Preentrega del Hangman

A continuació hi ha una llista detallada dels punts que es recomana tenir fets a la primera entrega. No cal que tots els punts estiguin estrictament implementats, tot i que és molt recomanable fer-ho. Per tenir la màxima nota, caldrà haver implementat aproximadament un **90% d'aquests punts**.

Per aquesta raó, cal que la majoria de pantalles de l'app estiguin implementades, tot i que les dades que es mostrin no siguin les correctes o no hi hagi totes les funcionalitats programades. Per exemple, la pantalla de puntuacions es pot crear amb un RecyclerView, tal com hem vist a classe. En canvi, com que encara no hem vist com accedir a una Realtime Database, les dades que es mostren poden estar hardcodejades per aquesta entrega.

Recordeu que l'app ha de tenir implementades les següents **pantalles**:

- SplashScreen
- Home
- Scores
- Configuration
- Game

També es recomana que l'app utilitzi els **dissenys definitius** que heu dissenyat en les classes amb la Ruth. A continuació es detallen les funcionalitats que es poden incorporar en aquesta entrega.

- **Inici de sessió** amb FirebaseAuth
    - Login automàtic en cas d'haver iniciat sessió prèviament
    - Login anònim
- Comunicació amb **l'API** del Hangman
- **Mecànica** del joc
    - Validar lletres
    - Dibuixar el penjat
    - Comptar el temps
    - Botó pausa
- **RecyclerView** amb les puntuacions (no cal que les puntuacions provinguin de la base de dades externa encara)
- Configuracions guardades amb les **SharedPrefs** i a **Firestore**

En aquesta entrega es valorarà la interfície de l'app, però sobretot la correctesa del codi utilitzat. Per això, és molt important fer servir tots els coneixements de Kotlin que s'han donat a teoria. Cal que la informació estigui estructurada en una **jerarquia de classes** i cal fer ús de funcions lambda sempre que ens permetin simplificar el codi.

A més, el codi ha d'estar ben estructurat, ordenat i comentat. La nomenclatura ha de seguir sempre els estàndards definits, tant pels fitxers, com per les classes, funcions i variables. Per veure en més detall els punts que es tindran en compte podeu consultar el document annex.

Per **l'entrega final** quedarà implementar:

- Les puntuacions amb la **RealtimeDatabase**
- Estadístiques d'ús de l'app amb **Firebase Analytics**
- Anuncis amb **Google Admob**
- Aplicar **Model View View model**
- **Notificacions**
- Afegir **música** i efectes de so
- Poder controlar les configuracions del joc (audio, notis, etc.)

Annex: Punts que es valoraran

Imprescindible per procedir a l'avaluació	
El projecte es pot **obrir i executar** amb la versió més nova d'Android Studio (2021.3.1)	
El **minSdkVersion** és: - Com a mínim API 22 (Android Lollipop 5.0) - Com a màxim API 28 (Android Pie)	Recomanat: API 24, 25 o 26
Codi	
El **codi és clar, net** i està ordenat	- No hi ha trossos de codi vell comentat - És seqüencial - No hi ha coses "brutes"
Comentaris al codi	- Comentaris en aquells trossos de codi que costa d'entendre què fan a simple vista. - Funcions importants - No abusar dels comentaris
Correctesa **organització** dels fitxers	- Els fitxers estan organitzats en carpetes i subcarpetes intuïtives - Cada classe està en un fitxer diferent
Correctesa en el **nom** dels fitxers	El nom dels fitxers és intuïtiu i segueix les pautes de nomenclatura
No hi ha errors ni **suggeriments d'Android Studio**	Qualsevol warning d'AndroidStudio pot ser penalitzat
Flux de la informació	Les dades que calen, es passen a través dels intents, dels fragments, etc
Coherència	Els botons fan el que diuen que fan, la transició entre pantalles és coherent
Robustesa	No peta mai
Rendiment	L'aplicació s'executa sense lag. Per exemple: No hi ha execucions pesades en l'onCreate, onResme, onPause, etc
No hi ha **copy-pastes**	No hi ha dos trossos de codi igual en dues parts del codi diferents

Les constants no estan **harcodejades**	Les constants numèriques i de text estan dins d'un fitxer *xml* o en un *companion object*. Tenir en compte els IDs que s'utilitzen per cridar funcions del SO
Els fitxers tenen **màxim 200 línies**	Recomanat són 150
Kotlin	
Jerarquia de classes	- Informació ben estructurada - Ús de **data class** quan cal
Es fa una bona gestió de la **nullability**	- No s'utilitza !! - Les variables només són ? quan cal
Es fa un ús correcte de **var i val**	S'utilitza sempre val, excepte quan és necessari un var
Els **noms** de les classes, funcions i variables segueixen la **nomenclatura** oficial	- camelCase - No guions baixos
Els **noms** de les classes, funcions i variables són **clars,** intuïtius i autoexplicatius	
Les variables estan correctament **tipades**	
Es fa un bon ús dels **modificadors de visibilitat**	S'utilitza sempre el més restrictiu (public, internal, protected, private)
Es fa un ús correcte dels **companion,** dels **objects** i de les **funcions lambda**	Es fan servir per emmagatzemar constants o funcions estàtiques i singletons
Layout & UX	
Coherència de l'**estil**	L'estil de tota l'app és coherent (mateixos colors, tipografia, formes)
Els widgets són **user-friendly** (botons, inputs, etc)	Es respecta el Material Design: https://m2.material.io/components
Es fa ús del **Material Design**	S'utilitzen widgets de Material.io
Feedback cap a l'usuari	S'informa l'usuari quan: - Hi ha un error - L'app està fent un càlcul llarg (o petició a internet) - Una tasca s'ha completat amb èxit
S'utilitza **view binding**	Per tots els Fragments i Activities

Android	
Es respecta el **cicle de vida** de les Activities i Fragments	Les operacions es realitzen al lloc que toca
Els Strings estan a la carpeta de **strings.xml**	