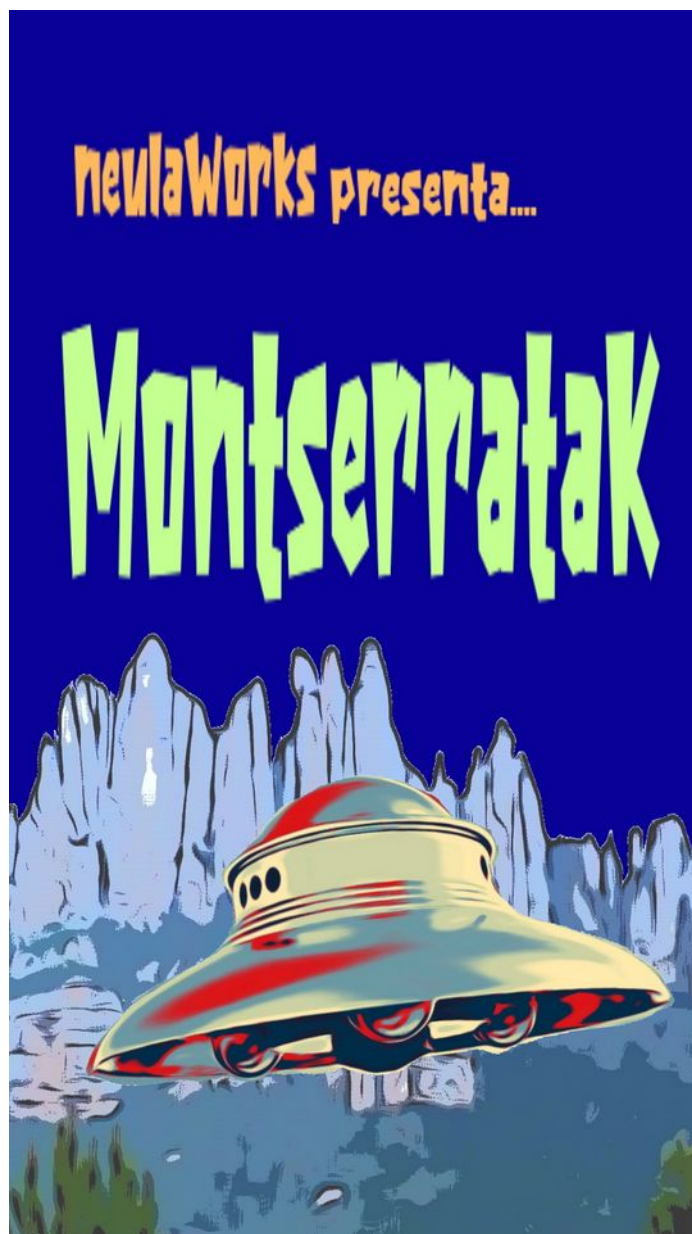


# Tutorial

## Montserratak



*Sergi Gotarra Juliol-Agost 2022*

<b>Introducció</b>	<b>3</b>
<b>Enunciat del projecte</b>	<b>4</b>
<b>Fase 1: Configuració de Firebase i pantalla splash</b>	<b>5</b>
<b>Fase 2: Registre i Login de jugadors</b>	<b>25</b>
<b>Fase 3: Identificador de Jugador</b>	<b>44</b>
<b>Fase 4: Menú</b>	<b>52</b>
<b>Fase 5: Pantalla de selecció de nivell</b>	<b>71</b>
<b>Fase 6: Nivell 1</b>	<b>81</b>
<b>Fase 7: Implementació de Nivells 2 i 3</b>	<b>109</b>
<b>Fase 8: Imatges amb Picasso i Intents implícits</b>	<b>113</b>
<b>Fase 9: Llistar jugadors</b>	<b>134</b>

# Introducció

Projecte: creem un joc que enllaça amb firebase i ens permet practicar la programació en l'entorn Android Studio i conèixer el procés de desenvolupar una aplicació per a mòbil.

Inspirat al tutorial: [https://www.youtube.com/watch?v=LBRrW08\\_P8c&list=PLhcYacorV7U6HHVsfXnWodwEPI0E38BXD](https://www.youtube.com/watch?v=LBRrW08_P8c&list=PLhcYacorV7U6HHVsfXnWodwEPI0E38BXD)

Com a llenguatge utilitzarem **Kotlin**

Codelabs bàsics de Android Studio en kotlin: <https://developer.android.com/courses/kotlin-android-fundamentals/overview>

Utilitzarem la base de dades de **Firestore Realtime Database**; també es pot fer servir cloud firestore (és més fàcil accedir a les dades, estan distribuïdes en col·leccions i documents) <https://www.youtube.com/watch?v=t5yyc1XfQrs&t=43s>

Aquest tutorial s'ha de complementar amb una primera introducció a Kotlin i petits exercicis individuals Ex2 i Ex3

Per a editar el readme.md utilitzem Dillinger com a patró <https://dillinger.io/> i editem desde **cmd** amb **notepad readme.md**

# Enunciat del projecte

Els alumnes han de crear un joc similar a Montserratak, que compleixi els següents requisits:

1. Ha de tenir una Pantalla splash
2. Identificació amb Firebase
3. Utilitzar una base de dades de jugadors feta amb Realtime Database o Cloud Firestore, i utilització del servei Storage de Firebase
4. Desenvolupar un joc (contrarellotge, puzzle, preguntes i respostes, mots encreuats, sopa de lletres, 3 en ratlla, joc del ahorcado, etc)
5. El joc ha de tenir un mínim de 3 nivells i una pantalla on s'indiqui a quin nivell s'està.
6. La puntuació de cada jugador ha de guardar-se a la base de dades
7. S'ha de poder llistar tots els jugadors i les seves puntuacions amb un RecyclerView. (No està implementat a Montserratak)
8. Si piquem sobre un jugador hem d'accedir a una finestra on podem veure totes les dades (edat, població, etc). Aquesta part no està implementada a Montserratak
9. S'ha d'utilitzar la llibreria Picasso per a gestió d'imatges
10. El projecte ha d'incloure un segon idioma (Montserratak només en té un)
11. Ha d'haver una opció per a canviar la contrasenya (Montserratak no en té)
12. El jugador ha de poder canviar la seva imatge de perfil agafant una imatge de la galeria o directament de la càmera (Aquesta part no està implementada a Montserratak)
13. Les imatges s'han de desar utilitzant Storage de Firebase
14. S'ha de fer una pantalla de crèdits utilitzant Fragments amb: el logo de l'escola, la data i els alumnes que hi han participat. La pantalla té un botó que porta al menú principal (no implementat a Montserratak)
15. Utilitzant MediaPlayer afegir música a la pantalla splash (no implementat)
16. Utilitzant Soudpool afegir so al joc (no implementat)

# Fase 1: Configuració de Firebase i pantalla splash

Creem una carpeta i la vinculem a git

```
sergi@EDUDLR0E68P1 MINGW64 ~  
$ mkdir montserratak
```

```
sergi@EDUDLR0E68P1 MINGW64 ~  
$ cd montserratak
```

```
sergi@EDUDLR0E68P1 MINGW64 ~/montserratak  
$ git init  
Initialized empty Git repository in C:/Users/sergi/montserratak/.git/
```

Ara creem la aplicació a Android Studio

**New Project**

**Empty Activity**

Creates a new empty activity

Name: Montserratak

Package name: com.example.montserratak

Save location: C:\Users\sergi\montserratak

Language: Kotlin

Minimum SDK: API 21: Android 5.0 (Lollipop)

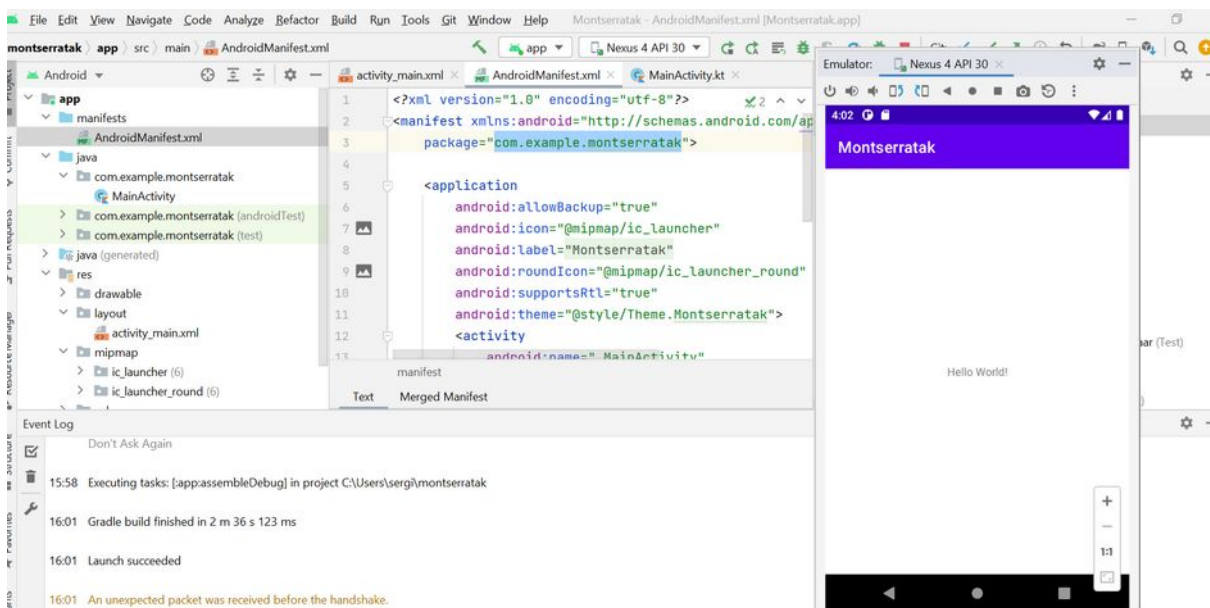
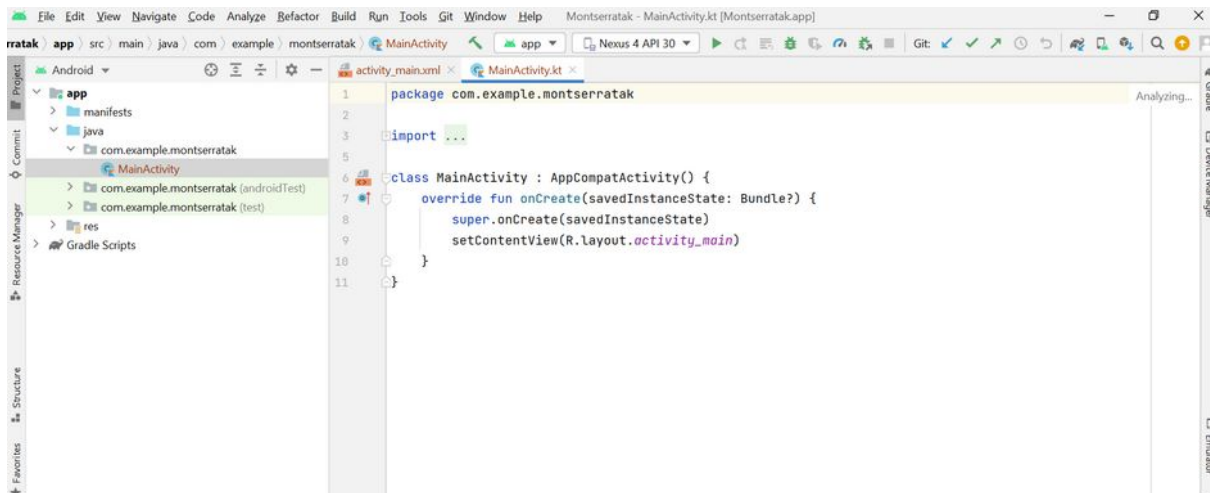
**i** Your app will run on approximately **98,6%** of devices.  
[Help me choose](#)

Use legacy android.support libraries [?](#)  
Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

**⚠** 'montserratak' already exists at the specified project location and it is not empty.

Previous Next Cancel **Finish**

Que apunti al directori que hem creat



Pujarem el projecte a github

```
sergi@EDUDLR0E68P1 MINGW64 ~/montserratak (main)
$ git add .
```

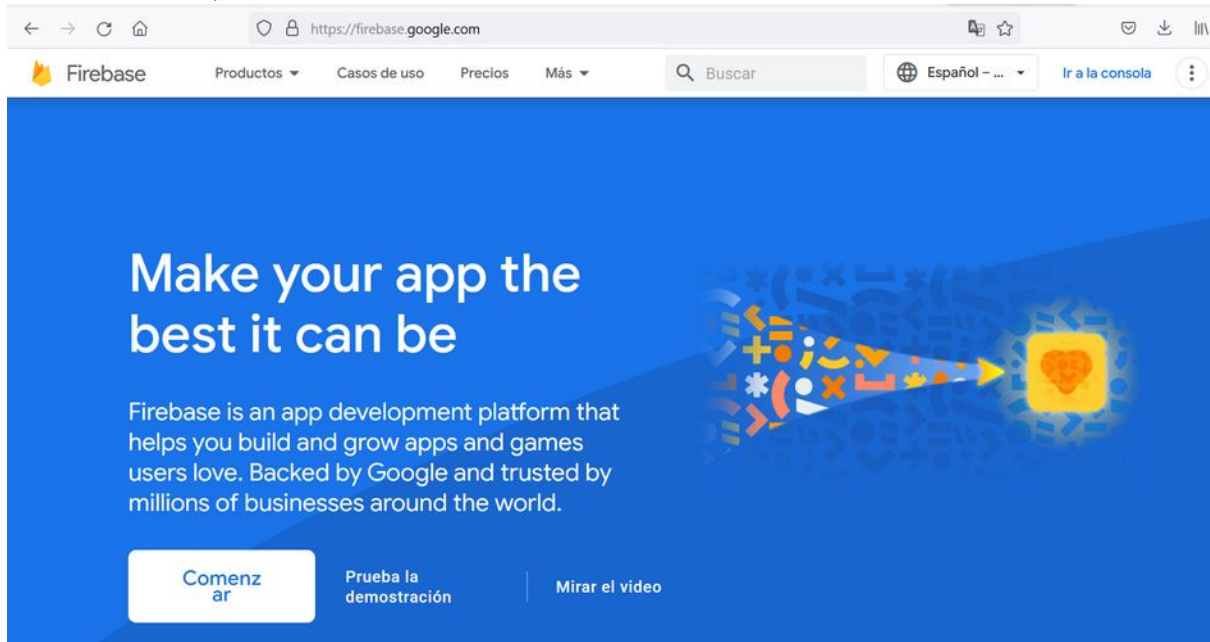
```
sergi@EDUDLR0E68P1 MINGW64 ~/montserratak (main)
$ git commit -m "1a pujada"
```

```
sergi@EDUDLR0E68P1 MINGW64 ~/montserratak (main)
$ git remote add origin https://github.com/sgotarra/montserratak.git
```

```
sergi@EDUDLR0E68P1 MINGW64 ~/montserratak (main)
$ git branch -M main

sergi@EDUDLR0E68P1 MINGW64 ~/montserratak (main)
$ git push -u origin main
Enumerating objects: 73, done.
Counting objects: 100% (73/73), done.
Delta compression using up to 4 threads
Compressing objects: 100% (54/54), done.
Writing objects: 100% (73/73), 97.17 KiB | 12.15 MiB/s, done.
Total 73 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/sgotarra/montserratak.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Ara entrem a Firebase i ens identifiquem amb el compte de google que vulguem (no fer servir el de xtec)



The screenshot shows the Firebase website homepage. The browser address bar displays "https://firebase.google.com". The navigation menu includes "Productos", "Casos de uso", "Precios", and "Más". A search bar with the text "Buscar" is present. The main content area features the headline "Make your app the best it can be" and a sub-headline: "Firebase is an app development platform that helps you build and grow apps and games users love. Backed by Google and trusted by millions of businesses around the world." Below the text are three buttons: "Comenzar", "Prueba la demostración", and "Mirar el video". The background is a blue gradient with a graphic of a yellow heart and various colorful symbols.

Comenzar

× Crear un proyecto(paso 1 de 3)

## Comencemos con el nombre de tu proyecto <sup>?</sup>

Nombre del proyecto

montserratak

✎ montserratak-bdbc5

📄 Seleccionar recurso superior

Continuar

## Configurar Google Analytics

Elige o crea una cuenta de Google Analytics <sup>?</sup>

📊 Default Account for Firebase



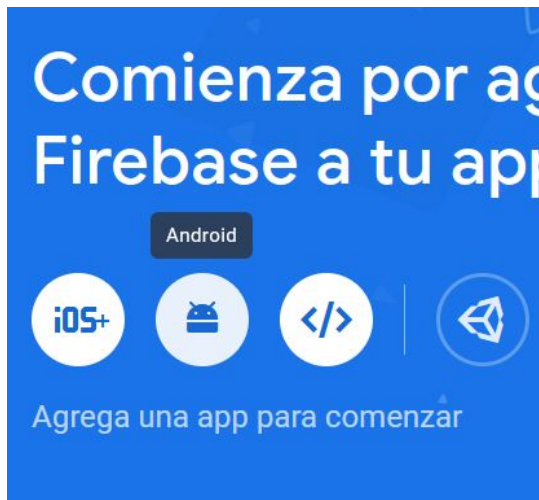
montserratak

✔ Tu proyecto nuevo está listo

Continuar

Fem clic a android





Ara ens cal del manifest, el nom del paquet, i un valor sha1 que s'aconsegueix, anant al directori del document per la consola de sistema i fent **gradlew signingReport**

```
Indicador d'ordres
C:\Users\sergi\montserratak>gradlew signingReport

Welcome to Gradle 7.2!

