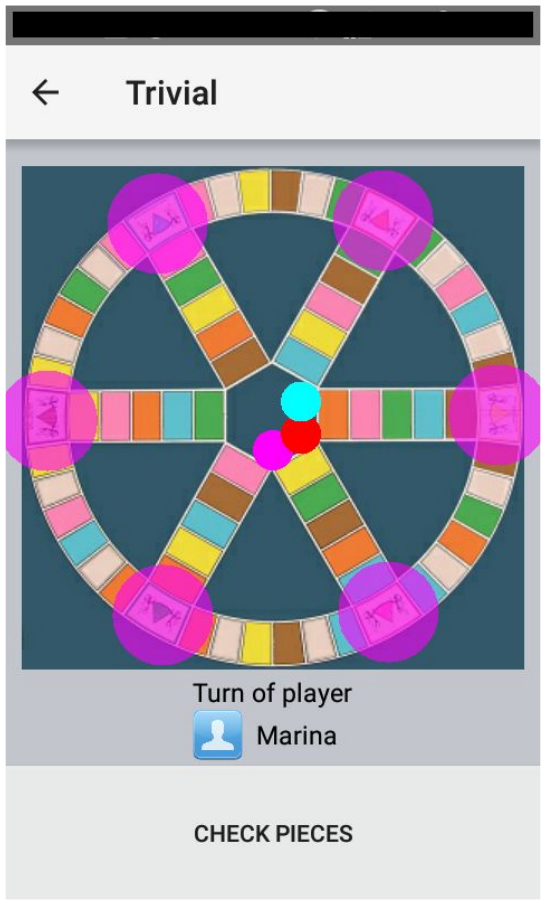


Figura 14: Casillas alcanzables con el lanzamiento del dado.

Para saber el punto de la pantalla que ha pulsado el jugador, se obtiene el *layout* de la *activity* (el cual se obtiene mediante R.id.*nombre-del-layout*) y se utiliza el método setOnTouchListener, el cual incluye un *MotionEvent* cuyos métodos getX y getY permitirán obtener el punto, para conocer si ese punto pertenece a un círculo que envuelve a una casilla se utiliza la fórmula matemática de cómo obtener si un punto está dentro de un círculo [5]. El método getClickedDestiny obtiene la casilla a la cual pertenece el círculo que el jugador ha pulsado en su interior. Retornará null si se ha pulsado en la pantalla pero en ningún círculo. En el momento que el jugador pulse en un círculo, el juego obtendrá la casilla y proseguirá con el flujo del juego.

```
    getLayoutScreen().setOnTouchListener(new View.OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {

            FloatPoint floatPoint = new FloatPoint(event.getX(),
                    event.getY());

            Square square = getClickedDestiny(floatPoint);

            if(square == null) return false;

            else {

                trivialGame.onClickedSquare(square);

            }

            return false;
        }
    });
}
```

Figura 15:Código que permite seleccionar una casilla.

6.6.4. Añadiendo fichas y visualizadores de casillas destino a la interfaz gráfica

Se tomó la decisión de representar las fichas del juego como círculos. Para poder mostrarlas en pantalla, se extiende la clase *View* (por ejemplo denominándose CircleDraw) y se sobreescribe su método *draw* para que utilice el método drawCircle con los datos necesarios como se puede ver a continuación:

```
public void draw(Canvas canvas){

    super.draw(canvas);

    Paint paint = new Paint();

    paint.setColor(circle.getColor());

    canvas.drawCircle(circle.getCenterX(),circle.getCenterY(),circle.getRadius(),paint);


}
```

Figura 16: Creando *View* personalizados que dibujan círculos para mostrar las fichas.

Este método se llamará automáticamente una vez se crea la *View*, no hay que llamarlo. Si se desea que una *view* tenga transparencia, una vez creada hay que utilizar el método *setAlpha*. Como cada casilla tiene su centro, las fichas serán dibujadas en dicho centro o cercano a él en caso que la casilla tenga más de una ficha. Estas *view* se añadirán al *layout*

de la pantalla, que se puede obtener mediante R.id.*nombre-del-layout* si se le asigna en el XML un id a la propiedad *android:id*. Una vez obtenido, las *view* se añadirán mediante el método *addView*, y *removeView* las eliminará.

