Especificación de la práctica

En esta práctica desarrollaremos el **Juego del ahorcado** utilizando una API pública para la lógica del juego, Firebase para los servicios de base de datos, tabla de puntuaciones, autenticación, publicidad y analíticas. Por eso utilizaremos el patrón de arquitectura MVVM utilizando Android Studio.

Prototipo de referencia:
https://www.figma.com/file/p1TlKHPzTneXehldluFzhq/Juego-del-ahorcado?node-id=3%3A942

Servicios

Firebase Authentication

https://firebase.google.com/docs/auth/android/start?hl=es
Utilizaremos autenticación anónima para los usuarios no alquilados. Los usuarios podrán ir al menú de configuración, **crear un usuario** con correo y contraseña y después conectarse a ellos. Si un usuario ya se ha **autenticado**, la siguiente vez que abra la app se conectará con las credenciales y no de forma anónima.

Firebase Realtime Database

https://firebase.google.com/docs/database/android/start?hl=es
Utilizaremos la base de datos en tiempo real para la tabla de puntuaciones. Cada vez que finalicemos una partida, si la puntuación es mayor que la que teníamos almacenada, actualizaremos la puntuación.

Las puntuaciones serán consultadas desde el **menú de puntuaciones**.

Firestore

https://firebase.google.com/docs/firestore/quickstart?hl=es
Utilizaremos esta **base de datos NoSQL** para almacenar la configuración del usuario y el nombre de usuario. La configuración la podrá cambiar desde el **menú de configuración** y el nombre desde la home pulsando sobre la sección de profile.

Firebase Analytics

https://firebase.google.com/docs/analytics/get-started?hl=es&platform=android#kotlin+ktx
Lanzaremos los siguientes eventos de analíticas:

level_start

Lo lanzaremos cada vez que **comience una partida**, sea la primera o porque ha terminado una palabra y va por la siguiente.

Parámetros:
level: *int* -> Número de palabras acertadas

new_chance

Cuando el usuario pierde, se da la **oportunidad de ver un anuncio** para tener un intento más, o no verlo, una vez seleccionada la opción se lanza este evento.

Parámetros:
view_ad: *bool* -> true o false según la elección del usuario.

show_ad

Cada vez que **mostramos un anuncio**.

Google Admob

https://developers.google.com/admob/android/quick-start
Lo utilizaremos para mostrar anuncios de recompensa cuando el usuario pierda, así le ofreceremos ver un anuncio para tener un intento nuevo.

Notificaciones

Si el usuario permanece inactivo más de 3 días, se lanzará una notificación de promoción para invitarle a jugar. Debe tenerse en cuenta que el usuario puede desactivarlas desde la configuración.

Hangman API

https://hangman-api.herokuapp.com/api
Es la API que utilizaremos para la lógica del juego. Las peticiones a la API se realizan utilizando la librería Android Retrofit.

Pantallas

La aplicación consta de tres pantallas: Init, Menu y Home. La pantalla de carga de splash la utilizaremos sólo entre la primera escena y el menú.

Init

Ésta será la primera pantalla y es donde identificamos al usuario y cargaremos los datos necesarios. Una vez todo esté cargado pasaremos a la escena de Menu.

Si el usuario ya está logueado utilizaremos sus credenciales, de lo contrario haremos un login anónimo.

Tras el proceso del login, se obtienen los datos del usuario. Si es un usuario nuevo, solicitamos un nombre y guardamos sus datos.

Durante este proceso, mostramos una pantalla de splash con el título del juego y una animación de carga.

Menu

El menú tiene tres secciones: Home, Score y Configuration.

Home

Se muestran los datos del perfil. Existe un botón que permite editar estos datos. Este nombre debe almacenarse en Firestore.Tendremos un botón para ir al juego.

Score

Aquí veremos todas las puntuaciones almacenadas en Firebase Realtime Database. Dado que puede haber muchas puntuaciones podemos hacer scroll sobre las mismas.

La tabla de puntuaciones está ordenada de mayor a menor puntuación y se mostrará: Posición, nombre del usuario, puntuación y tiempo de partida.

Configuration

El usuario puede crear una cuenta con correo y contraseña, o hacer login si ya tiene una. Al hacer login debemos almacenar los datos en PlayerPrefs encriptados (puede utilizar cualquier librería o algoritmo externo para encriptar los datos). Estos datos los utilizaremos al arrancar la app para hacer login sin que el usuario deba hacerlo manualmente.

Si el usuario ya tenía una cuenta y se autentica en otro dispositivo (o en éste), la información del profile deberá actualizarse. El usuario también podrá activar/desactivar las notificaciones y audio. Esta configuración se guardará en Firestore.

Game

El apio que utilizaremos para la lógica es la siguiente:
https://hangman-api.herokuapp.com/api

Al **arrancar el juego** solicitaremos una palabra nueva en la API. Deberemos mostrar el estado actual de la palabra y los otros intentos.

Cada vez que el usuario **marca una letra**, se pregunta a la API si la palabra contiene la letra y actualizamos el estado de la palabra con el resultado. Cuando se obtiene la respuesta, también debe desactivarse la letra e indicar si ha sido correcta o no. En caso de que no sea correcta, restamos un intento y actualizamos el gráfico del ahorcado.

Si el usuario se queda sin intentos, mostraremos un *dialog* para ver un anuncio y recuperar un intento, sólo lo mostraremos una vez.

Si el usuario se queda **sin intentos** definitivamente, mostraremos un *dialog* de derrota con la puntuación y el tiempo utilizado. Tanto si gana como si pierde actualizaremos la tabla de puntuaciones, sólo en caso de que la puntuación actual sea superior. En este *dialog* veremos un botón para reintentar que reiniciará el juego y otro para volver al menú.

Una sugerencia por la fórmula para la puntuación:

[larga_palabra] * 10 - [consonantes_usadas] - [vocales_usadas] * 5

Si el usuario **acierta** la palabra, mostraremos un *dialog* de victoria con el resultado y un botón para continuar o ir al menú . Si pulsa continuar pediremos otra palabra y volveremos a jugar pero acumulando los puntos y con los restantes intentos que quedaban.

El juego debe contener un botón para **pausar** el juego (detendremos el tiempo) y mostrar un *dialog* con el que resumir, reiniciar o volver al menú.

El juego tendrá **música** de fondo (que no moleste), y **sonidos** de acierto/error al seleccionar letras. Los audios estarán sujetos a la configuración del usuario que podrá activar o desactivarlos.

Formato de la entrega

Se entregará un **zip** con el proyecto de Android Studio (sin .gradle ni .idea) y un .txt con el enlace al repositorio de **git**. El repositorio debe contener el código a entregar a la rama main con una etiqueta y una release incluyendo el **código y el apk**, esto por cada una de las entregas. El repositorio debe estar compartido con el correo: pau.garcia@enti.cat

Evaluación

3 entregas + diseño

El código debe ser creado por los componentes del grupo en su totalidad, esto implica que no se podrá compartir código con los compañeros de clase o cualquier fuente externa. Si se detecta una copia, esto implicará un 0 en la evaluación de la asignatura y los alumnos implicados no tendrán derecho a recuperarla.

El código entregado debe compilar sin ningún error, de lo contrario será un 0.