Here are the highlights of this release:
- Toolchain support for Scala
- More cache hits when Java source files have platform-specific line endings
- More resilient remote HTTP build cache behavior

For more details see https://docs.gradle.org/7.2/release-notes.html

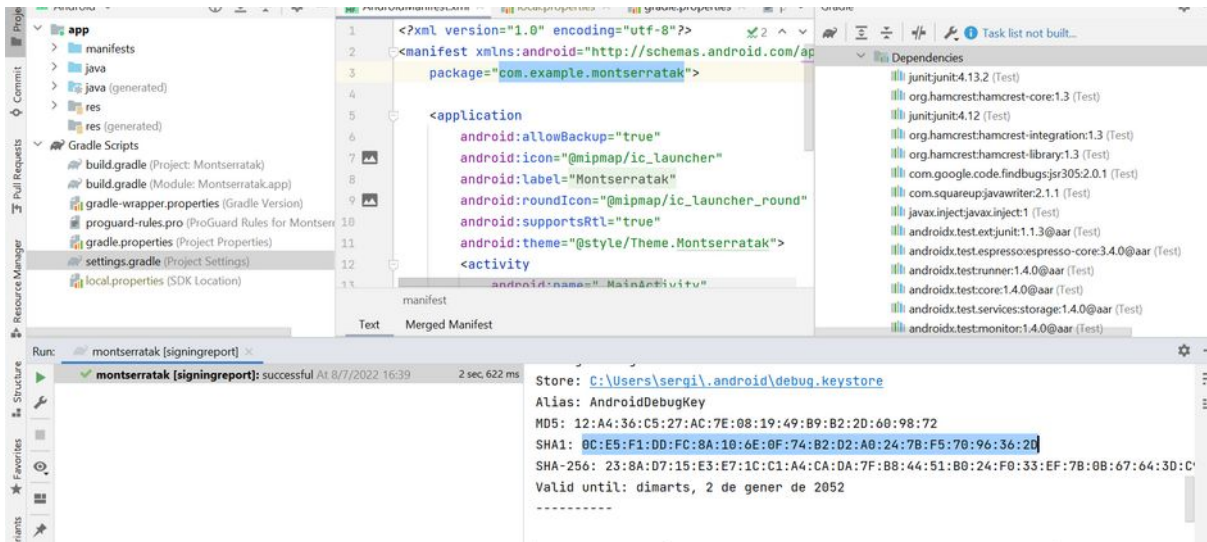
Starting a Gradle Daemon, 1 incompatible Daemon could not be reused, use --status for details

> Task :app:signingReport
Variant: debug
Config: debug
Store: C:\Users\sergi\.android\debug.keystore
Alias: AndroidDebugKey
MD5: 12:A4:36:C5:27:AC:7E:08:19:49:B9:B2:2D:60:98:72
SHA1: 0C:E5:F1:DD:FC:8A:10:6E:0F:74:B2:D2:A0:24:7B:F5:70:96:36:2D
SHA-256: 23:8A:D7:15:E3:E7:1C:C1:A4:CA:DA:7F:B8:44:51:B0:24:F0:33:EF:7B:0B:67:64:3D:C9:2C:06:2F:4C:EE:07
Valid until: dimarts, 2 de gener de 2052
-----
Variant: release
Config: null
Store: null
Alias: null
-----
Variant: debugAndroidTest
Config: debug
Store: C:\Users\sergi\.android\debug.keystore
Alias: AndroidDebugKey
MD5: 12:A4:36:C5:27:AC:7E:08:19:49:B9:B2:2D:60:98:72
SHA1: 0C:E5:F1:DD:FC:8A:10:6E:0F:74:B2:D2:A0:24:7B:F5:70:96:36:2D
SHA-256: 23:8A:D7:15:E3:E7:1C:C1:A4:CA:DA:7F:B8:44:51:B0:24:F0:33:EF:7B:0B:67:64:3D:C9:2C:06:2F:4C:EE:07
Valid until: dimarts, 2 de gener de 2052
-----
BUILD SUCCESSFUL in 13s
1 actionable task: 1 executed
```

## Una altra manera de aconseguir el SHA1

Anem a Gradle a la pestanya de la dreta, i piquem sobre l'elefant, s'obrirà una finestra de comandes que diu run anything: allà escrivim: **gradle signingreport**

I ara, ja tenim la clau a la finestra de run



Amb aquests dos valors ja podem tornar a firebase

**1 Registrar app**

Nombre del paquete de Android ⓘ

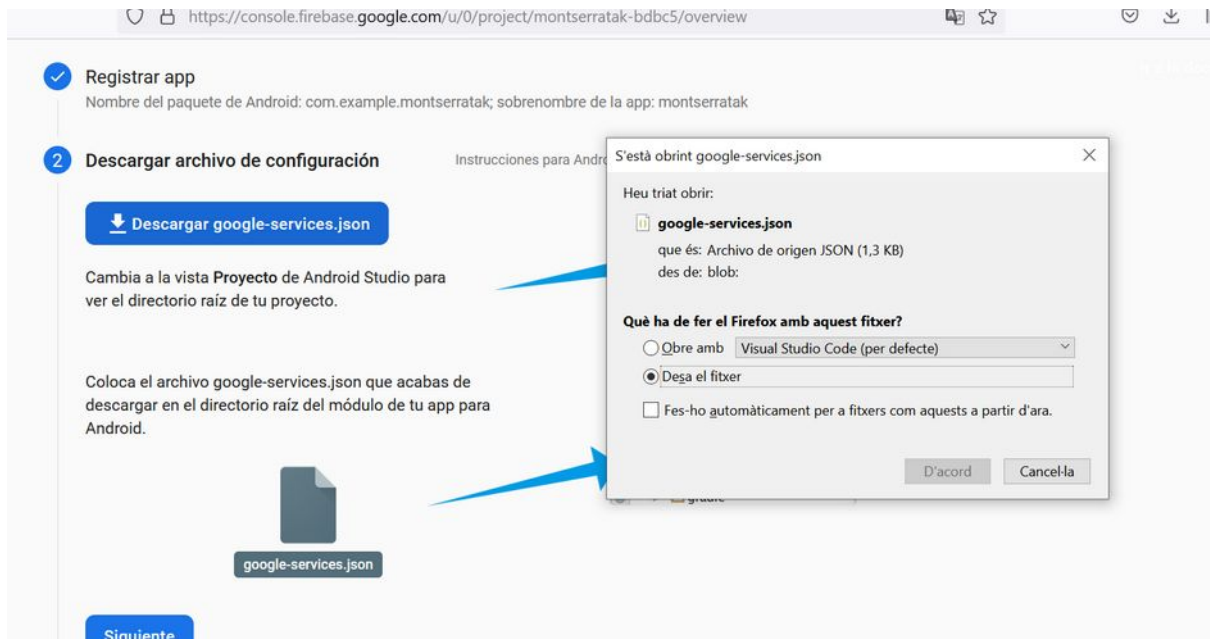
Sobrenombre de la app (opcional) ⓘ

Certificado de firma SHA-1 de depuración (opcional) ⓘ

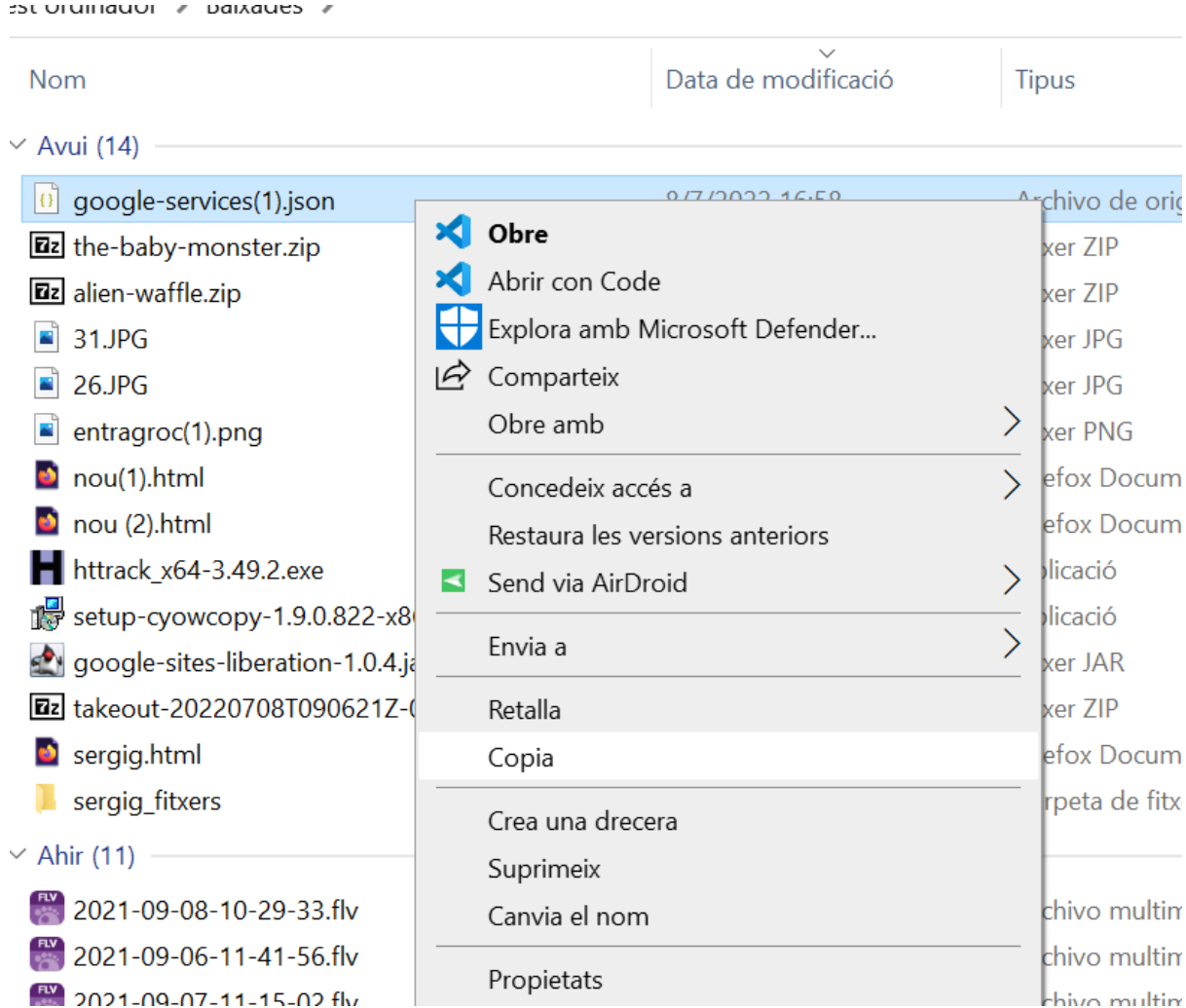
Obligatoria para Dynamic Links y el Acceso con Google o la asistencia con un número de teléfono en Auth. Puedes editar las claves SHA-1 en Configuración.

**Registrar app**

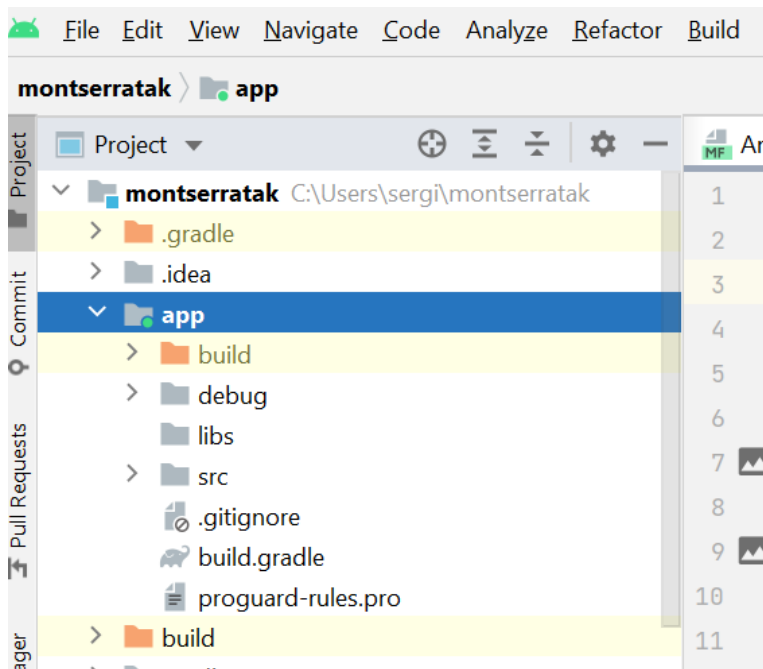
Descarreguem el fitxer



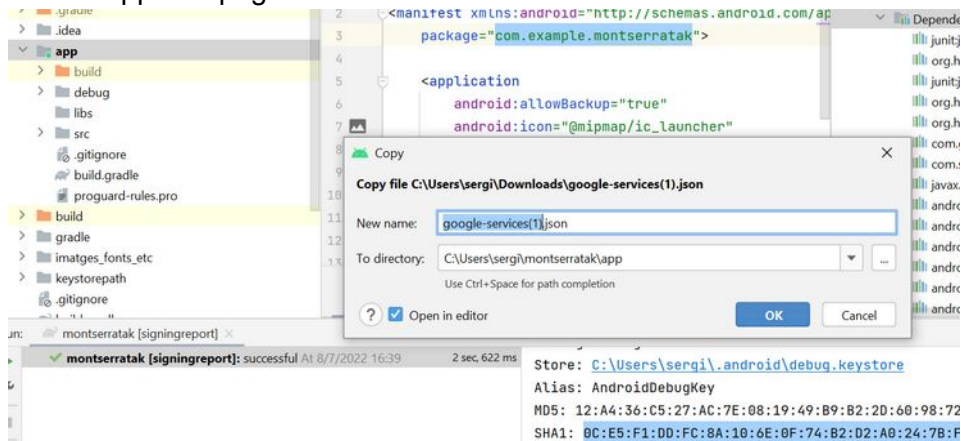
Anem a downloads i fem copiar sobre l'arxiu



Ara a Android Studio, canviem la vista a projecte



i sobre app fem pegar



I ja el tindrem carregat

Ara agreguem els sdks a gradle

### 3 Agregar el SDK de Firebase

Instrucciones para Gradle | [Unity](#) [C++](#)

El complemento de los servicios de Google para [Gradle](#) carga el archivo `google-services.json` que acabas de descargar. Modifica tus archivos `build.gradle` para usar el complemento.

Archivo `build.gradle` de nivel de proyecto (`<project>/build.gradle`):

```
buildscript {
    repositories {
        // Check that you have the following line (if not, add it):
        google() // Google's Maven repository
    }
    dependencies {
        ...
        // Add this line
        classpath 'com.google.gms:google-services:4.3.13'
    }
}

allprojects {
    ...
    repositories {
        // Check that you have the following line (if not, add it):
        google() // Google's Maven repository
        ...
    }
}
```

Java  Kotlin

Archivo `build.gradle` de nivel de app (`<project>/<app-module>/build.gradle`):

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
    // Import the Firebase BoM
    implementation platform('com.google.firebase:firebase-bom:30.2.0')

    // Add the dependency for the Firebase SDK for Google Analytics
    // When using the BoM, don't specify versions in Firebase dependencies
    implementation 'com.google.firebase:firebase-analytics-ktx'

    // Add the dependencies for any other desired Firebase products
    // https://firebase.google.com/docs/android/setup#available-libraries
}
```

Afegim aquestes dades als gradle (aplicació i projectes)

**Han quedat així (els meus)**

Project:

```
// Top-level build file where you can add configuration options
```

*common to all sub-projects/modules.*

```
buildscript {
    repositories {
        google() // Google's Maven repository
    }

    dependencies {
        classpath 'com.google.gms:google-services:4.3.13'
    }
}

plugins {
    id 'com.android.application' version '7.1.2' apply false
    id 'com.android.library' version '7.1.2' apply false
    id 'org.jetbrains.kotlin.android' version '1.5.30' apply false
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

### **Module:**

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
}

apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'

android {
    compileSdk 32

    defaultConfig {
        applicationId "com.neulaworks.montserratak"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner
        "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
```



```

        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
}

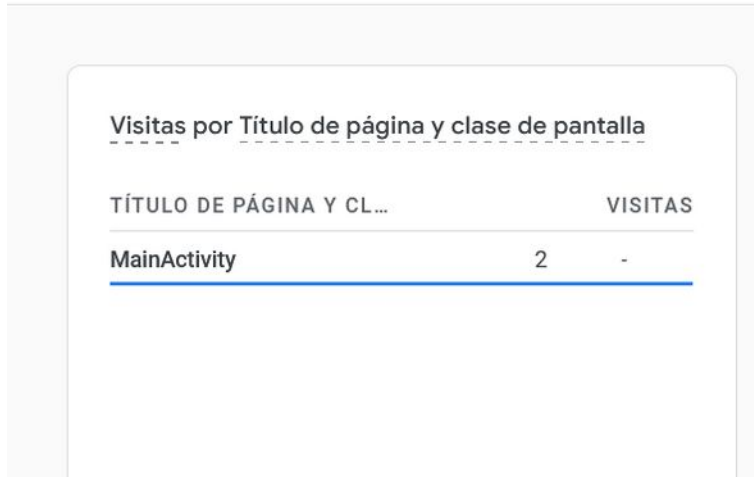
dependencies {
    implementation platform('com.google.firebase:firebase-
bom:30.2.0')
    implementation 'com.google.firebase:firebase-analytics-ktx'
    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.4.2'
    implementation 'com.google.android.material:material:1.6.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.4.0'
}

```

Per provar a veure si funciona executem l'aplicació a l'emulador (ha de tenir accés a internet) i seguidament anem a firebase i mirem a analítics i ha de sortir com a mínim una visita

montserratak ▾

 Analytics | Dashboard



Visitas por Título de página y clase de pantalla

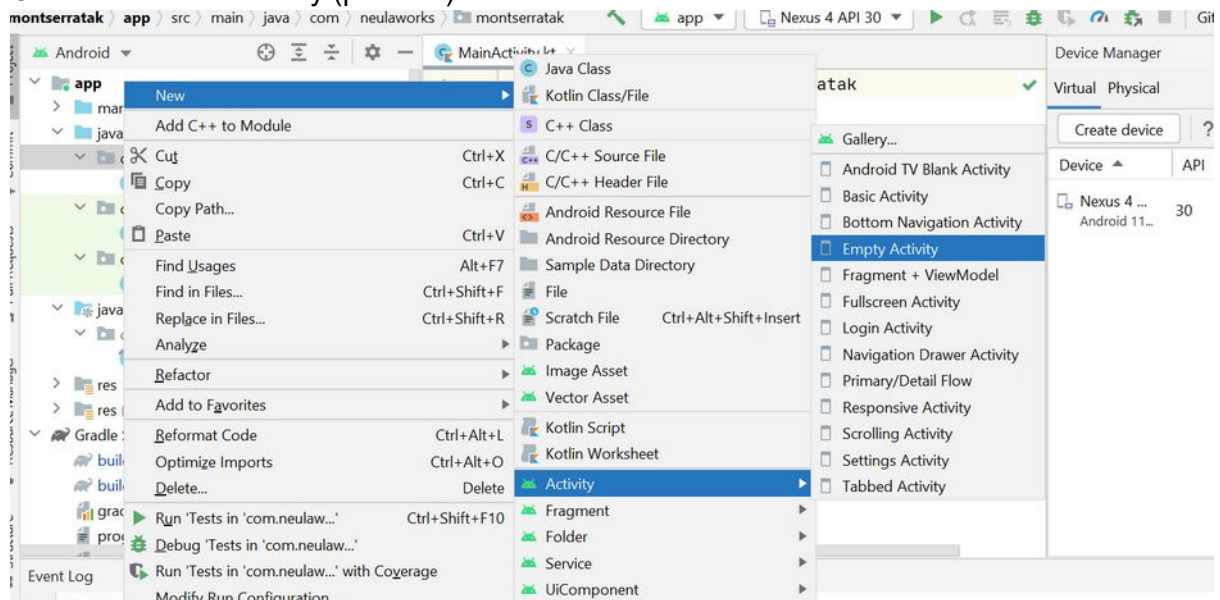
TÍTULO DE PÁGINA Y CL...	VISITAS
MainActivity	2

## Pantalla d'splash

(pantalla que s'executa uns segons al principi i després s'executa l'aplicació)

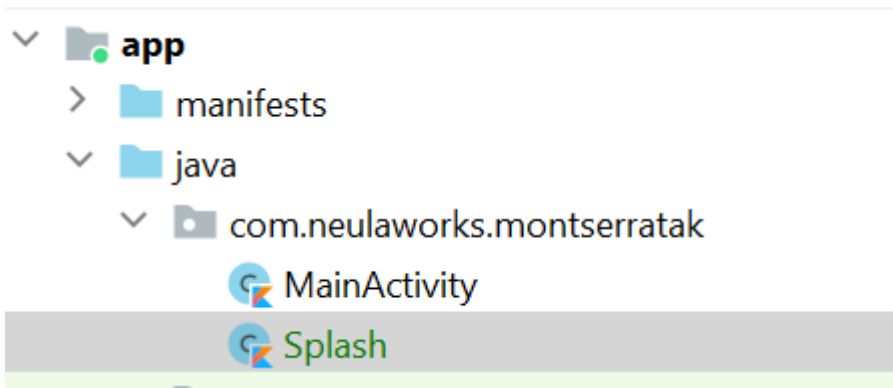
Utilitzem el video: [Crear Splash Screen en Android Studio con Kotlin](#)

## Creem una nova activity (pantalla)



Un cop creada





Anem al manifest

És un arxiu de configuració on hem de declarar la configuració necessària, (si pot fer servir internet, la càmera, localització...)

```
<activity
    android:name=".Splash"
    android:exported="false" />
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Aquí veiem que la MainActivity s'executa perquè té assignat el Launcher; com volem que l'splash sigui la primera activitat que s'executi.

Canviem així:

```
<activity android:name=".Splash"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER"
/>
    </intent-filter>
</activity>

<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.DEFAULT"
/>
```

```
        </intent-filter>
    </activity>
</application>
```

Ara canviem el codi de splash.kt

```
class Splash : AppCompatActivity() {
    private val duracio: Long=3000;

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)