6.6.5. Construcción de los caminos de las casillas destino

Para la construcción de estos caminos, se utiliza una clase auxiliar denominada Camino (*Path*) la cual se encarga de obtener tanto las casillas destino disponibles a partir de una casilla, tanto el número de pasos que son necesarios para acceder de una casilla a otra (es decir, el número de casillas adyacentes que hay que recorrer para acceder de una casilla a otra)

Como se indicó en apartados anteriores, cada casilla está conectada con las casillas adyacentes, por tanto, de una casilla determinada se puede obtener cuales son las casillas adyacentes.

El algoritmo para obtener estos resultados contiene una la lista de exploración, la cual contendrá la casilla que se está explorando, el número de pasos y la casilla de la cual proviene. Por ejemplo, (B,1,A) se refiere a que la casilla actual es B, el número de pasos es uno y proviene de la casilla A.

El algoritmo desarrollado se explicará mediante un ejemplo, en este caso se supondrá que la casilla actual es A y el número de pasos (la tirada del dado) es dos:

- En primer lugar, se rellena la información de la casilla con el número de pasos posibles y la casilla anterior. Como se inicia el algoritmo, no hay casilla anterior, al iniciarse el algoritmo, los pasos son cero y no hay casilla previa, se rellenará a null. Por tanto la lista se quedará de la siguiente manera: (A,0,null)

- A continuación, se obtienen las casillas adyacentes a A, que se supondrá que son B,C y D, se añadirá uno al contador de pasos y se añadirá la casilla anterior, que es A. Como ninguna casilla adyacente es igual a la última (ninguna de las adyacentes es igual a null), no se descarta. Por tanto la lista contendrá (B,1,A),(C,1,A) y (D,1,A)

- Se volverá a repetir el algoritmo, suponiendo que las casillas adyacentes a B son A y E, las casillas adyacentes a C son A y F, y las casillas adyacentes a E son A y G. En el caso de (B,1,A) serían (E,2,B) y (A,2,B), sin embargo, como la casilla anterior y la casilla adyacente son A, se descarta quedando únicamente (E,2,B). La lista finalmente queda de la siguiente manera (E,2,B),(F,2,B) y (G,2,B). Como la tirada es 2 y la lista contiene elementos con indicador de pasos igual a dos, el algoritmo se termina y las casillas destino son (E,F,G)

Este algoritmo también se puede emplear para obtener el coste de ir de una casilla a otra. La finalización del algoritmo ocurriría cuando la casilla destino se encuentra en la lista de casillas y el coste sería el número de pasos. Por ejemplo, el coste de ir a la casilla A a D

sería uno. La manera en la cual se ejecuta garantiza que se obtendrá el coste mínimo ya que primero obtiene los destinos con coste uno, seguidamente los destinos con coste dos…

Mediante el algoritmo de costes se puede calcular que a partir de una casilla inicial y un conjunto de casillas, cuál está más cerca de la inicial. Este es el algoritmo que utiliza la CPU para, a partir de un resultado del lanzamiento del dado, obtener que casilla destino está más cerca de una casilla quesito que no haya obtenido el quesito y por tanto elegir dicha casilla destino.

7. Pruebas

Las pruebas de la aplicación se han ido desarrollando al mismo tiempo que la aplicación en sí, cuando se comprobaba que una funcionalidad no era la requerida o la esperada, se ha acudido a corregir los fallos en ese momento.

Para las pruebas de interfaz gráfica se ha recurrido a amigos y familiares con rango de edades entre los 20 y los 50 años con conocimientos de usuario en aplicaciones móviles y de videojuegos para tener una segunda opinión y conocer qué sería mejorable.

8. Conclusiones y proyectos futuros

Con este TFG, se ha conseguido el objetivo propuesto, llevar el original juego Trivial adaptado completamente sin modificaciones con el objetivo que sea un Trivial de mesa portable y que se pueda jugar con cualquier persona sin necesidad que esté presente físicamente.

Como proyecto personal, destacar que esta ha sido la primera aplicación Android que hemos hecho, con lo que ha costado un tiempo adaptarse al entorno, tanto la manera que tiene Android de funcionar como al entorno Android Studio, sin embargo lo que más ha costado ha sido la integración con Google Play Game Services, una vez realizada, se tuvieron que realizar varias pruebas para comprender su funcionamiento.

No obstante, se ha cumplido el propósito personal de conseguir aprender a desarrollar aplicaciones en Android que además sean multijugador e incluyan acceso a una base de datos.

Han quedado opciones y detalles del juego que hubieran sido interesantes en caso de ampliar el contenido del proyecto pero no se han realizado por cubrir por falta de tiempo. Se enumeran a continuación:

- Estadísticas de los jugadores, por cada categoría, obtener el porcentaje de acierto de las preguntas realizadas para hacer que los jugadores compitan y jueguen más, aumentando así su conocimiento cultural.

- Ranking en línea de jugadores según puntuación que podría ser el ratio de preguntas respondidas totales entre preguntas respondidas o ranking de puntuaciones por categorías de preguntas.

- Integración con redes sociales, como Facebook y Twitter con el objetivo de dar el juego a conocer a amigos y publicar resultados de partidas y récords.

8. Bibliografía

En la siguiente sección se mostrará tanto la bibliografía referenciada durante el texto, como la bibliografía que se ha consultado para la realización de este proyecto:

Referencias:

[1] Wikipedia.org, Android, Wikipedia.
https://es.wikipedia.org/wiki/Android [Consulta: Febrero 2017]

[2] Google, Setting Up Google Play Games Services, Google Developers.
https://developers.google.com/games/services/console/enabling [Consulta: Febrero 2017]

[3] Github.com, Android-basic-samples, Github.
https://github.com/playgameservices/android-basic-samples/tree/master/BasicSamples/SkeletonTbmp/src/main/java/com/google/example/tbmpskeleton [Consulta: Febrero 2017]

[4] Stack Exchange Inc, Calculating the position of points in a circle, StackOverflow.
http://stackoverflow.com/questions/5300938/calculating-the-position-of-points-in-a-circle
[Consulta: Febrero 2017]

[5] Stack Exchange Inc, Equation for testing if a point is inside a circle, StackOverflow.
http://stackoverflow.com/questions/481144/equation-for-testing-if-a-point-is-inside-a-circle
[Consulta: Febrero 2017]

Bibliografía:

Google.com, Accessing the Play Games Services APIs in Your Android Game, Google Developers.
https://developers.google.com/games/services/android/init [Consulta: Febrero 2017]

Google, Adding Turn-based Multiplayer Support to Your Android Game, Google Developers.
https://developers.google.com/games/services/android/turnbasedMultiplayer [Consulta: Febrero 2017]

Google.com, Atriviate, Google Play.
https://play.google.com/store/apps/details?id=aul.irm.triviados&hl=es
[Consulta: Febrero 2017]

Google, Creating custom views, Google Developers.
https://developer.android.com/training/custom-views/index.html [Consulta: Febrero 2017]

Google.com, Cuestionados, Google Play.

https://play.google.com/store/apps/details?id=com.diablins.android.leagueofquiz&hl=es [Consulta: Febrero 2017]

Google, Compatibilidad con diferentes tipos de pantalla, Google Developers. https://developer.android.com/training/multiscreen/screensizes.html [Consulta: Febrero 2017]

Google.com, Saving Key-Value Sets, Android Developers, Google. https://developer.android.com/training/basics/data-storage/shared-preferences.html [Consulta: Febrero 2017]

Google, Supporting Different Screens, Google Developers. https://developer.android.com/training/basics/supporting-devices/screens.html [Consulta: Febrero 2017]

Stack Exchange Inc, What does android:layout_weight mean?, StackOverflow. http://stackoverflow.com/questions/3995825/what-does-androidlayout-weight-mean [Consulta: Febrero 2017]

Revelo, James, Tutorial de *layouts* en Android, Hermosa Programación. http://www.hermosaprogramacion.com/2015/08/tutorial-layouts-en-android/ [Consulta: Febrero 2017]

Wikipedia.org, Trivial Pursuit, Wikipedia. https://es.wikipedia.org/wiki/Trivial_Pursuit [Consulta: Febrero 2017]

Wordpress.com, Ejemplo de ListView en Android, AmatellanesWordpress.com. https://amatellanes.wordpress.com/2013/04/14/ejemplo-de-listview-en-android/ [Consulta: Febrero 2017]

Anexo A. Manual de usuario

Al iniciar la aplicación, se mostrarán las tres opciones de juego, multijugador en el mismo dispositivo, multijugador en línea y modo individual.