        //amaguem la barra, pantalla a full
        supportActionBar?.hide()

        canviarActivity();
    }

    private fun canviarActivity(){
        Handler().postDelayed({
            val intent=Intent(this,MainActivity::class.java)
            startActivity(intent)
        }, duracio)
    }
}
```

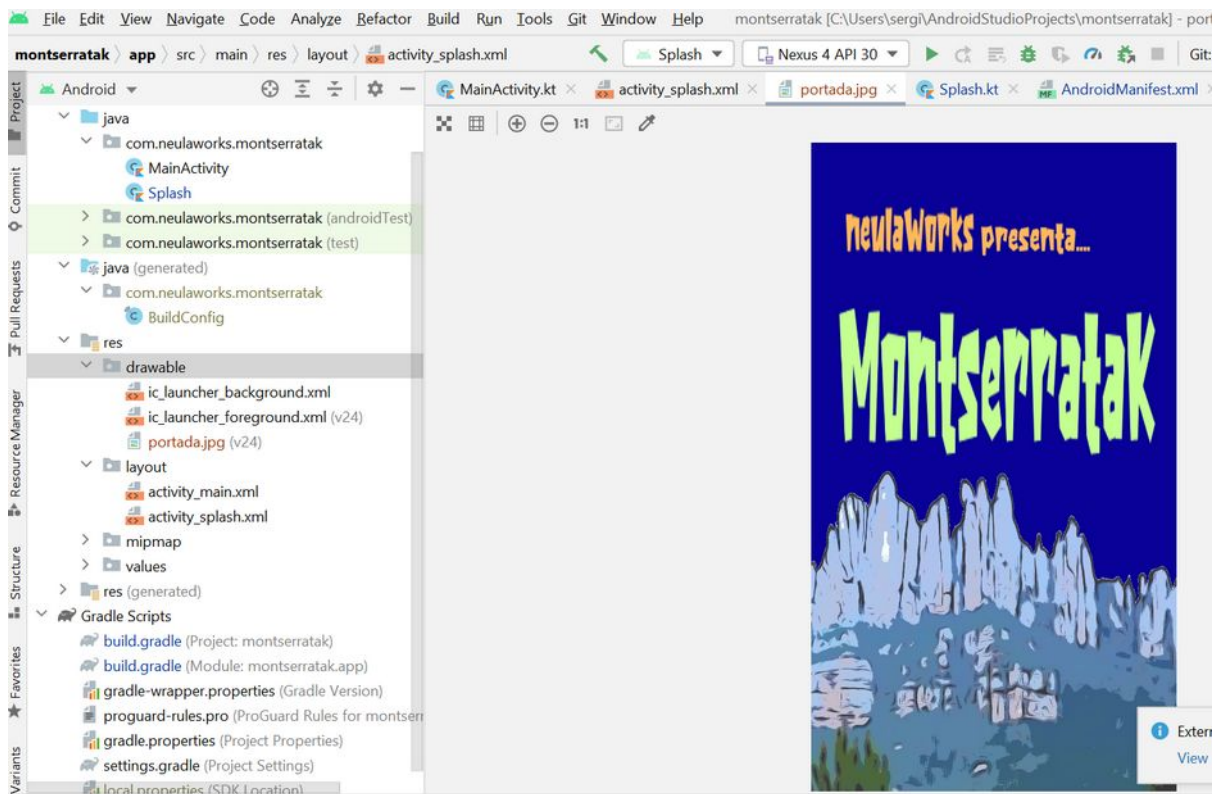
Fem un paréntesis per parlar de variables que poden ser Null. Kotlin no permet variables Null, només aquelles que previament s'han definit com variable?, indicant que poden tenir un valor null. És interesant de veure: <https://khan.github.io/kotlin-for-python-developers/#null-safety> per entendre com funciona el símbol ?

També podem mirar <https://kotlinlang.org/docs/null-safety.html#checking-for-null-in-conditions> per entendre com funcionen els operadors !! i ?:

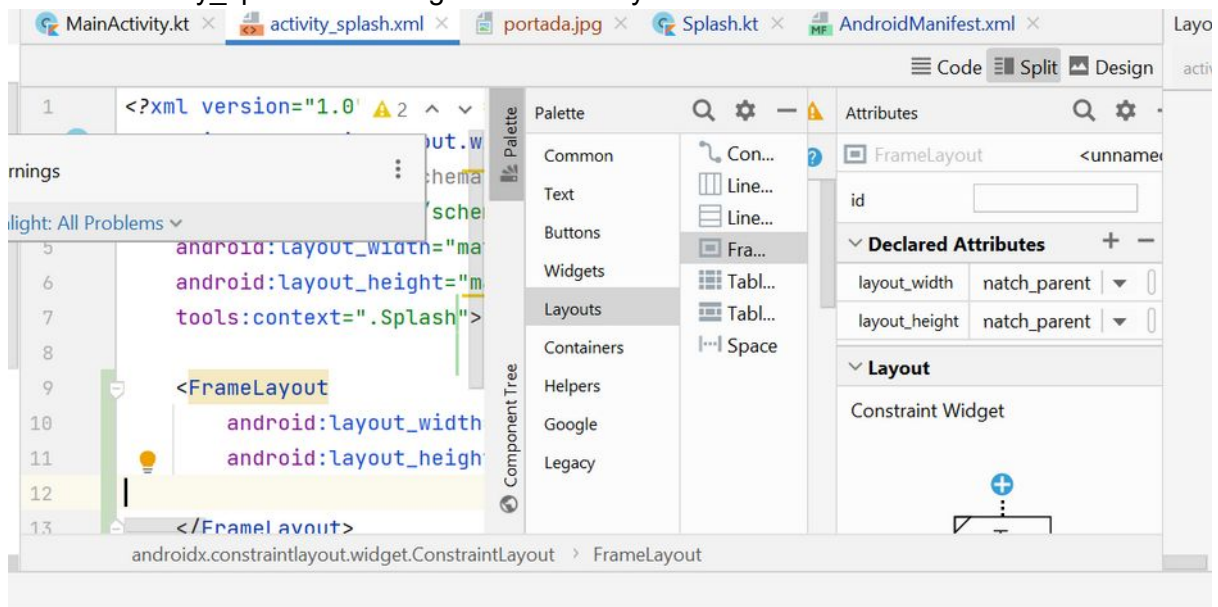
Continuant, handler ens surt amb una ratlla a sobre, perquè es deprecated, després en parlarem.

Ara posarem la imatge que farem servir per a la pantalla de càrrega i per fer-ho farem servir frame layout (veiem un tutorial aquí) [Android desde Cero - Frame Layout](#)

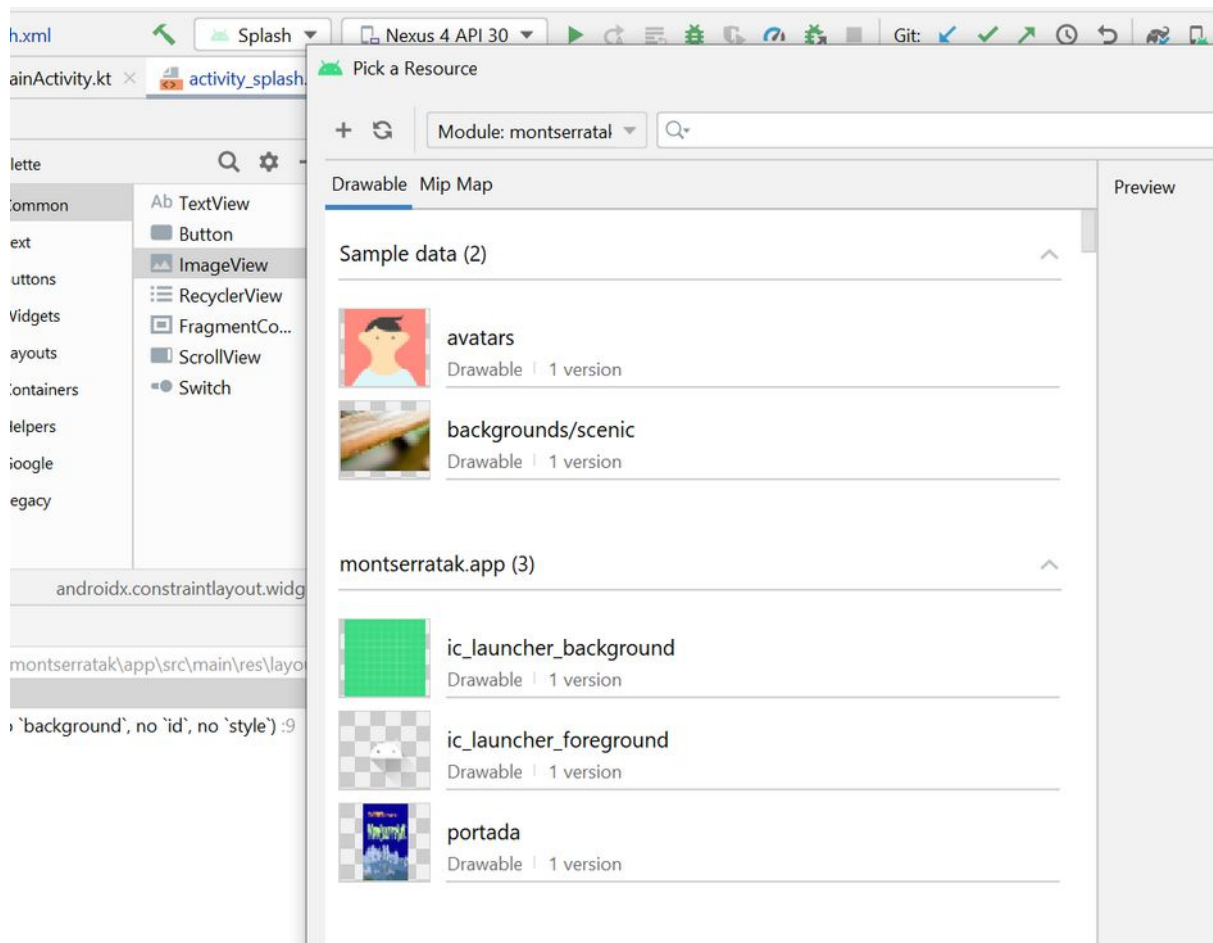
Per a posar la imatge (ha de ser tot en minúscules i sense números), la arrosseguem a drawable (la proporció de la imatge ha de ser 16:9 per exemple 1600x900)



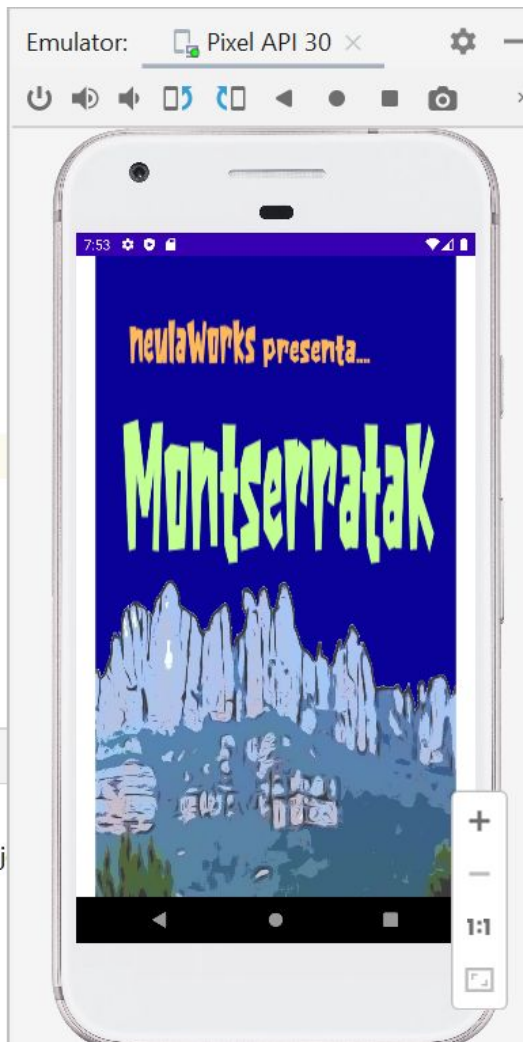
Anem a Activity\_splash.xml i afegim un frame layout



Arrosseguem al centre una imageview i sel·leccionem portada



Ara provem si funciona fins aquí l'aplicació



Per a evitar el marc blanc i centrar la imatge fem una mica de padding i posem un fons verd; també es pot fer que la imatge s'estiri fins a ocupar tota la pantalla, ho veurem també més endavant.

Volem el color verd marciano  
per aixó anem a <https://html-color-codes.info/>



I agafem el valor #58FA58

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="#58FA58"
  android:padding="15dp"
  tools:context=".Splash">
```



Anem a canviar el Manifest i posarem que no es permeti girar la pantalla (no hi haurà landscape)

```
<activity android:name=".Splash"
  android:exported="true"
  android:screenOrientation="portrait" >
```

Millorem l'aplicació eliminant Handler que està deprecated en Kotlin  
I utilitzem la éina Timer

a import:

```
import java.util.Timer
import kotlin.concurrent.schedule
```

**I el codi queda:**

```
class Splash : AppCompatActivity() {
    private val duracio: Long=10000

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)

        //amaguem la barra, pantalla a full
        supportActionBar?.hide()
        //cridem a la funció de canviar activitat
        canviarActivity()

    }

    private fun canviarActivity(){

        Timer().schedule(duracio){
            saltainici()

        }
    }
    fun saltainici()
    {
        val intent=Intent(this, MainActivity::class.java)
        startActivity(intent)
    }
}
```

**Tasques pendents: Afegir una musiqueta a splash utilitzant MediaPlayer**

## Fase 2: Registre i Login de jugadors

Ara configurem el layout de la activity main

Primer canviem a RelativeLayout (Per saber més de com configura RelativeLayout:

<https://developer.android.com/guide/topics/ui/layout/relative> )

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://sc...
```

I a dins col·loquem un LinearLayout (és un dels dissenys més senzills, col·loca els components alineats o horitzontalment o verticalment)

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"></LinearLayout>
```

També afegim gravetat: central

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"></LinearLayout>
```

i canviarem el background del layout amb el fons verd marciano

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#58FA58"
    tools:context=".MainActivity">
```

Ara afegirem 2 botons però abans anem al arxiu d'strings i creem dues variables



```
1 <resources>
2     <string name="app_name">montserratak</string>
3
4     <string name="BTMLOGIN">Login</string>
5     <string name="BTMREGISTRO">Registre</string>
6
7 </resources>
```

Ja podem incorporar els botons (per entendre com funciona el layout verue el tutorial **Layouts en android studio**)

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#58FA58"
tools:context=".MainActivity">
```

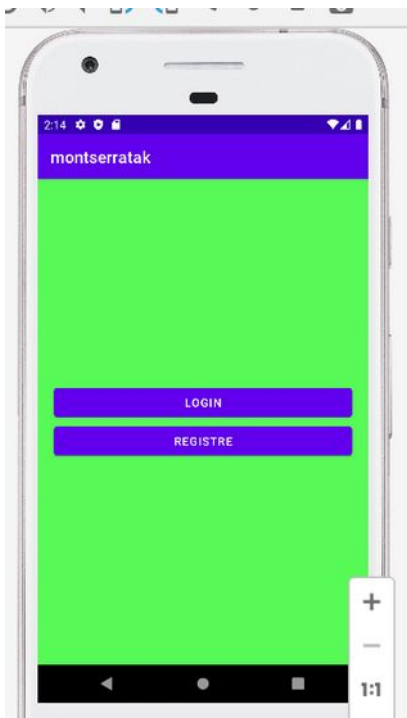
```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20dp"
    android:gravity="center">
    <!-- botó de login -->
    <Button
        android:id="@+id/BTMLOGIN"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/BTMLOGIN"/>
    <!-- botó de registrar-se -->
    <Button
```

```

        android:id="@+id/BTMREGISTRO"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/BTMREGISTRO"
    />
</LinearLayout>
</RelativeLayout>

```

Ara el text del botó va referenciat a la string, això va bé per a canviar-lo i també per a afegir idiomes a la web



Ara donarem funcionalitat als botons

```

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        var BTMLOGIN = findViewById<Button>(R.id.BTMLOGIN);
        var BTMREGISTRO = findViewById<Button>(R.id.BTMREGISTRO);

        BTMLOGIN.setOnClickListener() {
            Toast.makeText(this, "click botó
login", Toast.LENGTH_LONG).show();
        }

        BTMREGISTRO.setOnClickListener() {
            Toast.makeText(this, "click botó

```

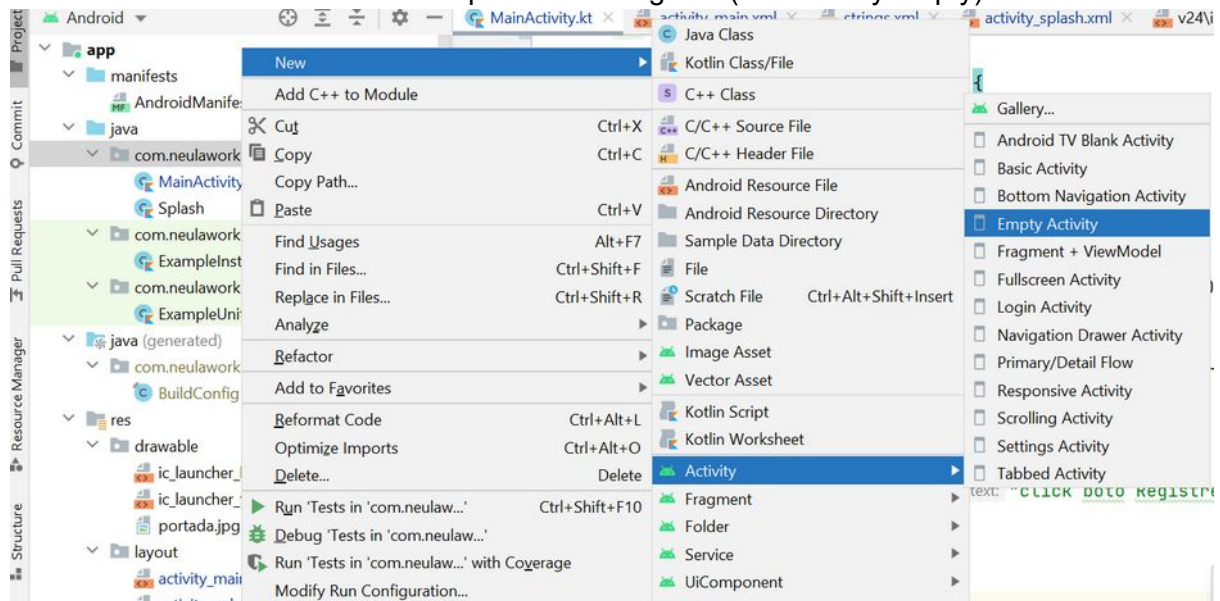
```

    registre", Toast.LENGTH_LONG).show();
    }
}
}

```

Creem 2 variables (es poden dir com es vulguin, hem posat aquests noms perquè siguin fàcils d'identificar i provem el codi amb uns toasts)  
Si això funciona, anem a implementar el registre

Per fer-ho creem una classe nova que es dirà Registro (una activity empty)



ara afegirem els sdk de firebase a la app, com diu aquí:

<https://firebase.google.com/docs/android/setup?hl=es-419#kotlin+ktx>

#### Paso 4: Agrega los SDK de Firebase a tu app

1. Usa la BoM de Firebase para Android y declara las dependencias de los productos de Firebase que quieres usar en tu app. Decláralas en el archivo Gradle (generalmente `app/build.gradle`) de tu módulo (nivel de app).



```
dependencies {
    // ...