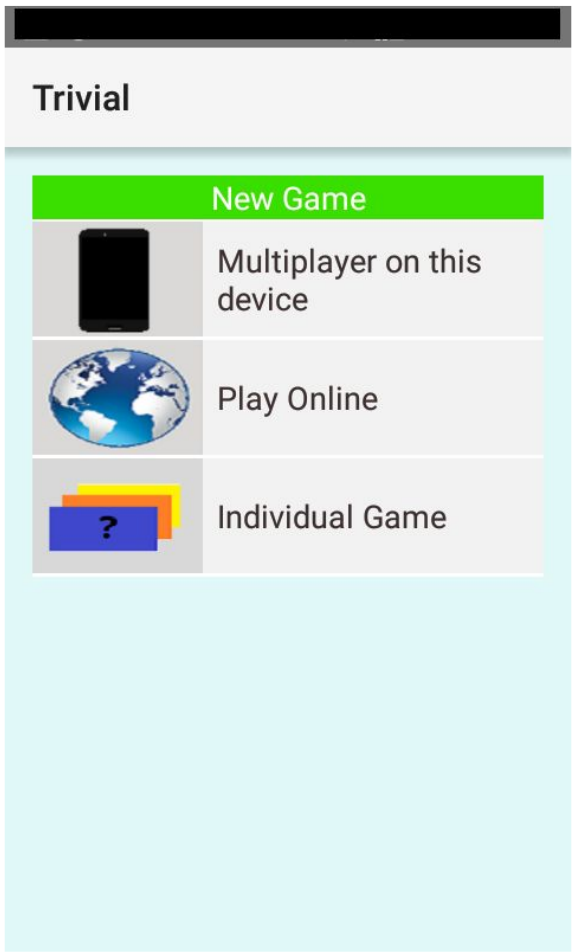


Figura 1: Inicio del juego

En el juego en línea, se presentarán las opciones de buscar partidas o crear una nueva partida. Al buscar una partida ya existente (Figura 4), seleccionarla bastará para cargarla. Mediante el menú de la derecha se podrá consultar los jugadores de la partida.

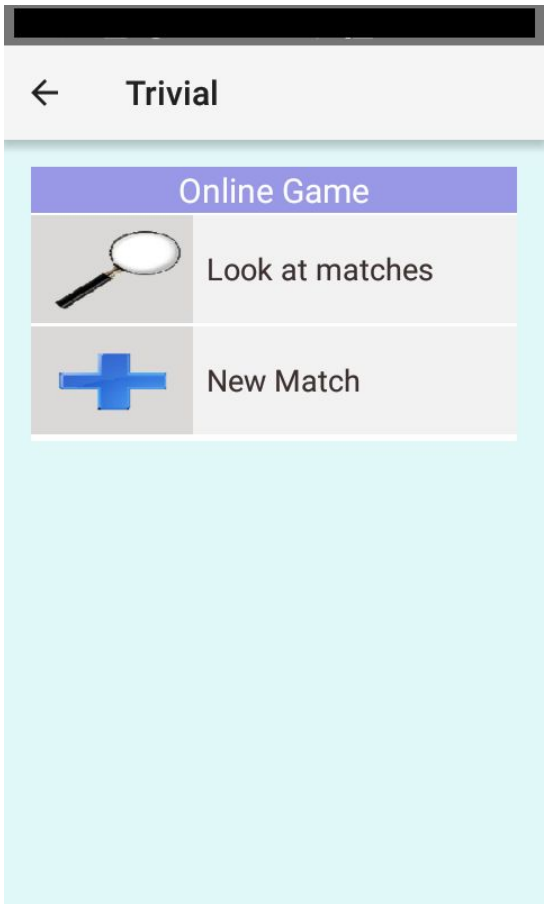


Figura 2: Opciones en línea.

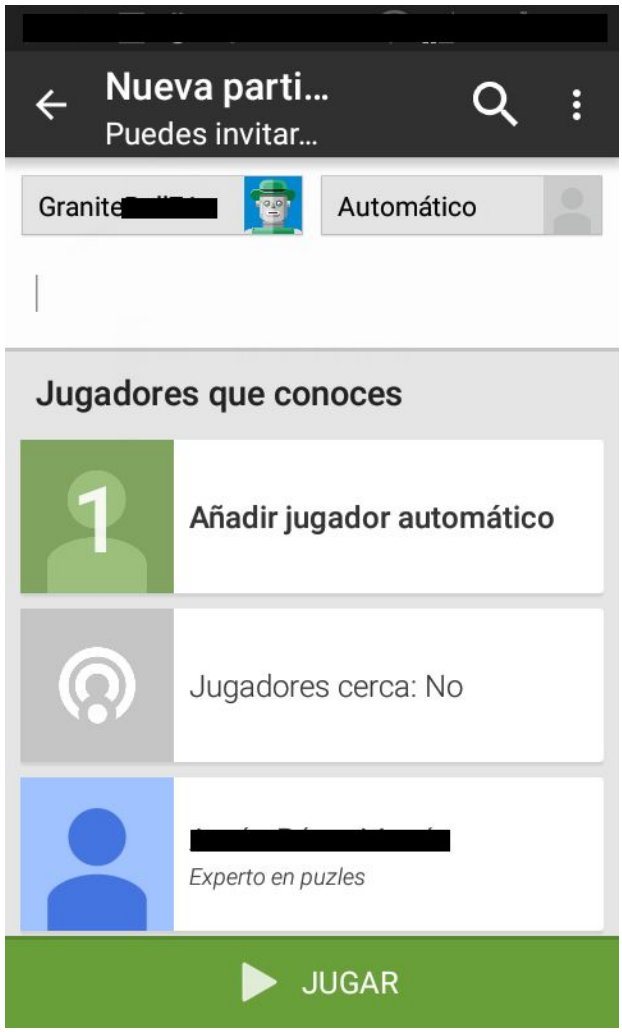


Figura 3: Añadiendo jugadores al juego en línea.

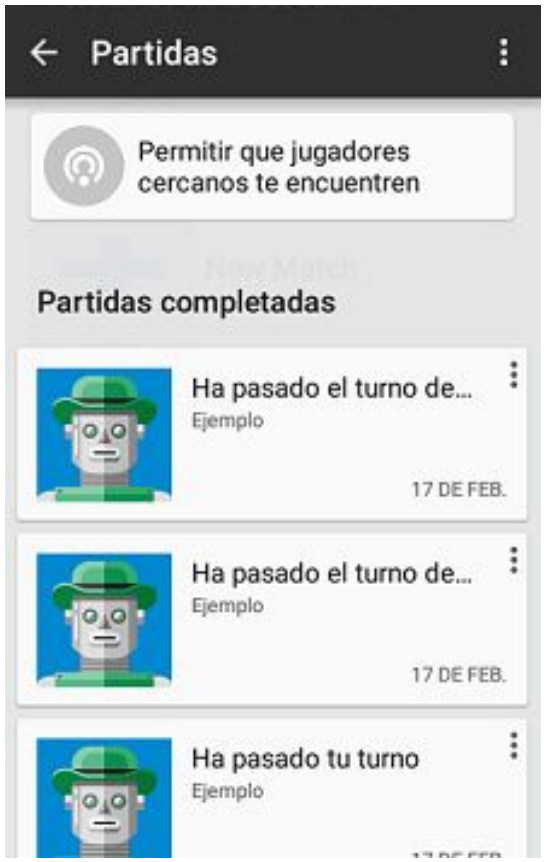


Figura 4: Partidas en línea del jugador.

En el caso de seleccionar una partida en el dispositivo, se ofrece la opción de añadir jugadores y cuando estén añadidos empezar partida, también se ofrece la posibilidad de cargar una partida existente.

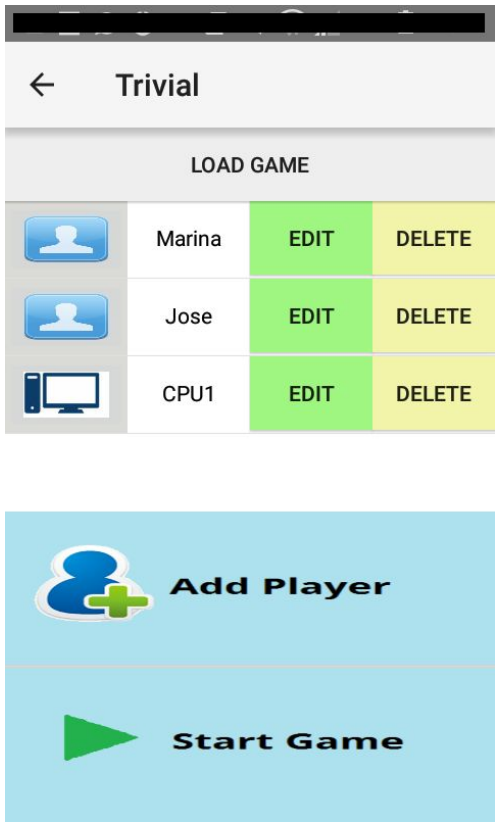


Figura 6: Gestión de los jugadores de una partida en el dispositivo.