    // Import the Firebase BoM
    implementation platform('com.google.firebase:firebase-bom:30.1.0')

    // When using the BoM, you don't specify versions in Firebase library dependencies

    // Declare the dependency for the Firebase SDK for Google Analytics
    implementation 'com.google.firebase:firebase-analytics-ktx'

    // Declare the dependencies for any other desired Firebase products
    // For example, declare the dependencies for Firebase Authentication and Cloud Firestore
    implementation 'com.google.firebase:firebase-auth-ktx'
    implementation 'com.google.firebase:firebase-firestore-ktx'
}
```

Si usas la BoM de Firebase para Android, tu app siempre utilizará versiones compatibles de las bibliotecas de Firebase para Android.

2. Sincroniza tu app para garantizar que todas las dependencias tengan las versiones necesarias.

#### També necessitarem una base de dades

<https://firebase.google.com/docs/android/setup?hl=es-419#kotlin+ktx>

Producto	Identificador	Versión	Estado
Authentication	com.google.firebase:firebase-auth-ktx	21.0.6	
Cloud Firestore	com.google.firebase:firebase-firestore-ktx	24.2.0	
SDK cliente de Cloud Functions para Firebase	com.google.firebase:firebase-functions-ktx	20.1.0	
Cloud Messaging	com.google.firebase:firebase-messaging-ktx	23.0.6	✓
Cloud Storage	com.google.firebase:firebase-storage-ktx	20.0.1	
Crashlytics	com.google.firebase:firebase-crashlytics-ktx	18.2.11	✓
NDK de Crashlytics	com.google.firebase:firebase-crashlytics-ndk	18.2.11	✓
Complemento de Crashlytics	com.google.firebase:firebase-crashlytics-gradle	2.9.1	✓
Compatibilidad con módulos de funciones dinámicas	com.google.firebase:firebase-dynamic-module-support	16.0.0-beta01	
Dynamic Links	com.google.firebase:firebase-dynamic-links-ktx	21.0.1	✓
In-App Messaging	com.google.firebase:firebase-inappmessaging-ktx	20.1.2	✓ (obligatorio)
Visualización de In-App Messaging	com.google.firebase:firebase-inappmessaging-display-ktx	20.1.2	✓ (obligatorio)
Instalaciones de Firebase	com.google.firebase:firebase-installations-ktx	17.0.1	
API de Firebase ML Model Downloader	com.google.firebase:firebase-ml-modeldownloader-ktx	24.0.3	
Performance Monitoring	com.google.firebase:firebase-perf-ktx	20.1.0	
Complemento de Performance Monitoring	com.google.firebase:perf-plugin	1.4.1	
Realtime Database	com.google.firebase:firebase-database-ktx	20.0.5	
Remote Config	com.google.firebase:firebase-config-ktx	21.1.0	✓

Al final les dependències del build gradle (module:app) queden així:  
dependencies {

```
implementation 'androidx.core:core-ktx:1.7.0'
implementation 'androidx.appcompat:appcompat:1.4.2'
implementation 'com.google.android.material:material:1.6.1'
implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
```

```

    // Import the Firebase BoM
    implementation platform('com.google.firebase:firebase-
bom:30.2.0')

    // Declare the dependency for the Firebase SDK for Google
Analytics
    implementation 'com.google.firebase:firebase-analytics-ktx'

    // Declare the dependencies for any other desired Firebase
products
    // For example, declare the dependencies for Firebase
Authentication and Cloud Firestore
    implementation 'com.google.firebase:firebase-auth-ktx'
    implementation 'com.google.firebase:firebase-firestore-ktx'

    //base de dades
    implementation 'com.google.firebase:firebase-database-ktx'

    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.4.0'
}

```

## Fem el disseny del Registre

Primerament afegim alguns noms a strings

```

<string name="correoEt">E-mail</string>
<string name="nombreEt">Nom</string>
<string name="passEt">Password</string>
<string name="fechaTxt">Data</string>
<string name="Registrar">REGISTRAR</string>

```

Després canviem el constraintLayout per un RelativeLayout

*RelativeLayout permet que vistes secundàries especifiquin la seva posició relativa a la vista superior o entre si (especificada per ID). D'aquesta manera, podem alinear dos elements per la vora dreta o fer que un estigui per sota de l'altre, al centre de la pantalla, al centre a l'esquerra, i així successivament.*

I a dins fem un LinearLayout vertical

Utilitzem colors i format similar al activity\_main.xml i afegim 4 edit\_text, un text\_view i un

## Button

El codi xml del layout queda així:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#58FA58"
    tools:context=".Registro">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical"
        android:padding="30dp">

        <!-- ENTREM EL MAIL -->
        <EditText
            android:id="@+id/correoEt"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress"
            android:hint="@string/correoEt" />
        <!-- forcem que la entrada sigui un correu electronic -->

        <!-- ENTREM EL PASSWORD -->
        <EditText
            android:id="@+id/passEt"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:hint="@string/passEt" />
        <!-- forcem que la entrada sigui un password -->

        <!-- ENTREM EL NOM DEL JUGADOR -->
        <EditText
            android:id="@+id/nombreEt"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```

        android:hint="@string/nombreEt" />

<!-- DATA ACTUAL -->
<TextView
    android:id="@+id/fechaEt"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:hint="@string/fechaTxt"
    android:textSize="20sp"
    android:layout_marginTop="10dp"
/>

<!-- BOTO DE ENTRAR -->
<Button
    android:id="@+id/Registrar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/Registrar"
    android:layout_marginTop="10dp"
/>

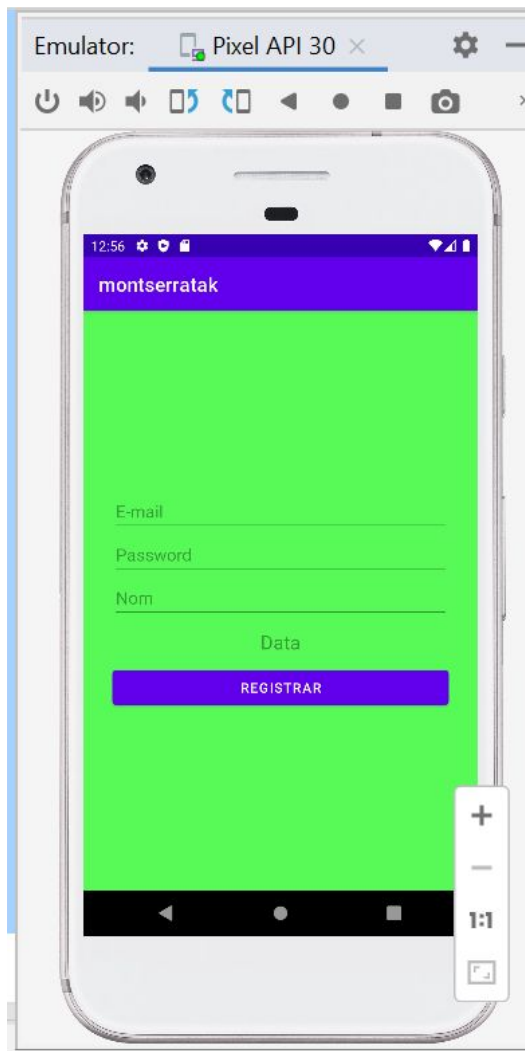
</LinearLayout>

</RelativeLayout>

```

Per entendre com funciona la resolució de pantalla **dp i sp**:

[https://www.altova.com/manual/es/MobileTogether/mobiletogetherdesigner/mtdobjsfeatures\\_sizes.html](https://www.altova.com/manual/es/MobileTogether/mobiletogetherdesigner/mtdobjsfeatures_sizes.html)



Ara codifiquem registro.kt

A la classe creem les variables que farem servir

```
//Definim les variables que farem servir
//lateinit ens permet no inicialitzar-les encara
lateinit var correoEt :EditText
lateinit var passEt :EditText
lateinit var nombreEt :EditText
lateinit var fechaTxt :TextView
lateinit var Registrar : Button
```

I a dins de onCreate, les inicialitzarem

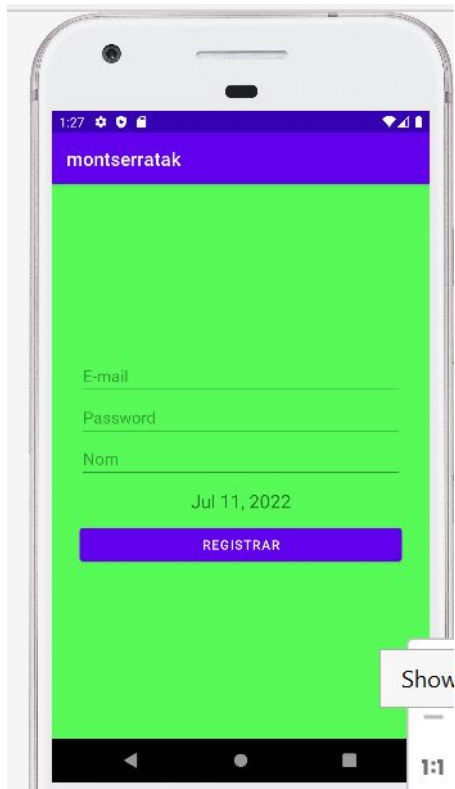
```
// Busquem a R els elements als que apunten les variables
correoEt =findViewById<EditText>(R.id.correoEt)
passEt =findViewById<EditText>(R.id.passEt)
nombreEt =findViewById<EditText>(R.id.nombreEt)
fechaTxt =findViewById<TextView>(R.id.fechaTxt)
Registrar =findViewById<Button>(R.id.Registrar)
```



Possem una data al textview de fechaTxt

```
//carreguem la data al TextView
//Utilitzem calendar (hi ha moltes altres opcions)
val date = Calendar.getInstance().time
val formatter = SimpleDateFormat.getDateInstance() //or use
getDateInstance()
val formattedDate = formatter.format(date)
//ara la mostrem al TextView
fechaTxt.setText(formattedDate)
```

Si provem l'aplicació:



Ara queda linkar amb la autenticació firebase

Primer afegim una variable firebase

```
lateinit var auth: FirebaseAuth //FIREBASE AUTENTIFICACIO
```

Ara la inicialitzem al onCreate

```
//Instanciem el firebaseAuth
auth = FirebaseAuth.getInstance()
```

Ara creem el codi pel botó, però abans de crear el jugador, fem una comprovació de email i password

per l'email hi ha un format per defecte que comprova automàticament

pel que fa al password únicament mirem que el contingut sigui = o major que 6

```
Registrar.setOnClickListener() {
    //Abans de fer el registre validem les dades
```

```

var email:String = correoEt.getText().toString()
var pass:String = passEt.getText().toString()

// validació del correu
// si no es de tipus correu
if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
    correoEt.setError("Invalid Mail")
}
else if (pass.length()<6) {
    passEt.setError("Password less than 6 chars")
}
else
{
    RegistrarJugador(email, pass)
}

```

Si passa els dos ifs, crida al mètode RegistrarJugador

Aquest mètode el copiem descaradament de la documentació de firebase

[https://firebase.google.com/docs/auth/android/password-auth?hl=es-419#kotlin+ctx\\_3](https://firebase.google.com/docs/auth/android/password-auth?hl=es-419#kotlin+ctx_3)

4. Para crear una cuenta nueva, pasa la dirección de correo electrónico y la contraseña del usuario nuevo a

`createUserWithEmailAndPassword`:

```

Java Kotlin+KTX
Android Android

auth.createUserWithEmailAndPassword(email, password)
    .addOnCompleteListener(this) { task ->
        if (task.isSuccessful) {
            // Sign in success, update UI with the signed-in user's information
            Log.d(TAG, "createUserWithEmail:success")
            val user = auth.currentUser
            updateUI(user)
        } else {
            // If sign in fails, display a message to the user.
            Log.w(TAG, "createUserWithEmail:failure", task.exception)
            Toast.makeText(baseContext, "Authentication failed.",
                Toast.LENGTH_SHORT).show()
            updateUI(null)
        }
    }
}

```

EmailPasswordActivity.kt

En comptes de log fem toasts i no fem servir el metode updateUI (per ara)

Veiem com queda el codi del mètode registrarJugador:

```

fun RegistrarJugador(email:String, passw:String){
    auth.createUserWithEmailAndPassword(email, passw)
        .addOnCompleteListener(this) { task ->

```

```

        if (task.isSuccessful) {
            // Sign in success, update UI with the signed-in
            user's information
            Toast.makeText(
                this, "createUserWithEmail:success", Toast.LENGTH_SHORT).show()
            val user = auth.currentUser
            updateUI(user)
        } else {
            Toast.makeText(baseContext, "Authentication
            failed.", Toast.LENGTH_SHORT).show()
            //updateUI(null)
        }
    }
}

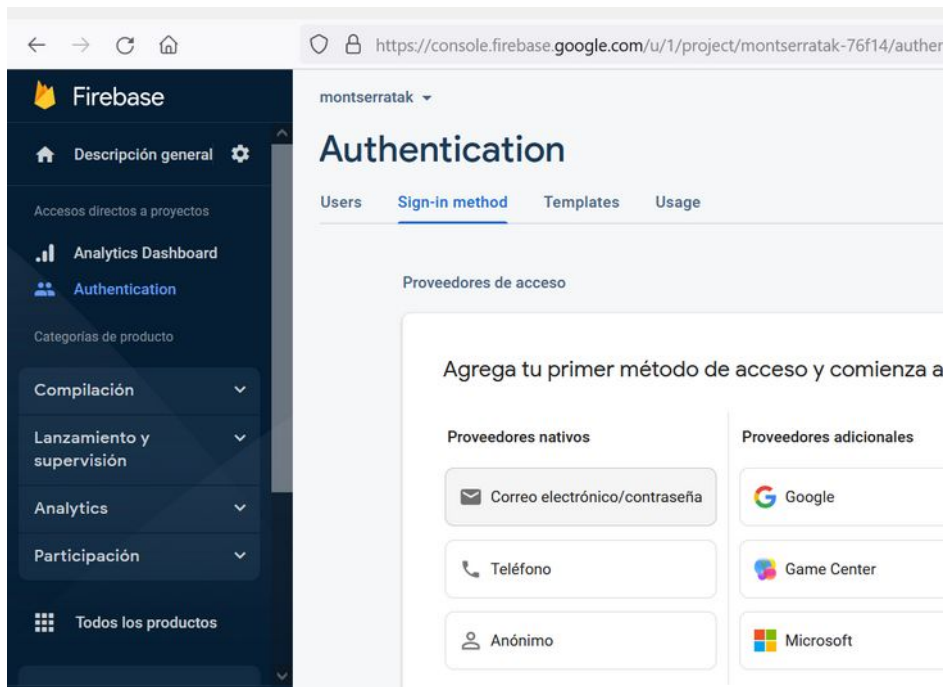
fun updateUI(user:FirebaseUser?) {
    //hi ha un interrogant perquè podria ser null
    if (user!=null)
    {
        var puntuacio: Int = 0
        var uidString: String = user.uid
        var correoString: String = correoEt.getText().toString()
        var passString: String = passEt.getText().toString()
        var nombreString: String = nombreEt.getText().toString()
        var fechaString: String= fechaTxt.getText().toString()

        //AQUI GUARDA EL CONTINGUT A LA BASE DE DADES
        // FALTA FER

    }
    else
    {
        Toast.makeText( this, "ERROR CREATE
        USER", Toast.LENGTH_SHORT).show()
    }
}
}

```

Registrar Jugador és un copy paste del que diu firebase. Ara anirem a la pàgina de firebase i permetrem crear nous usuaris amb el mail i password.



Correo electrónico/contraseña  Habilitar

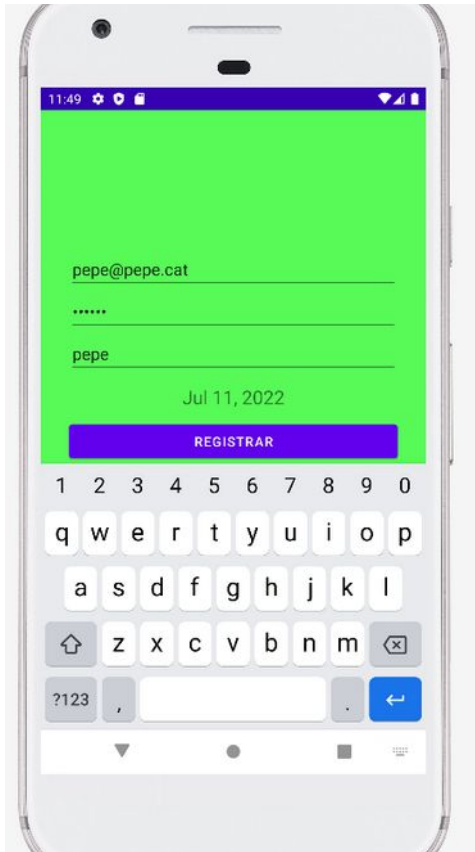
Permite que los usuarios se registren con su dirección de correo electrónico y contraseña. Nuestros SDK también proporcionan verificación de la dirección de correo electrónico, recuperación de contraseñas y primitivas de cambio de dirección de correo. [Más información](#)

Vínculo del correo electrónico (acceso sin contraseña)  Habilitar

Cancelar

Guardar

Ara si executem l'aplicació i entrem un usuari



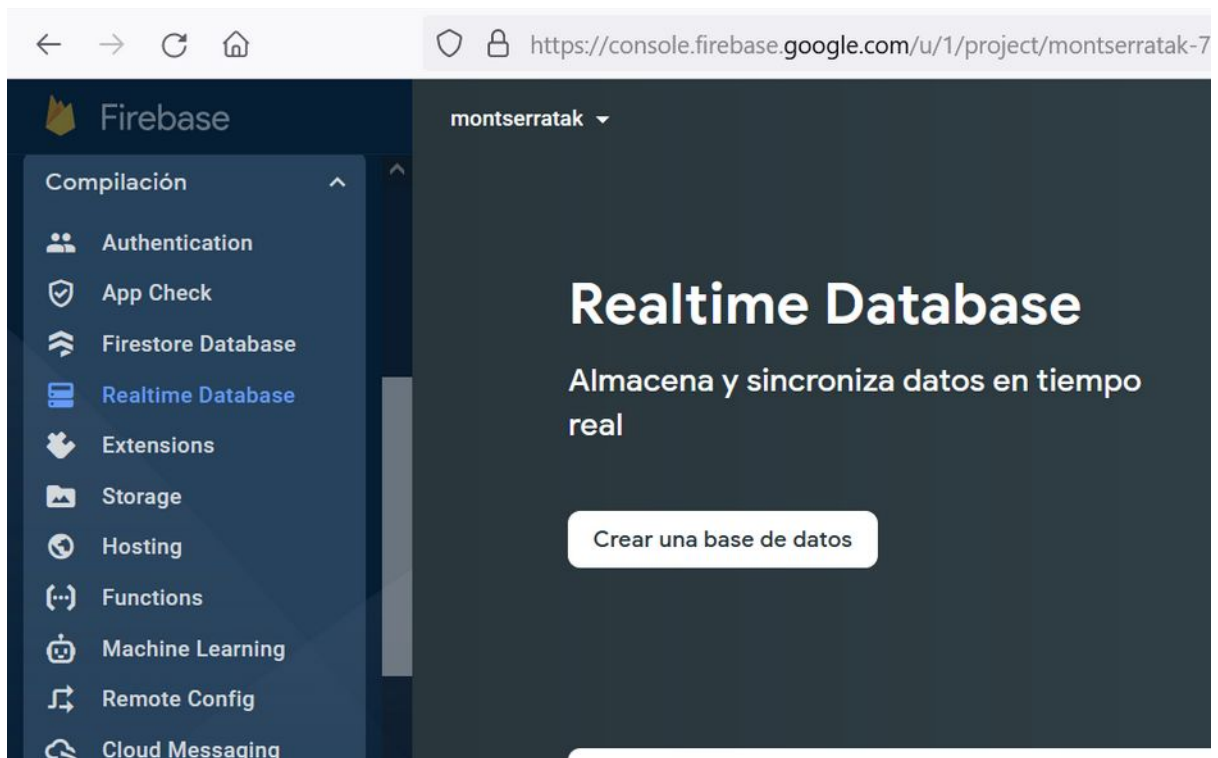
Podem trobar-lo a la base de dades

Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
pepe@pepe.cat		11 jul 2022	11 jul 2022	Yy9AP5tLNnZhEHabyS1BSzZk7aD3

També preparem la base de dades que utilitzarem a firebase



Anem a Realtime Database



## Configurar base de datos



- 1 Opciones de base de datos — 2 Reglas de seguridad

Cuando definas la estructura de los datos, **deberás crear reglas para protegerlos.**

[Más información](#)

**Comenzar en modo bloqueado**  
De forma predeterminada, tus datos son privados. El acceso de lectura/escritura de los clientes solo se otorgará como se indica en tus reglas de seguridad.

**Comenzar en modo de prueba**  
Para permitir una configuración rápida, los datos se abren de forma predeterminada. Sin embargo, debes actualizar las reglas de seguridad en un plazo de 30 días a fin de habilitar el acceso de lectura/escritura a largo plazo para los clientes.

```
{
  "rules": {
    ".read": "now < 1660082400000", // 2022-8-10
    ".write": "now < 1660082400000", // 2022-8-10
  }
}
```

**Las reglas de seguridad predeterminadas del modo de prueba permiten que cualquier usuario con acceso a tu referencia de base de datos pueda ver, editar y borrar todos los datos durante los siguientes 30 días.**

Cancelar

Habilitar

Inicialitzem en modo prova que no ens doni guerra amb els permisos (més endavant podem optar per a canviar-los)

Canviem les rules a true

montserratak ▾

# Realtime Database

Datos **Reglas** Copias de seguridad Uso

Editar reglas Supervisar reglas

**⚠** Tus reglas de seguridad están definidas como públicas,

```

1 {
2   "rules": {
3     ".read": true,
4     ".write": true
5   }
6 }
7

```

I creem la base de dades

# Realtime Database

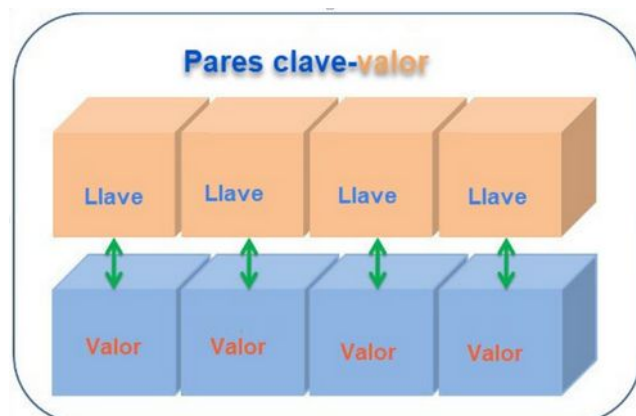
Datos **Reglas** Copias de seguridad Uso

Protege tus recursos de Realtime Database contra los abusos, como fraudes de facturación o suplantación d identidad.

<https://montserratak-76f14-default-rtdb.europe-west1.firebaseio.com>

Ara, a Android Studio, implementarem les dades del nou usuari a la base de dades. Entre les dades que hi posem, hi ha la puntuació, que serà 0

Per a fer-ho utilitzarem Hashmap, una col·lecció de claus-contingut





Exemple de Hashmap: <https://www.geeksforgeeks.org/kotlin-hashmap/>

#### Característiques de Hashmap

- Els valors es poden emmagatzemar en un mapa formant un parell clau-valor. El valor es pot recuperar usant la clau passant-la al mètode correcte.
- Si no hi ha cap element al Mapa, llançarà una 'NoSuchElementException' .
- HashMap només emmagatzema referències d'objectes . Per això, és impossible utilitzar tipus de dades primitives com double o int. Utilitzeu la classe contenidora (com Integer o Double) al seu lloc.

```
//AQUI GUARDA EL CONTINGUT A LA BASE DE DADES
// Utilitza un HashMap

var dadesJugador : HashMap<String,String> = HashMap<String, String>
()
dadesJugador.put ("Uid",uidString)
dadesJugador.put ("Email",correoString)
dadesJugador.put ("Password",passString)
dadesJugador.put ("Nom",nombreString)
dadesJugador.put ("Data",fechaString)
dadesJugador.put ("Puntuacio",puntuacio)
```

També es pot fer servir ANY com a tipus del HashMap

A continuació, afegim un punter a la base de dades i una taula que es dirà DATA BASE JUGADORS

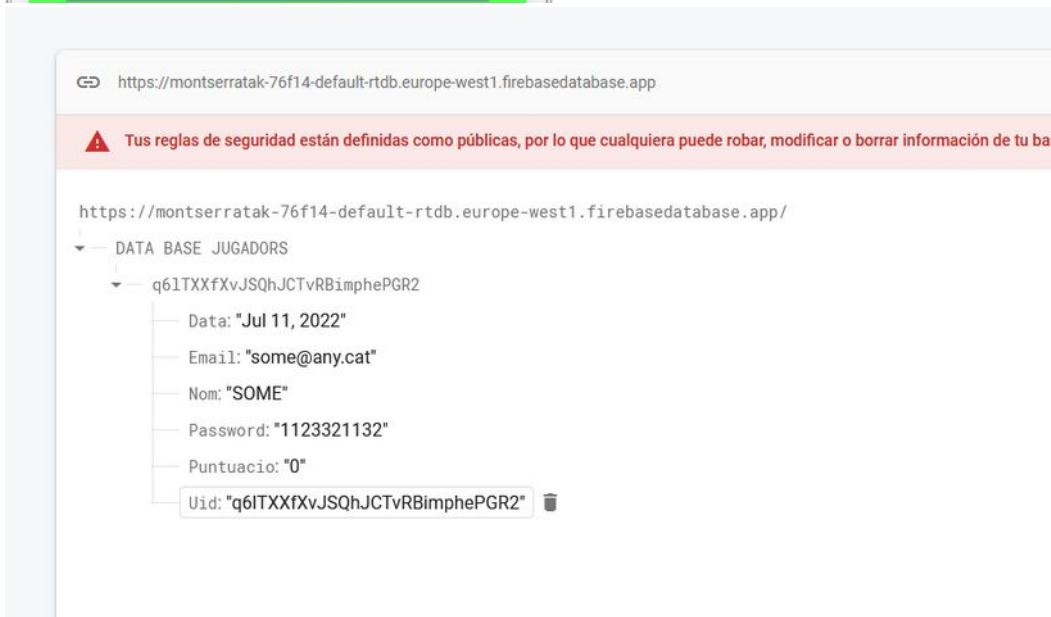
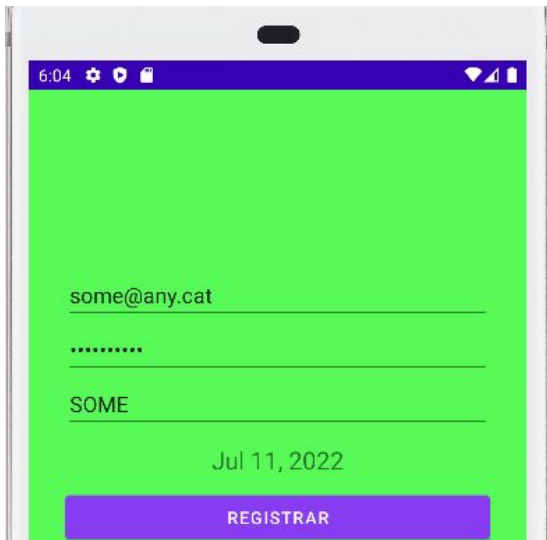
Bàsicament el sistema requereix un punter a la base de dades i fer servir el mètode child

```
// Creem un punter a la base de dades i li donem un nom
var database: FirebaseDatabase =
FirebaseDatabase.getInstance("https://montserratak-76f14-default-
rtadb.europe-west1.firebaseio.com/")
var reference: DatabaseReference = database.getReference("DATA BASE
JUGADORS")

if(reference!=null) {
//crea un fill amb els valors de dadesJugador
reference.child(uidString).setValue(dadesJugador)
Toast.makeText(this, "USUARI BEN REGISTRAT",
Toast.LENGTH_SHORT).show()
}
```

```
}  
else{  
    Toast.makeText(this, "ERROR BD", Toast.LENGTH_SHORT).show()  
}  
finish()
```

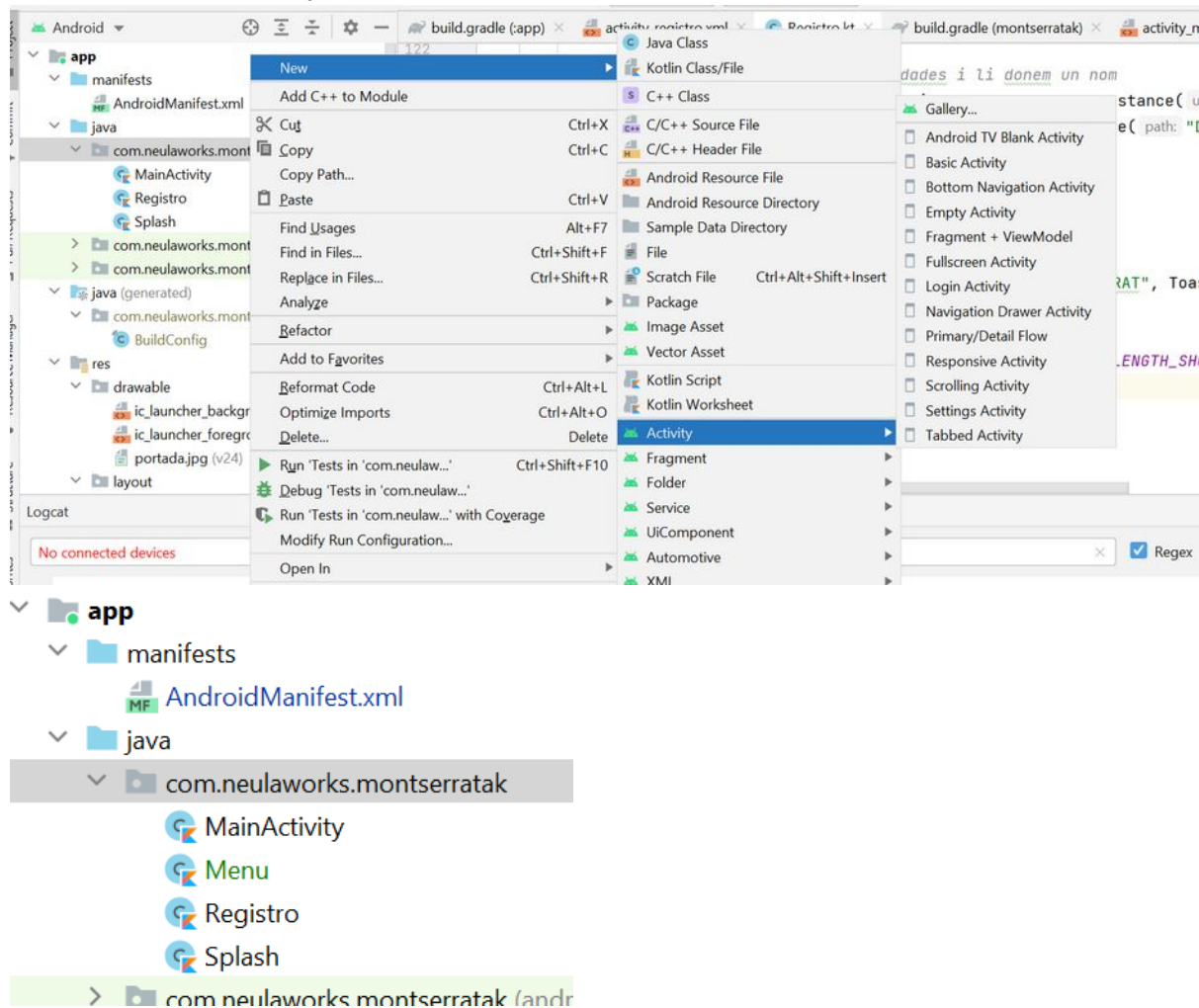
## Provem



Efectivament queda registrat

## Fase 3: Identificador de Jugador

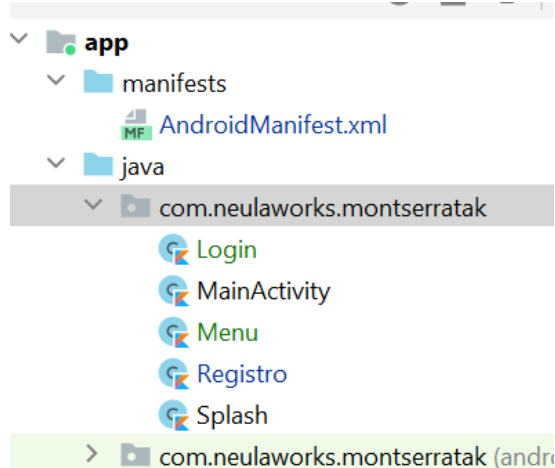
Creem una nova activity que es dirà menu



Ara fem que, després del registre, amb un intent ens envii a la activity menu

```
var reference: DatabaseReference = database.getReference( path: "DATA BASE JUGADORS")
if(reference!=null) {
    Log.i ( tag: "MYTAG",reference.toString())
    Log.i ( tag: "MYTAG", uidString)
    Log.i ( tag: "MYTAG",dadesJugador.toString())
    reference.child(uidString).setValue(dadesJugador)
    Toast.makeText( context: this, text: "USUARI BEN REGISTRAT", Toast.LENGTH_SHORT).show()
    val intent= Intent( packageContext: this, Menu::class.java)
    startActivity(intent)
}
else{
    Toast.makeText( context: this, text: "ERROR BD", Toast.LENGTH_SHORT).show()
}
finish()
```

Creem ara l'activity de Login, com hem creat la de menú

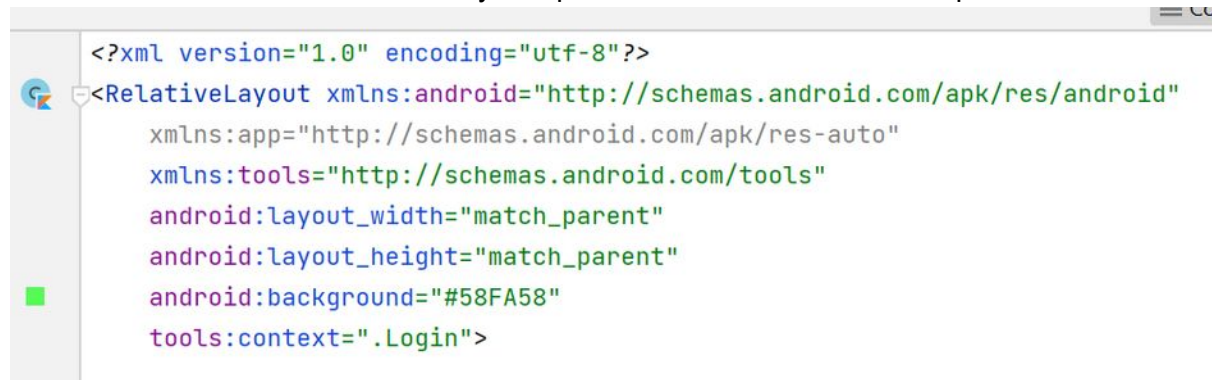


I ara des de la mainactivity fem un intent a Login quan es prem el botó superior

```
BTMLOGIN.setOnClickListener() {  
    val intent= Intent(this, Login::class.java)  
    startActivity(intent)  
}
```

A activity\_login.xml dibuixem el layout, que serà molt semblant al de Registre, amb menys opcions (podem agafar el de Registre i fer un copy paste eliminant i canviant algunes coses)

Primer recordem de fer un relativelayout i posem el color de fons de sempre



I ara fem un copy paste de Registro eliminant la data i el nom

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:gravity="center"  
    android:orientation="vertical"  
    android:padding="30dp">  
  
    <!-- ENTREM EL MAIL -->  
    <EditText  
        android:id="@+id/correoEt"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"
```

```

        android:inputType="textEmailAddress"
        android:hint="@string/correoEt" />

<!-- ENTREM EL PASSWORD -->
<EditText
    android:id="@+id/passEt"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:hint="@string/passEt" />

<!-- BOTO DE ENTRAR -->
<Button
    android:id="@+id/Registrar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/Registrar"
    android:layout_marginTop="10dp"
    />

</LinearLayout>

```

Ara ja tenim el format però cal encara canviar els id i les crides a string  
Primerament anem a strings.xml i creem unes noves referències

```

<string name="correoLogin">E-mail</string>
<string name="passLogin">Password</string>
<string name="BtnLogin">ENTRAR</string>

```

Ara canviant les referències pràcticament ja ho tindriem

```

<!-- ENTREM EL MAIL -->
<EditText
    android:id="@+id/correoLogin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textEmailAddress"
    android:hint="@string/correoLogin" />

<!-- ENTREM EL PASSWORD -->
<EditText
    android:id="@+id/passLogin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:hint="@string/passLogin" />

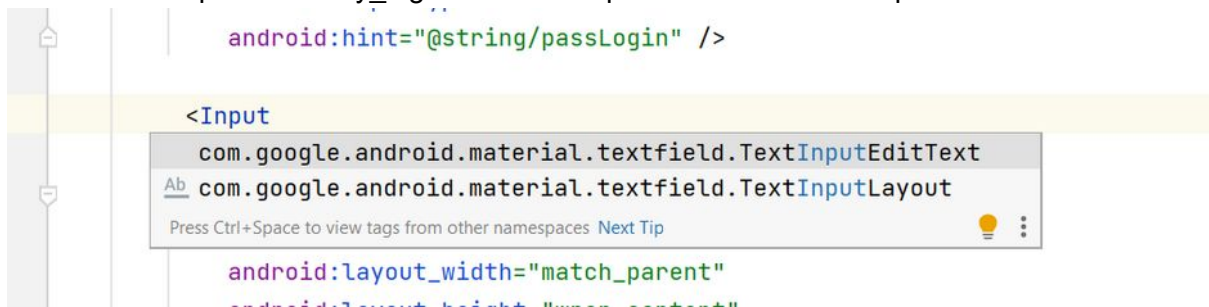
```

```

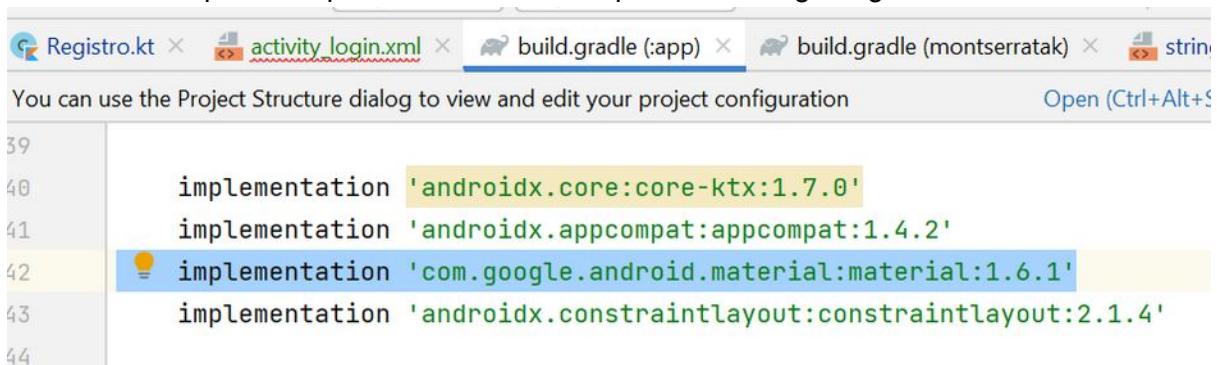
<!-- BOTO DE ENTRAR -->
<Button
    android:id="@+id/BtnLogin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/BtnLogin"
    android:layout_marginTop="10dp"
/>

```

Però anem a afegir un efecte de format que ens dona la llibreria de firebase  
Si escrivim <Input a activity\_login.xml veiem que ens mostra dues opcions



Són formats importats a partir de les llibreries que hem carregat a gradle



Per a utilitzar-los seleccionem TextInputLayout i col·loquem els editText a dins d'aquesta manera:

```

<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <!-- ENTREM EL MAIL -->
    <EditText
        android:id="@+id/correoLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress"
        android:hint="@string/correoLogin" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

```

```

<!-- ENTREM EL PASSWORD -->
<EditText
    android:id="@+id/passLogin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:hint="@string/passLogin" />
</com.google.android.material.textfield.TextInputLayout>

```

Afegim també un marge al password perquè no es vegin tant junts (el password i el mail) i de pasada col·locarem un Toggle que fa que el password no es vegi quan es pica

```

<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:passwordToggleEnabled="true">
    <!-- ENTREM EL PASSWORD -->
    <EditText
        android:id="@+id/passLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:inputType="textPassword"
        android:hint="@string/passLogin" />
</com.google.android.material.textfield.TextInputLayout>

```

Amb això aconseguirem un efecte xulo cada vegada que l'usuari entri el correu i el password



Les etiquetes E-mail i Password es fan petites quan fem clic sobre les finestres, i el ull de password és el Toggle que hem afegit al format de google.

Anem ara a donar funcionalitat al botó d'entrar

Com a Registre, despleguem les variables i les assignem a elements de R

```

class Login : AppCompatActivity() {

```



```

//Despleguem les variables que farem servir
lateinit var correoLogin : EditText
lateinit var passLogin : EditText
lateinit var BtnLogin : Button

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_login)

    // Busquem a R els elements als que apunten les variables
    correoLogin =findViewById<EditText>(R.id.correoLogin)
    passLogin =findViewById<EditText>(R.id.passLogin)
    BtnLogin =findViewById<Button>(R.id.BtnLogin)

```

Donem funcionalitat al botó:

Fem un còpia i enganxa del botó de Registre i canviem les variables per les que fem servir aquí