En la pantalla de la Figura 6, se puede pulsar en “Load Game” para obtener las partidas en el dispositivo guardadas, que dirigirá a Figura 6. Con “Check” se pueden comprobar los datos de la partida y pulsando en ello se carga la partida y se prosigue con la partida.

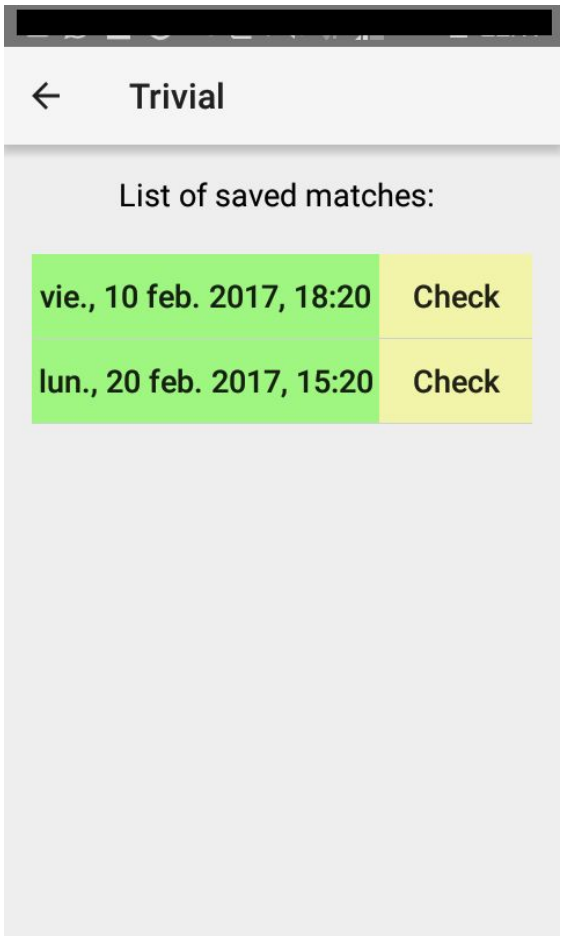


Figura 6: Partidas en el mismo dispositivo almacenadas.

Una vez en la partida, ya sea en el mismo dispositivo o en línea, al iniciar el turno del jugador, aparecerá un dado, en el cual habrá que hacer click (Figura 7). En el caso de un jugador CPU, se detendrá en unos pocos segundos.

Figura 7: Dado de la partida.

A continuación, habrá que seleccionar una casilla. En caso de la CPU, la seleccionará automáticamente. Las casillas a las que sea posible acceder aparecerán en un color distinto, que será de una tonalidad de la ficha del jugador.