```

BtnLogin.setOnClickListener() {
    //Abans de fer el registre validem les dades
    var email:String = correoLogin.getText().toString()
    var passw:String = passLogin.getText().toString()

    // validació del correu
    // si no es de tipus correu
    if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
        correoLogin.setError("Invalid Mail")
    }
    else if (passw.length<6) {
        passLogin.setError("Password less than 6 chars")
    }
    else
    {
        // aquí farem LOGIN al jugador
    }
}

```

Abans de fer el procediment, crearem la variable de firebase

```
lateinit var auth: FirebaseAuth //FIREBASE AUTENTIFICACIO
```

i a dins d'OnCreate

```

//Instanciem el firebaseAuth
auth = FirebaseAuth.getInstance()

```



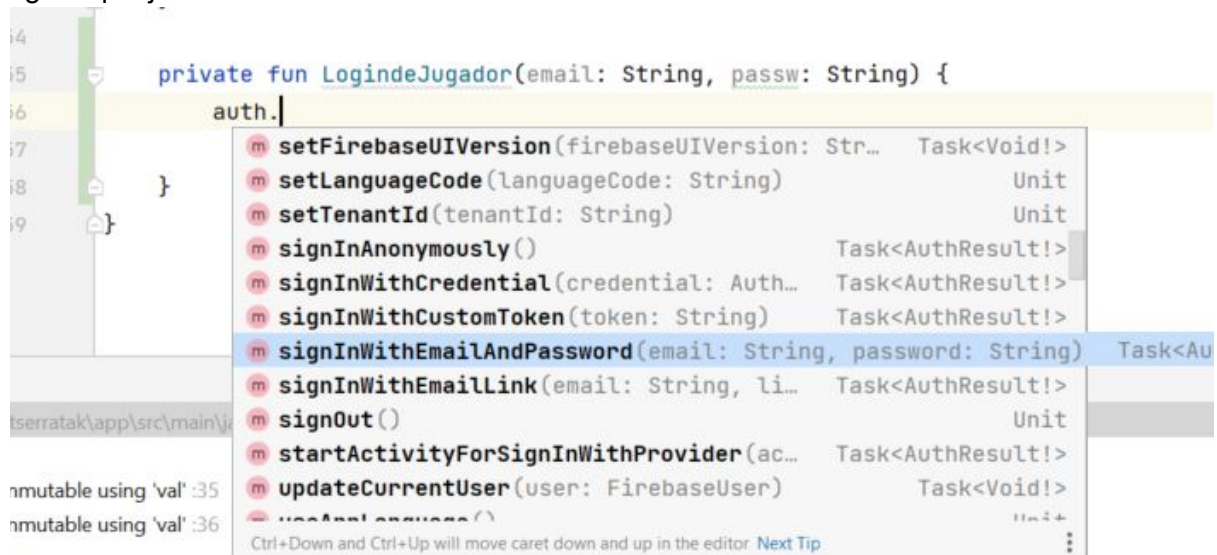
Ara ja podem fer una funció que faci Login al Jugador

```
// aquí farem LOGIN al jugador  
LogindeJugador(email, passw)
```

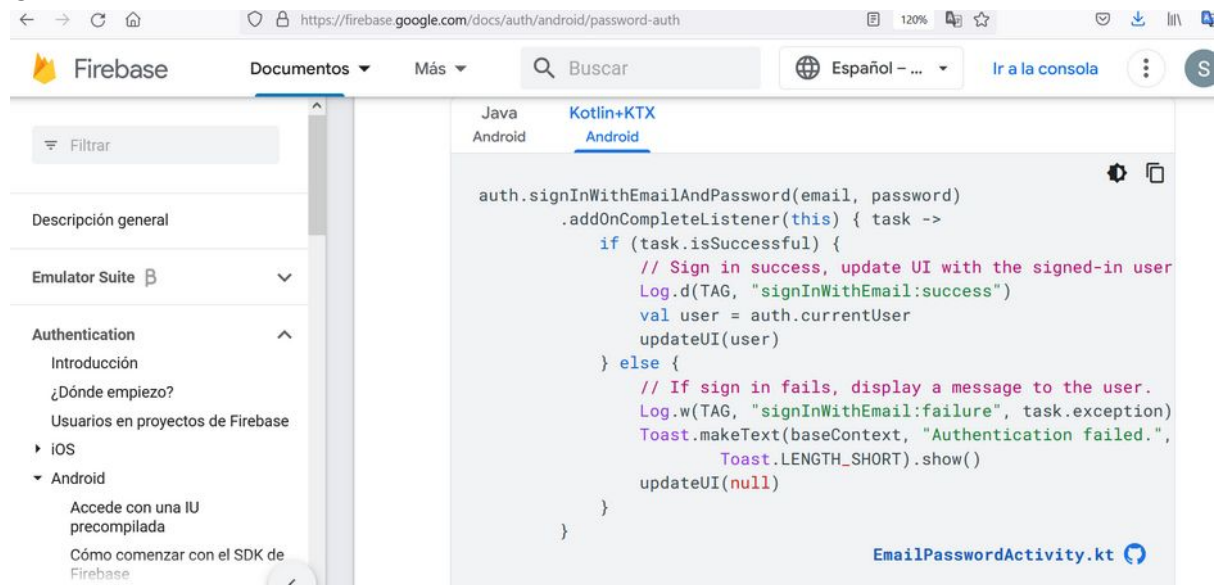
i creem el mètode

```
private fun LogindeJugador(email: String, passw: String) {  
  
}
```

a dins cridem a auth. i ens surten una sèrie de mètodes, agafarem signin... perquè l'usuari figura que ja està creat



Com diu la documentació de firebase



Queda el Login així

```
private fun LogindeJugador(email: String, passw: String) {
    auth.signInWithEmailAndPassword(email, passw)
        .addOnCompleteListener(this)
        { task ->
            if (task.isSuccessful) {
                val tx: String = "Benvingut "+ email
                Toast.makeText(this, tx, Toast.LENGTH_LONG).show()
                val user = auth.currentUser
                updateUI(user)
            } else {
                Toast.makeText(this, "ERROR Autenticació",
                    Toast.LENGTH_LONG).show()
            }
        }
}

fun updateUI(user:FirebaseUser?)
{
    val intent= Intent(this, Menu::class.java)
    startActivity(intent)
    finish()
}
```

En aquest cas al mètode updateUI no caldria passar el user (com vam fer a la activity registre), només canviem d'activity

## Fase 4: Menú

Seguim amb el view del menú. Primerament haurem de comprovar les identifikacions, per a aixó creem dues variables i busquem el usuari en cridar a onCreate

```
class Menu : AppCompatActivity() {
    //creem unes variables per comprovar usuari i autenticació
    lateinit var auth: FirebaseAuth
    var user:FirebaseUser? = null;

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_menu)

        auth= FirebaseAuth.getInstance()
        user =auth.currentUser

    }
}
```

Ara farem un procediment tal que si el usuari esta logejat, tira un toast i continua, però si no ho està t'envia a la mainActivity, on pots anar a crear usuari o logejar-te

```
private fun Usuarilogejat()
{
    if (user !=null)
    {
        Toast.makeText(this,"Jugador logejat",
Toast.LENGTH_SHORT).show()
    }
    else
    {
        val intent= Intent(this, MainActivity::class.java)
        startActivity(intent)
        finish()
    }
}
```

Ara reescriurem el mètode onStart, recordem que: onCreate () es diu quan es crea l'activitat per primera vegada i onStart () s'anomena quan l'activitat es torna visible per a l'usuari.

```
// Aquest mètode s'executarà quan s'obri el minijoc
override fun onStart() {
    usuariLogejat()
    super.onStart()
}
}
```

Ara crearem al menú un botó per tancar la sessió

Anem a `activity_menu` i el configurarem similar al de Registre, en aquest cas i per a començar només col·loquem un botó, el de tancar la sessió

A strings creem una nova variable

```
<string name="TancarSessio">Tancar Sessió</string>
```

I l'arxiu `activity_menu` queda així

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#58FA58"
    tools:context=".Menu">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical"
        android:padding="30dp">

        <!-- BOTO DE ENTRAR -->
        <Button
            android:id="@+id/tancarSessio"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/tancarSessio"
            android:layout_marginTop="10dp"
            />

    </LinearLayout>
```

```
</RelativeLayout>
```

Ara modifiquem el codi de menu.kt, amb una nova variable

```
lateinit var tancarSessio: Button
```

Que apunta al botó:

```
tancarSessio =findViewById<Button>(R.id.tancarSessio)
```

a onCreate, seguit d'un listener

```
tancarSessio.setOnClickListener() {  
    tancalaSessio()  
}
```

I la funció de tancar que únicament crida al mètode signOut i després passa a la pantalla principal

```
private fun tancalaSessio() {  
    auth.signOut() //tanca la sessió  
    //va a la pantalla inicial  
    val intent= Intent(this, MainActivity::class.java)  
    startActivity(intent)  
    finish()  
}
```

Ara afegirem al mainActivity que comprovi si la sessió està inicialitzada o no, si ho està directament passa al menú (copiant el mateix sistema que hem fet a la finestra de menu, però en aquest cas si està logejat l'enviem directament al menú, perquè no cal que torni a identificar-se)

```
class MainActivity : AppCompatActivity() {  
  
    //per a comprovar si la sessió esta inicialitzada  
    lateinit var auth: FirebaseAuth  
    var user: FirebaseUser? = null;  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        //assigna valor a user  
        auth = FirebaseAuth.getInstance()  
        user = auth.currentUser  
    }  
}
```

```

var BTMLOGIN = findViewById<Button>(R.id.BTMLOGIN);
var BTMREGISTRO = findViewById<Button>(R.id.BTMREGISTRO);

BTMLOGIN.setOnClickListener() {
    val intent = Intent(this, Login::class.java)
    startActivity(intent)
}

BTMREGISTRO.setOnClickListener() {
    //Toast.makeText(this, "click botó
Registre", Toast.LENGTH_LONG).show();
    val intent = Intent(this, Registro::class.java)
    startActivity(intent)
}
}

// Aquest mètode s'executarà quan s'obri el menu
override fun onStart() {
    usuariLogejat()
    super.onStart()
}

private fun usuariLogejat() {

    if (user != null)
    {
        val intent= Intent(this, Menu::class.java)
        startActivity(intent)
        finish()
    }

}
}

```

Seguim amb la implementació del menú. Aquest ha de tenir:

Una imatge amb la puntuació, el nom, el mail i 4 botons: JUGAR, PUNTUACIONS, CREDITS, SORTIR

Anem a activity\_menu.xml

Canviarem primerament el RelativeLayout per un ScrollView, això ens permetrà adaptar la pantalla per a mòbils amb pantalles més petites

Primerament però anem a strings.xml i afegim unes variables

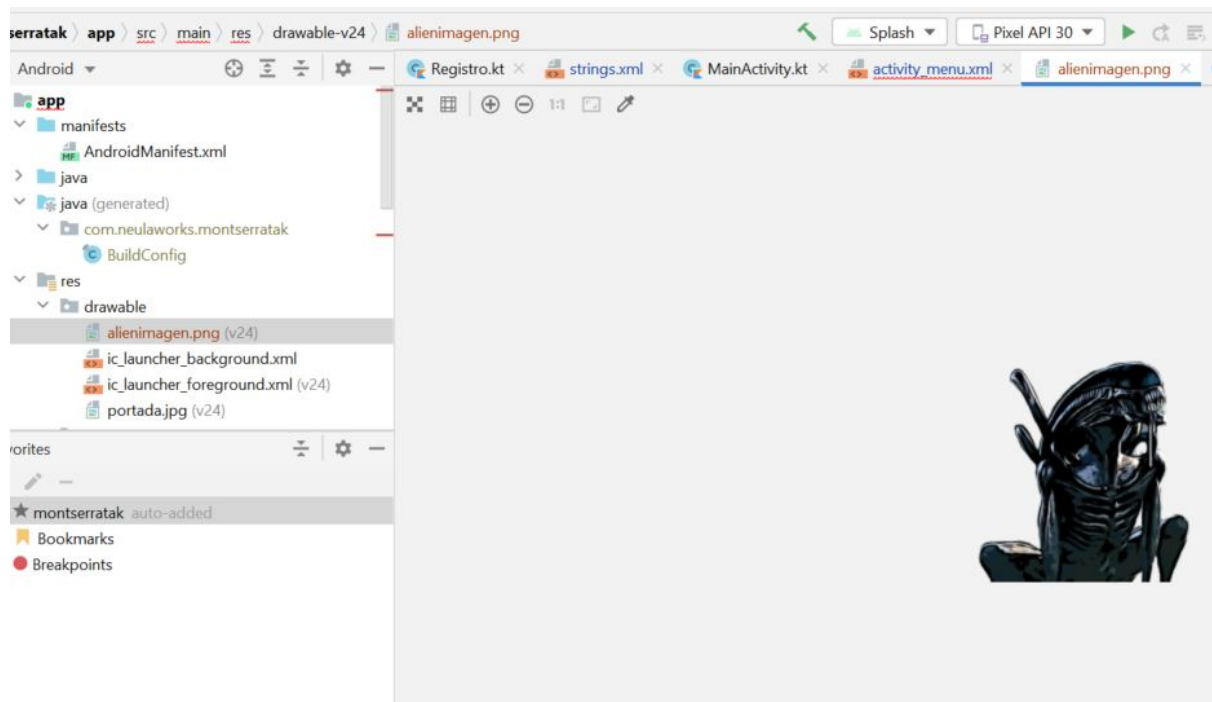
```

<string name="tancarSessio">Tancar Sessió</string>

<string name="miPuntuaciotxt">Puntuació</string>
<string name="puntuacio"> -- </string>
<string name="uid"> -- </string>
<string name="correo"> -- </string>
<string name="nom"> -- </string>
<string name="MenuTxt">MENÚ</string>
<string name="jugarBtn">Jugar</string>
<string name="PuntuacionsBtn">Puntuacions</string>
<string name="CreditsBtn">Crèdits</string>

```

També ens fa falta una imatge de 180x180, la copiem a drawable així: (ha de tenir només minúscules)



Ara ja podem anar a crear els botons i els textos al activity menu

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#58FA58"
    android:layout_gravity="center"
    tools:context=".Menu">

```

```

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="30dp">

    <!-- Text La meva puntuació -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/miPuntuaciotxt"
        android:text="@string/miPuntuaciotxt"
        android:textSize="30sp"
        android:gravity="center"/>

    <!-- contingut puntuació -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/puntuacio"
        android:text="@string/puntuacio"
        android:gravity="center"/>

    <!-- Imatge alien -->
    <androidx.appcompat.widget.AppCompatImageView
        android:layout_width="180dp"
        android:layout_height="180dp"
        android:id="@+id/alienimagen"
        android:src="@drawable/alienimagen" />

    <!-- uid -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/uid"
        android:text="@string/uid"
        android:gravity="center"/>

    <!-- correu -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/correo"
        android:text="@string/correo"
        android:gravity="center"/>

```



```

<!-- nom -->
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/nom"
    android:text="@string/nom"
    android:gravity="center"/>

<!-- BOTO DE jugar -->
<Button
    android:id="@+id/jugarBtn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/jugarBtn"
    android:layout_marginTop="10dp"
    />

<!-- BOTO DE puntuacions -->
<Button
    android:id="@+id/PuntuacionsBtn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/PuntuacionsBtn"
    android:layout_marginTop="10dp"
    />

<!-- BOTO DE Credits -->
<Button
    android:id="@+id/CreditsBtn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/CreditsBtn"
    android:layout_marginTop="10dp"
    />

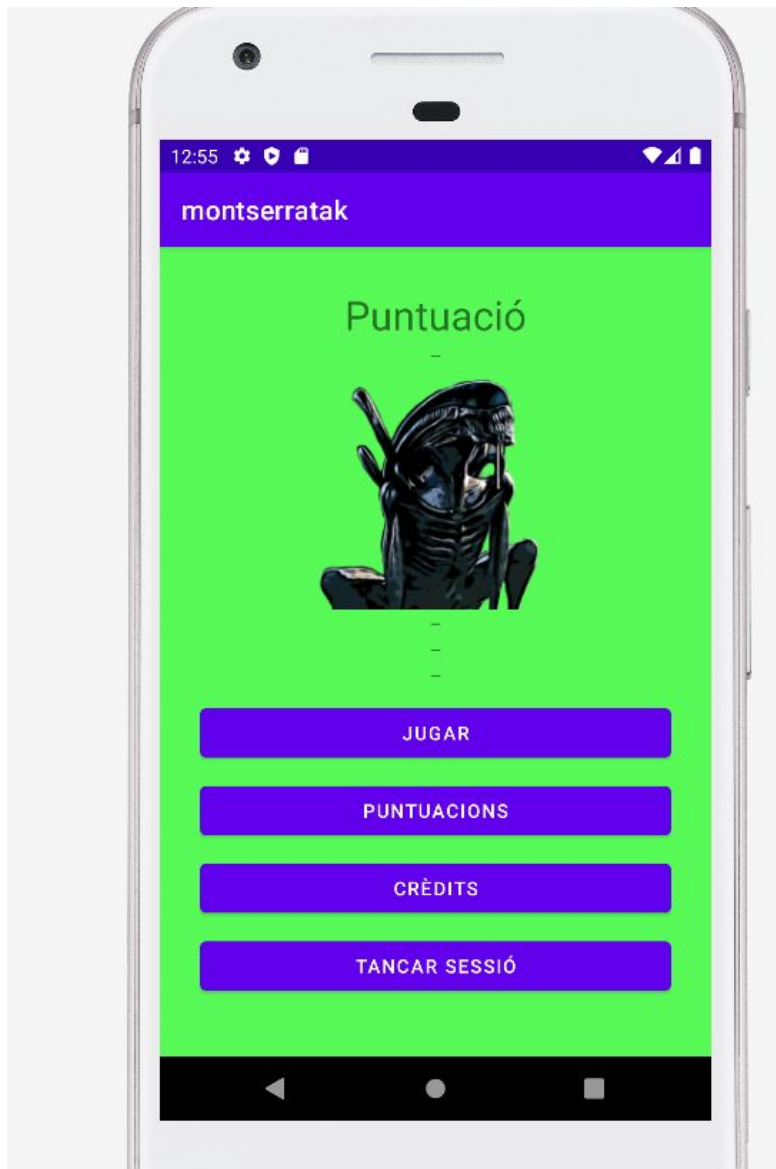
<!-- BOTO DE Tancar sessió -->
<Button
    android:id="@+id/tancarSessio"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/tancarSessio"
    android:layout_marginTop="10dp"
    />

</LinearLayout>

```

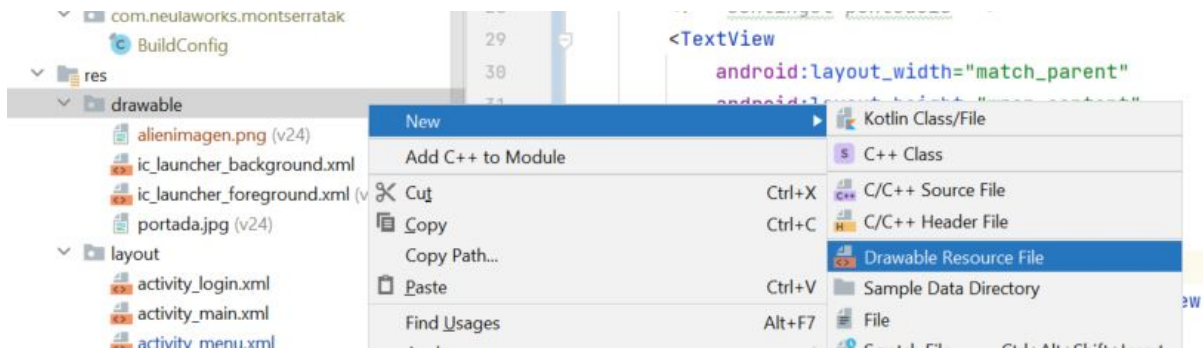
```
</ScrollView>
```

I el resultat és una cosa així

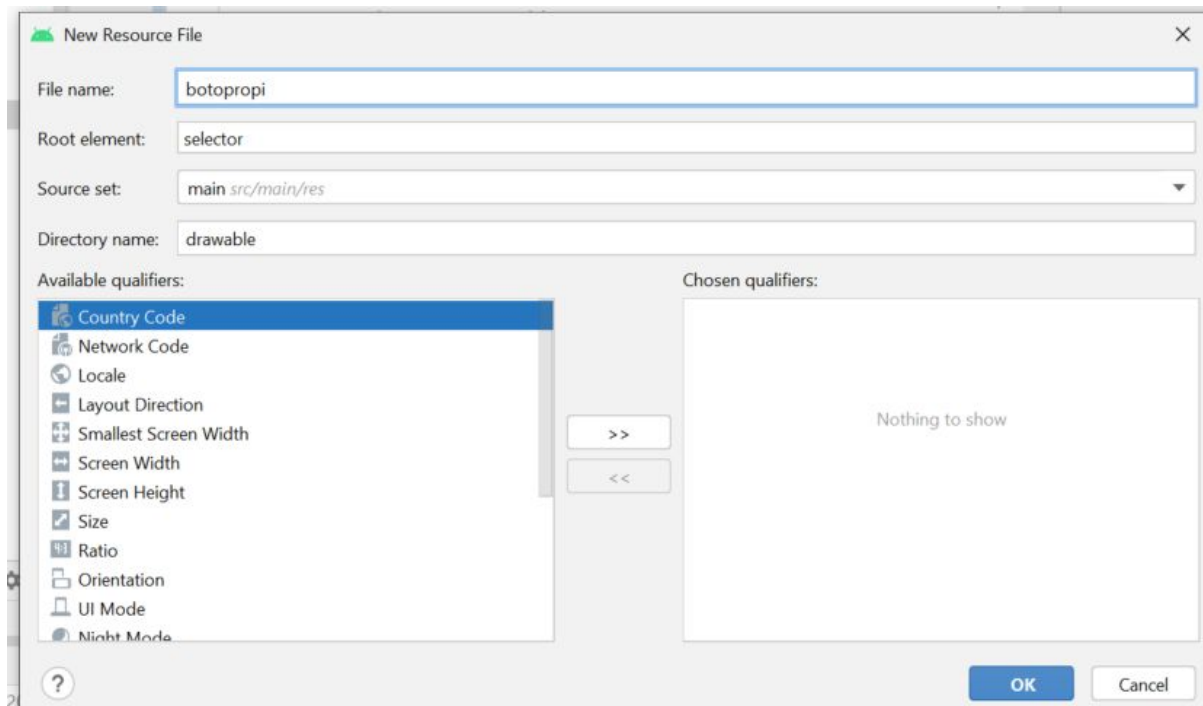


Els botons no ens agraden gaire, volem fer uns **botons personalitzats**, com es fa això?

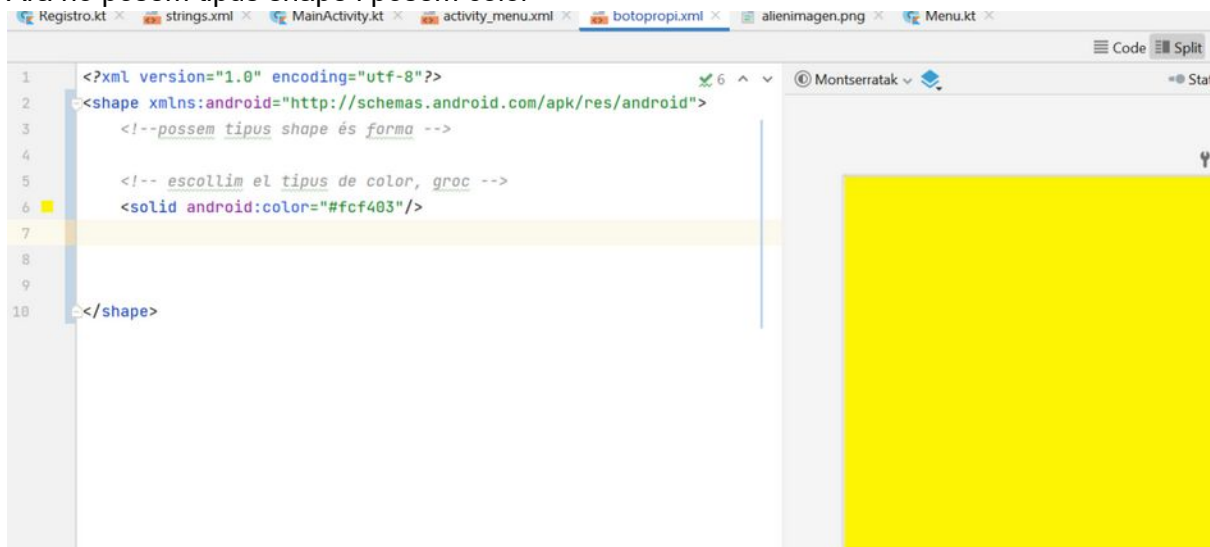
Anem a



creem un nou recurs drawable



Ara ho posem tipus shape i posem color



també posarem un radi

```
<!-- radi -->
<corners android:radius="50dp" />
```

Ara a la vista menú, al xml, cridarem al botó amb background dins de cada button

```
android:background="@drawable/botopropi"  
android:textColor="@color/black"
```

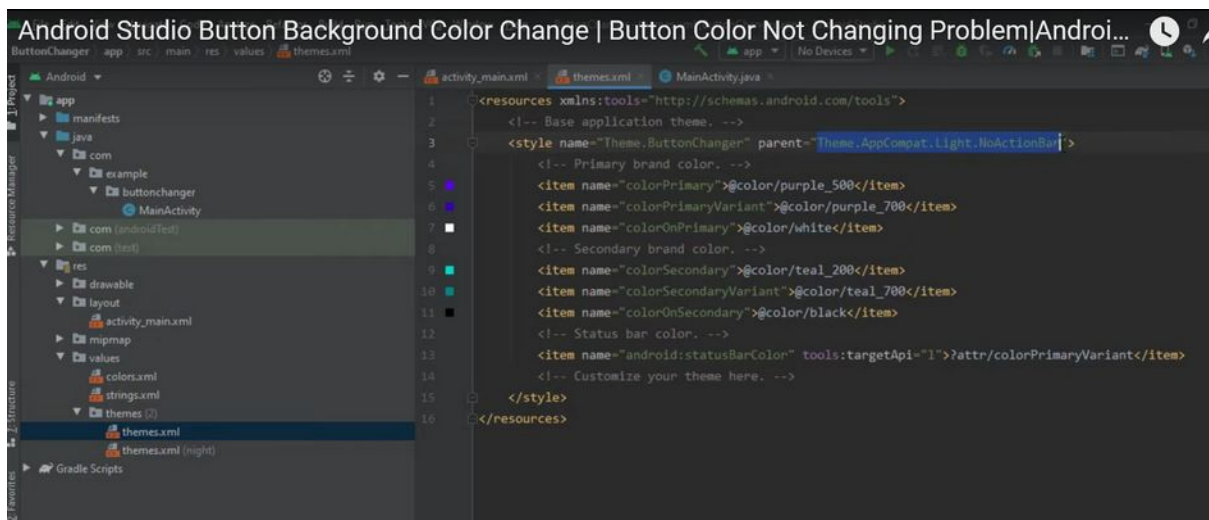
i la lletra en color negre

Però no canvia!!!!

Això és perquè per defecte tenim uns temes aplicats (nit/dia) que defineixen el color dels botons.

En aquest vídeo explica com canviar-ho a partir dels themes

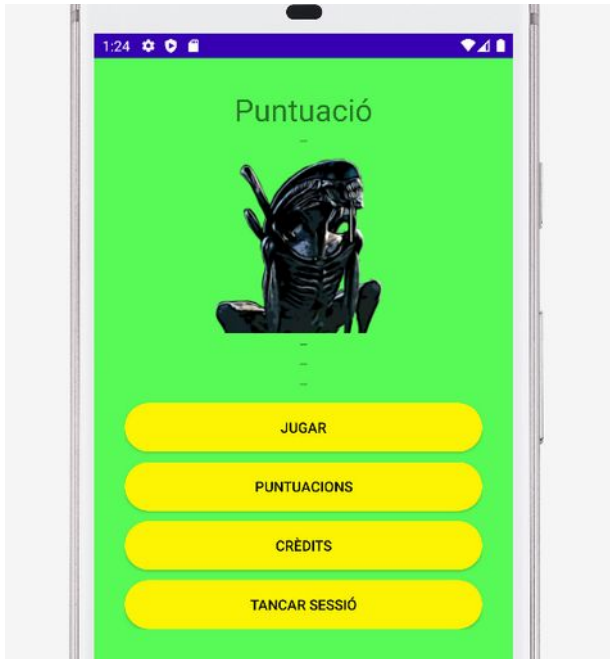
<https://www.youtube.com/watch?v=AdlxldYus14>



Canviarem el theme per un altre, per exemple Light NoactionBar

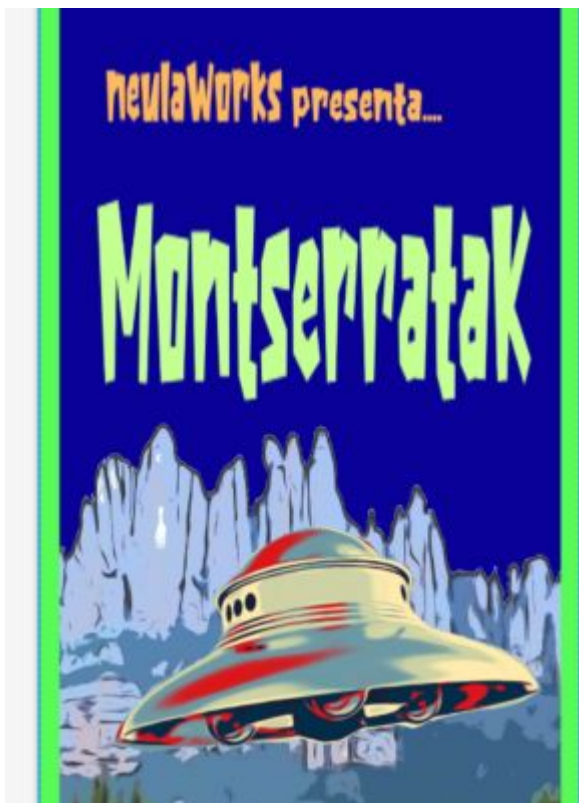
```
<style name="Theme.Montserratak"  
parent="Theme.AppCompat.Light.NoActionBar">
```

Ara sí:



Aprofitem i canviem tots els botons de l'aplicació (només afegint les dues línies al xml  
`android:background="@drawable/botopropi"`  
`android:textColor="@color/black"`

També canviem la pantalla d'splash incorporant un ovni, així es veurà més xula i s'entén millor de què va el joc.



Seguim amb el menú, ara implementarem els botons a menu.kt

Creem les variables...

```
lateinit var tancarSessio: Button
lateinit var CreditsBtn: Button
lateinit var PuntuacionsBtn: Button
lateinit var jugarBtn: Button
```

//les assignem a onCreate

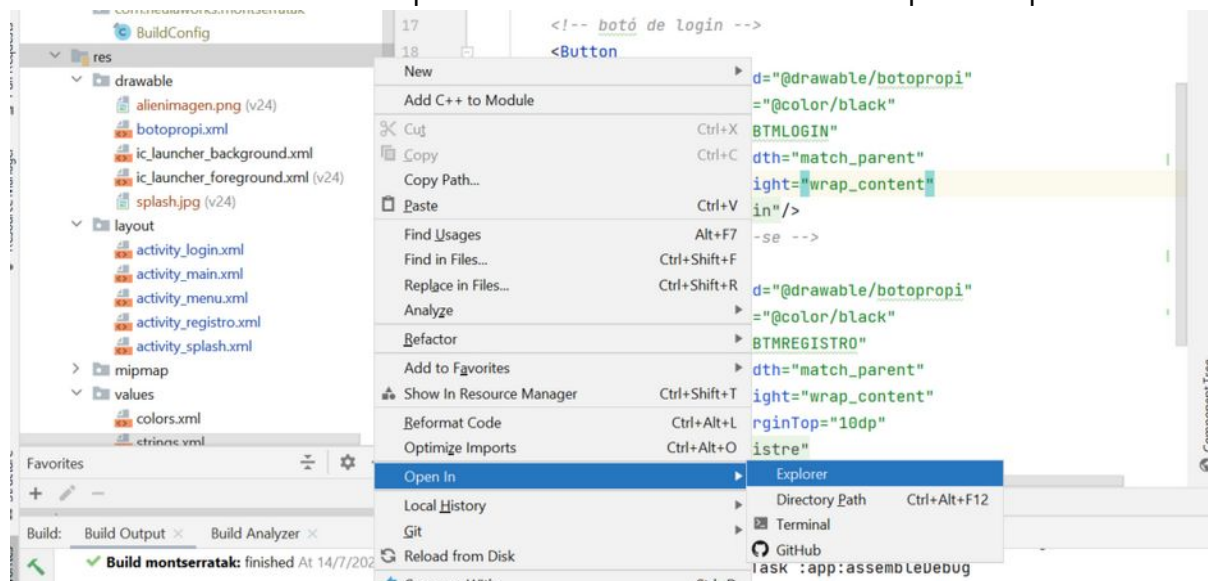
```
tancarSessio =findViewById<Button>(R.id.tancarSessio)
CreditsBtn =findViewById<Button>(R.id.CreditsBtn)
PuntuacionsBtn =findViewById<Button>(R.id.PuntuacionsBtn)
jugarBtn =findViewById<Button>(R.id.jugarBtn)
```

I els listeners

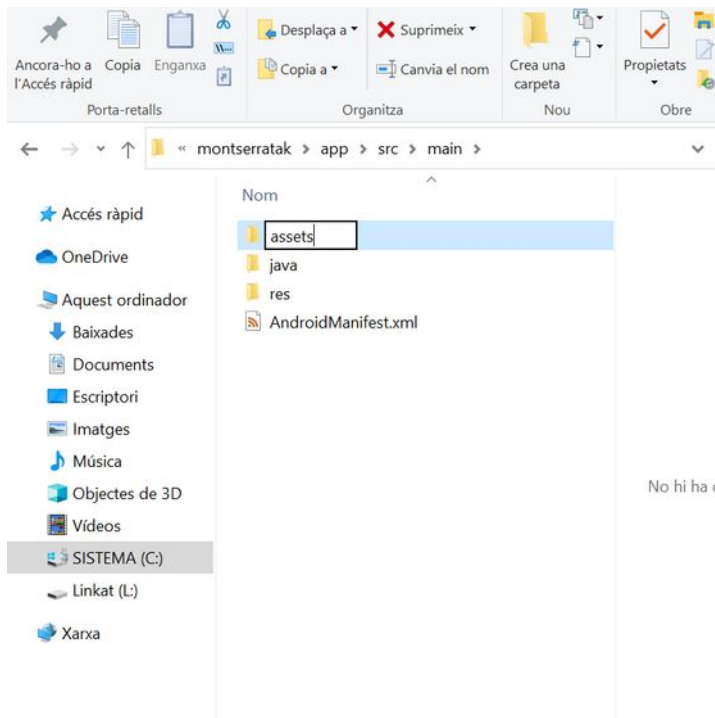
```
CreditsBtn.setOnClickListener() {
    Toast.makeText(this, "Credits", Toast.LENGTH_SHORT).show()
}
PuntuacionsBtn.setOnClickListener() {
    Toast.makeText(this, "Puntuacions", Toast.LENGTH_SHORT).show()
}
jugarBtn.setOnClickListener() {
    Toast.makeText(this, "JUGAR", Toast.LENGTH_SHORT).show()
}
```

### Canvi de tipus de lletra a partir d'un arxiu

Ara canviarem el tipus de lletra utilitzat, per a això crearem primerament una carpeta nova anomenada assets. Per fer-ho primerament anem a res i obrim la carpeta a explorar

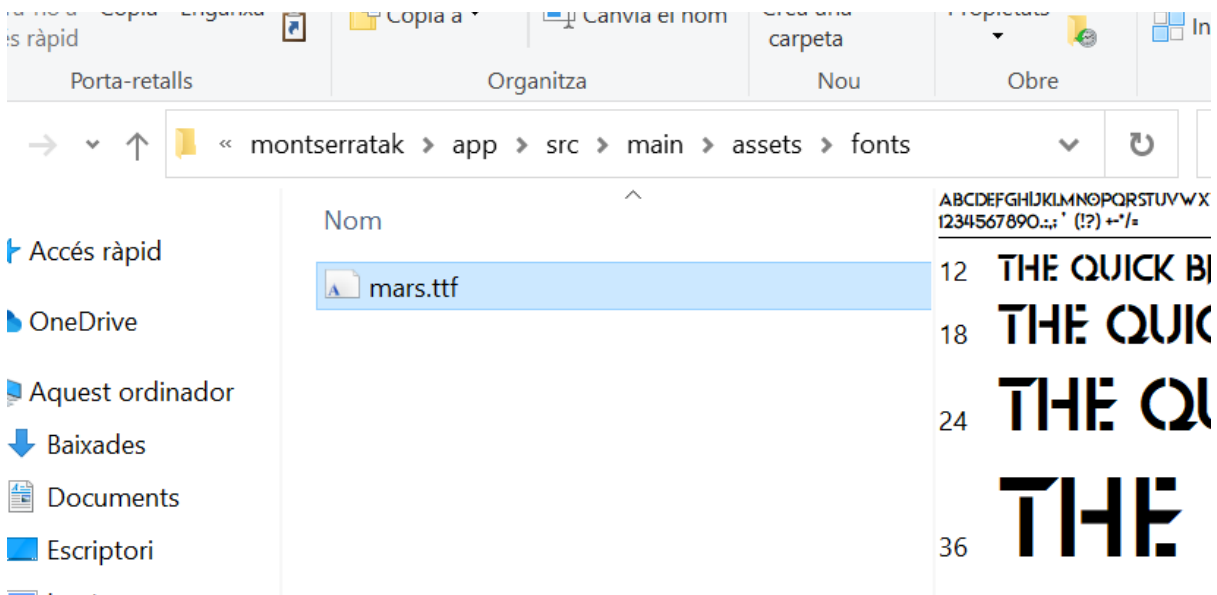


creem una carpeta anomenada assets

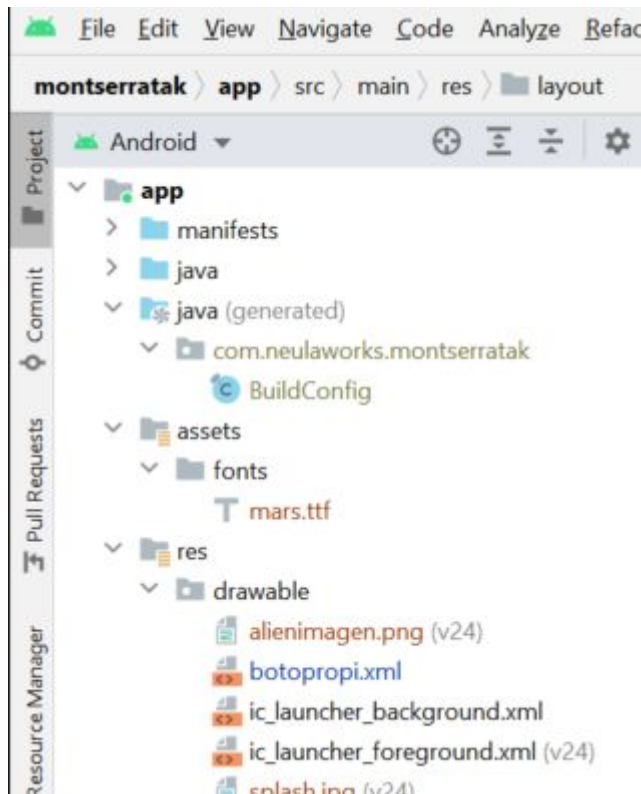


I a dins d'aquesta una altra carpeta que es digui fonts, on guardem la font que volem fer servir

**ALERTA!!! La font ha d'incorporar accents** (i si fos el cas també els símbols que fem servir per als textos de l'aplicació)



I ja surt al projecte



Ara és tan fàcil com anara a onCreate del menú i crear un typeface