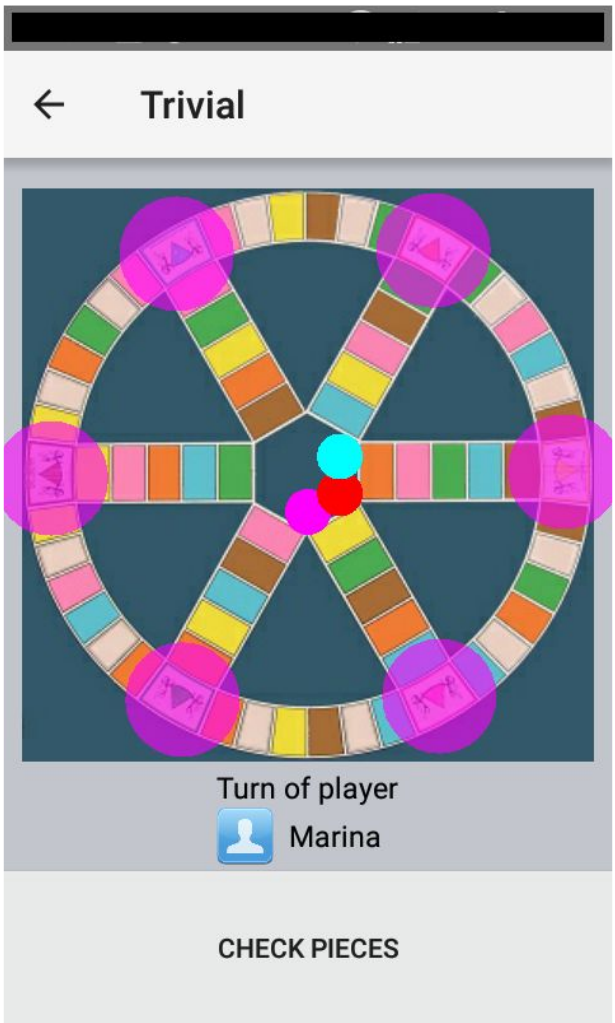


Figura 8: Casillas a las que es posible acceder.

Dependiendo del tipo de casilla, habrá una acción u otra. Puede ser, o ceder el turno al siguiente jugador al no tener todos los quesitos, o ganar la partida, o volver a lanzar los dados o que aparezca una pregunta con sus respuestas (Figuras 9 y 10).

Las casillas de volver a lanzar el dado son rectangulares y grises sin dibujos. Las casillas con preguntas y respuestas son de colores, cada color representando una categoría del conocimiento. Las casillas que proporcionan quesitos son también de colores y éstos representan una categoría, pero aproximadamente el doble de anchas. La casilla central, al llegar sin todos los quesitos, hará que el jugador ceda el turno. Si tiene todos los quesitos al llegar, el jugador gana la partida.

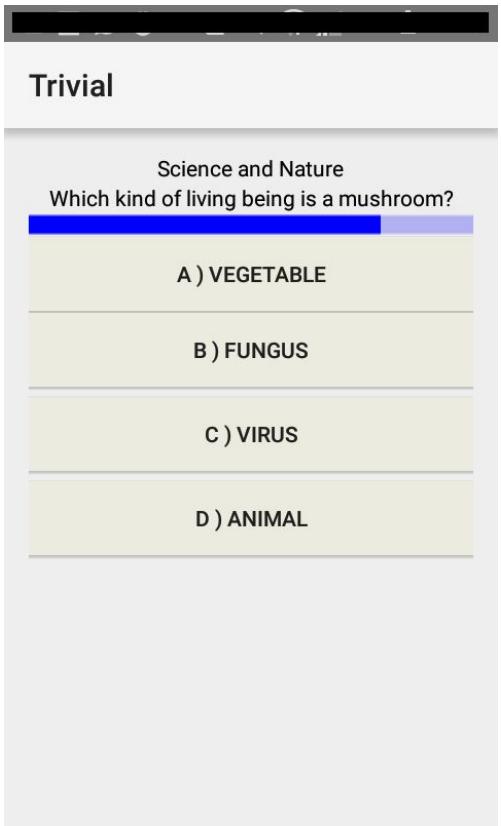


Figura 9: Preguntas con sus correspondientes respuestas.

Si es un jugador humano, se debe seleccionar una respuesta, si es un jugador CPU, la respuesta será seleccionada casi automáticamente. En la pantalla aparecerá un mensaje de si la respuesta es correcta o no. En el caso que el jugador no conteste antes que se acabe el tiempo, se interpretará como que contestado incorrectamente.

En el caso que un jugador obtenga un quesito al acertar y ser la casilla apropiada, será notificado en la pantalla.

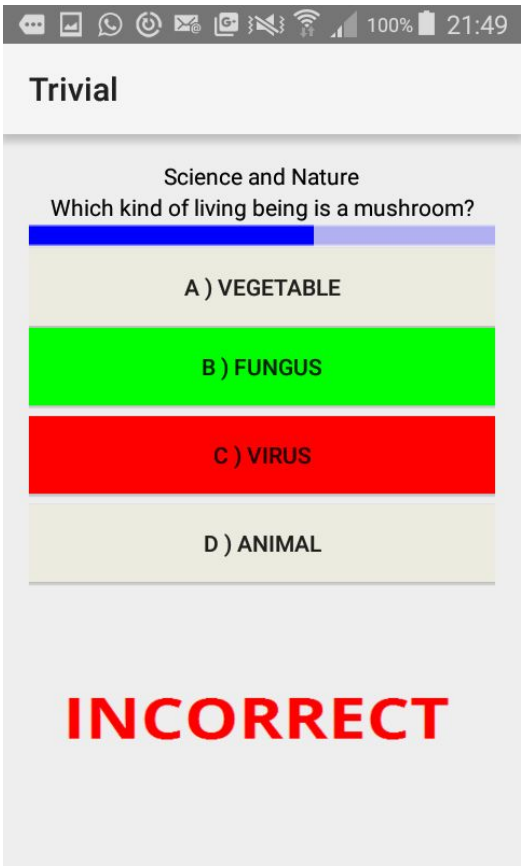


Figura 10: Muestra que la pregunta ha sido respondida de manera incorrecta.

Para acceder a los quesitos de los jugadores, habrá que pulsar en la opción CheckPieces, que mostrará a los jugadores de la partida con los quesitos obtenidos (Figura 11). El juego continuará hasta que un jugador llegue a la casilla central y tenga todos los quesitos en cuyo caso el juego anunciará al ganador. Si se desea abandonar la partida momentáneamente, se debe pulsar en la parte superior de la pantalla donde está indicado "Trivial" en cualquier momento, excepto durante el momento de responder a una pregunta. Si es una partida local, ofrecerá la opción de guardarla. Si es una partida en línea, se guarda automáticamente.

Figura 11: Quesitos de los jugadores.

Al iniciar el modo un jugador, aparecerá la pantalla indicando de qué trata el juego (Figura 12). Al pulsar ‘Jugar’ empezará el modo y el juego lanzará preguntas. Al fallar una se perderá una vida. Cuando no se tengan vidas, se acabará la partida.

Figura 12: Inicio modo individual.

Anexo B. Monetarización

En el caso que se lance la aplicación al mercado, se han contemplado varias maneras de obtener beneficio económico de ello. Asimismo durante todo el documento se ha enfatizado que tanto la plataforma Android como los dispositivos móviles son muy utilizados por el público en general y que se suele emplear para jugar sencillos como el presentado en este documento, por tanto, lanzar el juego al mercado sería una opción muy viable. Se muestra a continuación los modelos de negocio que se han pensado que se podrían llevar a cabo:

- Añadir publicidad a la aplicación, se puede destinar un espacio en algunas pantallas de la interfaz gráfica con ese fin. Si se trata de una publicidad no invasiva, no tendría porqué suponer un gran inconveniente a los usuarios.

- Cobrar por descargar o utilizar la aplicación, sin embargo aunque se trate de un precio pequeño, ya contaría con el descontento de parte de los usuarios además que existen juegos parecidos gratuitos. Quizá en vez de cobrar una pequeña cantidad sería mejor añadir publicidad para convertir la aplicación en más accesible y el dinero que se perdería cobrando, se ganaría en publicidad y popularidad. Otra variante sería cobrar transcurrido un tiempo o exigir un pago para poder seguir jugando tras un tiempo. Se ha considerado que presentaría inconvenientes parecidos a cobrar desde un primer momento, además el factor de cobrar después causaría mala imagen a los usuarios.

- Gratuito pero cobrar por “extras” en el juego. Hay tipos de juegos que son gratuitos pero se puede pagar para “acelerar” partes del juego. Por ejemplo en un juego de rol, se puede pagar para subir niveles automáticamente sin tener que luchar contra enemigos. En principio no se contempla qué opciones de juego se podría añadir para que funcionase este modelo. También el hecho de que este juego todos los jugadores parten desde el mismo punto, este modelo de pagar por mejoras, no tendría mucho sentido. Sin embargo lo que sí se podría hacer, es ofrecer la posibilidad a los usuarios de comprar sets de preguntas y respuestas temáticos o para un público determinado, por ejemplo sets para niños, mayores de cuarenta años, sets de cultura geek, como de señor de los anillos y la guerra de las galaxias… A la hora de jugar en línea, se podría ofrecer un modo para buscar jugadores que también tengan esos sets comprados y jugar con ellos.

Por tanto, en el caso que se lance el juego y se desee rentabilizar, se optaría por publicidad no invasiva y además la opción de comprar sets temáticos.