```
//Aquí creem un tipus de lletra a partir de una font
val tf = Typeface.createFromAsset(assets, "fonts/mars.ttf")
```

buscar els elements de la pantalla que són text i assignar el typeface (la font utilitzada)  
1er definim variables

```
lateinit var miPuntuaciotxt: TextView
lateinit var puntuacio: TextView
lateinit var uid: TextView
lateinit var correo: TextView
lateinit var nom: TextView
```

a onCreate les assignem i donem format al text mostrat

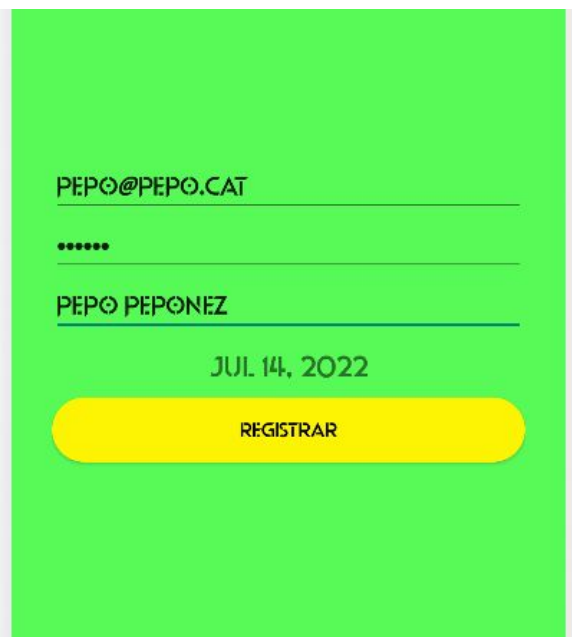
```
//busquem els textos
miPuntuaciotxt=findViewById(R.id.miPuntuaciotxt)
puntuacio=findViewById(R.id.puntuacio)
uid=findViewById(R.id.uid)
correo=findViewById(R.id.correo)
nom=findViewById(R.id.nom)

//els hi assignem el tipus de lletra
miPuntuaciotxt.setTypeface(tf)
puntuacio.setTypeface(tf)
uid.setTypeface(tf)
correo.setTypeface(tf)
nom.setTypeface(tf)
```



```
//fem el mateix amb el text dels botons
tancarSessio.setTypeface(tf)
CreditsBtn.setTypeface(tf)
PuntuacionsBtn.setTypeface(tf)
jugarBtn.setTypeface(tf)
```

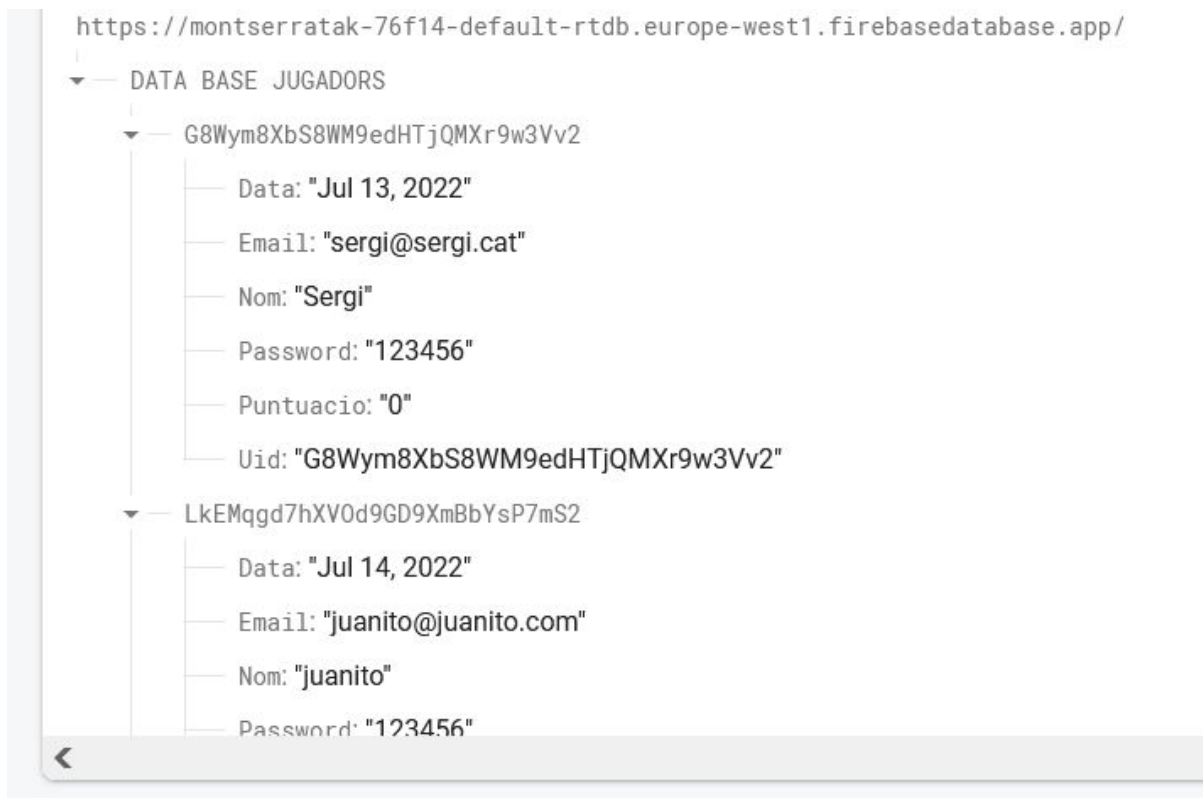
De la mateixa manera, canviem la lletra a MainActivity, Menu, i registre.  
I es veu així:



Ara al menú del jugador, volem que es connecti a firebase i ens mostri les dades del jugador a la pantalla.

Crearem un mètode per a fer una consulta que farà un recorregut a través de tots els jugadors fins que trobi el identificador del jugador que està jugant actualment

Primerament anem a firebase i mirem com vam anomenar els camps de la base de dades, que ens fara falta



Perfecte, ara creem primerament una variable a la classe Menu.tk, per a referenciar la base de dades

```
//reference serà el punter que ens envia a la base de dades de jugadors
lateinit var reference: DatabaseReference
```

Fem una funció anomenada consulta que és cridada des de onCreate

Aquesta funció busca a la base de dades les dades de l'usuari i les mostre per pantalla. Per fer-ho fa una recerca a tots els child, un per un, va comparant el mail amb el mail de l'usuari i quan el troba mostra les dades.

Aquesta iteració a la pràctica seria molt lenta amb moltes dades, s'ha optat per fer-ho així per entendre millor com estan distribuïdes les dades en aquesta base de dades.

És més correcte utilitzar un Query amb un valor de recerca.

```
private fun consulta() {
    //Amb Firebase no fem consultes realment, el que fem en connectar-nos a una referencia
    // i aquesta anirà canviant automàticament quan canviï la base de dades
    // reference apunta a "DATA BASE JUGADORS"
    // sempre es crea una referencia a un punt del arbre de la base de dades
    // Per exemple si tenim la base de dades
    // arrel
    // - nivell dos
    //     - nivell dos.1
```

```

//          - nom: "pepe"
//          - dni: "1231212121"
//      - nivell dos.2: "34"
//      - nivell dos.3: "36"
//var bdasereference:DatabaseReference =
FirebaseDatabase.getInstance().getReference()
//      .child("nivell dos")
//      .child("nivell dos.1")
// Ara estariem al novell dos.1 del arbre
// Ens podem subscriure amb un listener que té dos mètodes
//      onDataChange (es crida si s'actualitza les dades o és la
primera vegada que ens suscribim)
//      onCancelled Es crida si hi ha un error o es cancel·la la
lectura
// al primer mètode rebrem un objecte json que és la branca sobre
la que demanem actualització
// getKey retorna la clave del objecte
// getValue retorna el valor
// els subnodes (fills) es recuperen amb getChildren
//          es poden llegir com un llistat d'objectes
Dataspshots
//          o navegar a subnodes concrets amb
child("nomdelsubnode")

```

```

var database: FirebaseDatabase =
FirebaseDatabase.getInstance("https://montserratak-76f14-default-
rttdb.europe-west1.firebaseio.com/")
var bdreference:DatabaseReference = database.getReference("DATA
BASE JUGADORS")
bdreference.addValueEventListener (object: ValueEventListener {
    override fun onDataChange(snapshot: dataSnapshot) {

        Log.i ("DEBUG","arrel value"+
snapshot.getValue().toString())
        Log.i ("DEBUG","arrel key"+ snapshot.key.toString())
        // ara capturem tots els fills
        var trobat:Boolean =false
        for (ds in snapshot.getChildren()) {
            Log.i ("DEBUG","DS key:
"+ds.child("Uid").key.toString())
            Log.i ("DEBUG","DS value:
"+ds.child("Uid").getValue().toString())
            Log.i ("DEBUG","DS data:
"+ds.child("Data").getValue().toString())
            Log.i ("DEBUG","DS mail:
"+ds.child("Email").getValue().toString())
            //mirem si el mail és el mateix que el del

```

```

jugador
//si és així, mostrem les dades als textview
corresponents
    if
(ds.child("Email").getValue().toString().equals(user?.email)) {
        trobat=true
        //carrega els textview

                puntuacio.setText(
ds.child("Puntuacio").getValue().toString())
                uid.setText(
ds.child("Uid").getValue().toString())
                correo.setText(
ds.child("Email").getValue().toString())
                nom.setText(
ds.child("Nom").getValue().toString())

    }
    if (!trobat)
    {
        Log.e ("ERROR","ERROR NO TROBAT MAIL")
    }

}

}

override fun onCancelled(error: DatabaseError) {
    Log.e ("ERROR","ERROR DATABASE CANCEL")
}

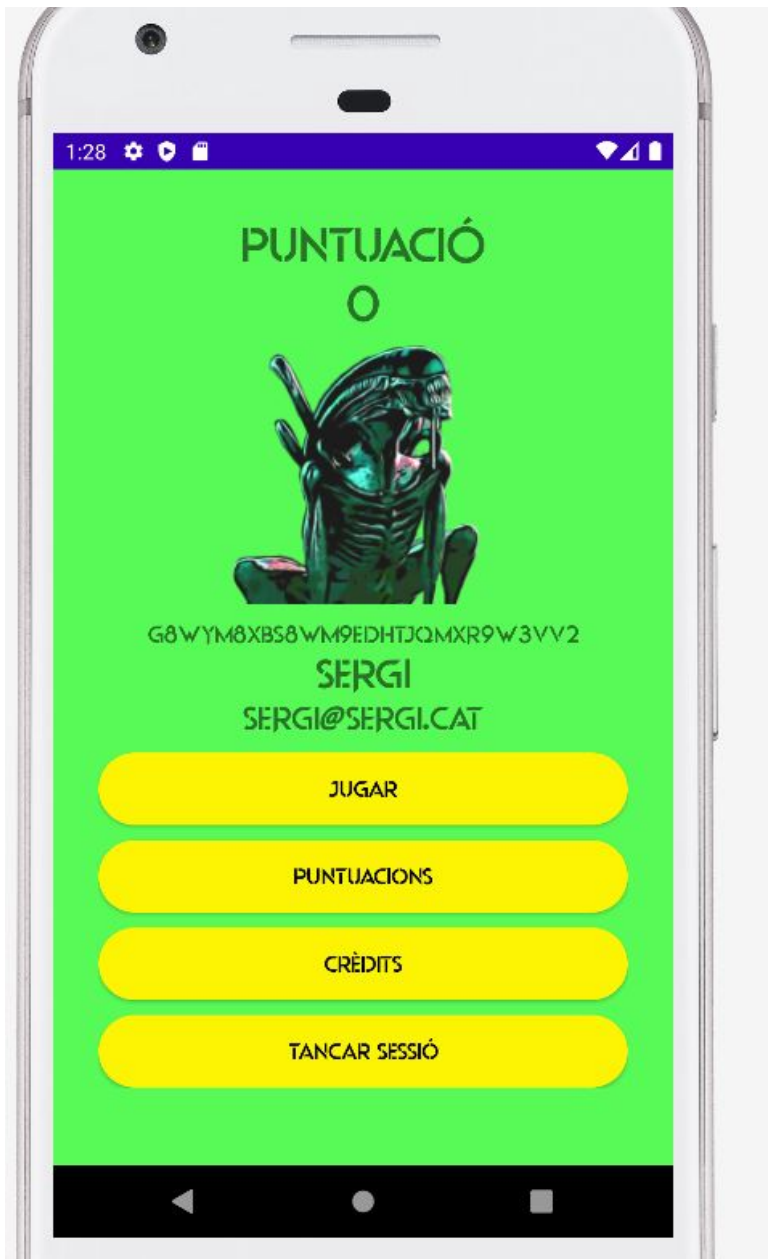
})

}

```

Aprofitem amb la nova lletra per fer uns quants ajustos i que quedi millor la pantalla del menú (bàsicament afegim als textview indicacions de textSize i algun layout\_marginTop per a separar els textos)

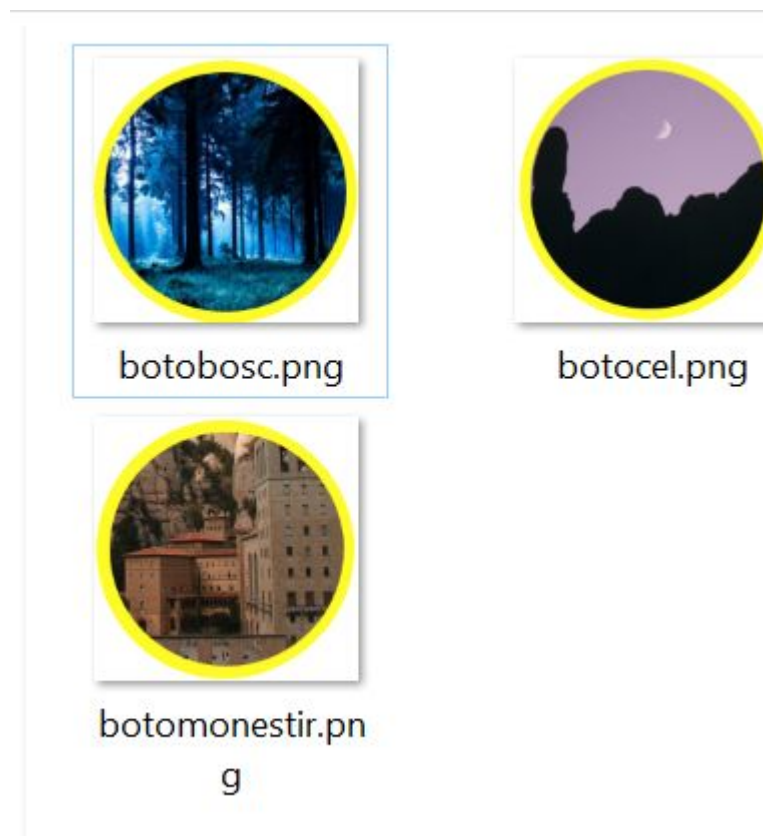
I queda així de xulo



## Fase 5: Pantalla de selecció de nivell

En aquesta fase veurem com es dona funcionalitat a ImageButton, i com es passen dades d'una activity a una altra.

Primerament afegim 3 imatges de 200x200 una per cada nivell amb una part que és transparent



Ara afegim una imatge format 16:9 de fondo de montserrat per a la mateixa pantalla

Ja podem crear una nova activity que es diu seleccionivell

A l'arxiu XML, posarem de fons la imatge del fondo de montserrat

```
<ImageView
    android:id="@+id/fondomontse"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/fondomontse" />
```

Fem constraint a les 4 cares

Ara afegirem els botons, anem a fer servir ImageButton

afegim els imagebuttons, hem de recordar de incloure background transparent perquè funcioni la transparència

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SeleccioNivell">
```

```
<ImageView
    android:id="@+id/fondomontse"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0"
    app:srcCompat="@drawable/fondomontse" />
```

```
<ImageButton
    android:id="@+id/imageButton"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_marginStart="50dp"
    android:layout_marginBottom="50dp"
    app:layout_constraintBottom_toBottomOf="parent"
```

```
app:layout_constraintStart_toStartOf="parent"  
android:background="@android:color/transparent"  
app:srcCompat="@drawable/botobosc" />
```

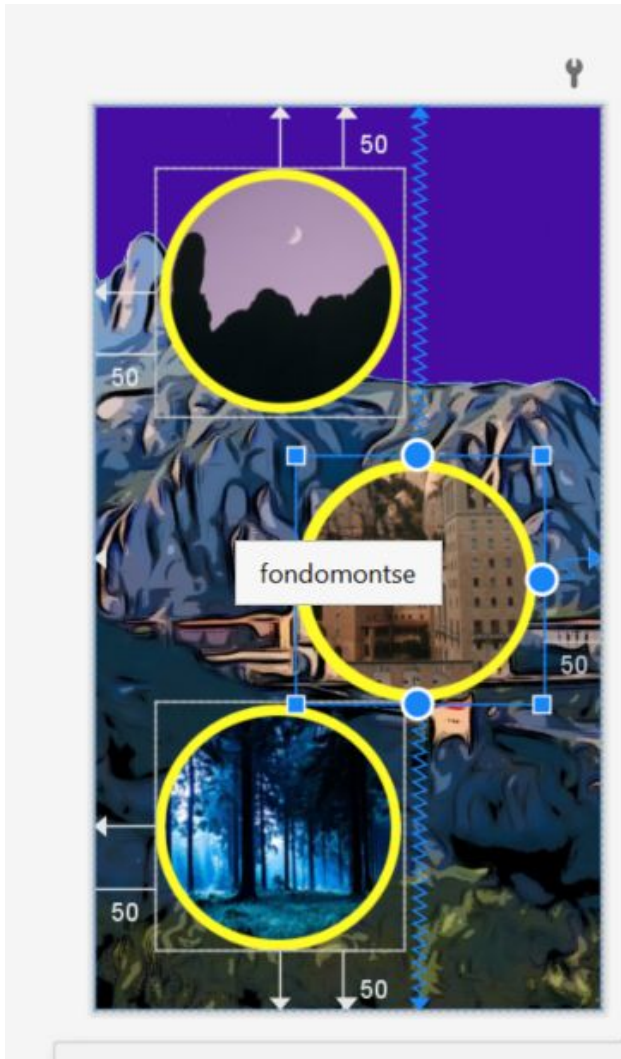
```
<ImageButton  
    android:id="@+id/imageButton2"  
    android:layout_width="200dp"  
    android:layout_height="200dp"  
    android:layout_marginEnd="50dp"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="@+id/fondomontse"  
    app:layout_constraintTop_toTopOf="parent"  
    app:layout_constraintVertical_bias="0.532"  
    android:background="@android:color/transparent"  
    app:srcCompat="@drawable/botomonestir" />
```

```
<ImageButton  
    android:id="@+id/imageButton3"  
    android:layout_width="200dp"  
    android:layout_height="200dp"  
    android:layout_marginStart="50dp"  
    android:layout_marginTop="50dp"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent"  
    android:background="@android:color/transparent"  
    app:srcCompat="@drawable/botocel" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

Fem constraint a les cantonades del layout amb uns marges, i el botó del mig es centra al layout i se li dona una coordenada de 50dp respecte al marge esquerra





Abans de tot hem de modificar-ho tot per a tenir una variable nova que es dirà nivell i pot ser 1,2 o 3

Anem a creació de personatge (registro) i afegim la variable nivell

```
fun updateUI(user:FirebaseUser?) {
    //hi ha un interrogant perquè podria ser null
    if (user!=null)
    {
        var puntuacio: String = "0"
        // Quan et registres es crea una clau única, la guardem per a
        identificar perfils
        var uidString: String = user.uid
        var correoString: String = correoEt.getText().toString()
        var passString: String = passEt.getText().toString()
        var nombreString: String = nombreEt.getText().toString()
        var fechaString: String= fechaTxt.getText().toString()
        var nivell: String = "1"

        //AQUI GUARDA EL CONTINGUT A LA BASE DE DADES
    }
}
```

```

// Utilitza un HashMap

var dadesJugador : HashMap<String,String> = HashMap<String,
String> ()
dadesJugador.put ("Uid",uidString)
dadesJugador.put ("Email",correoString)
dadesJugador.put ("Password",passString)
dadesJugador.put ("Nom",nombreString)
dadesJugador.put ("Data",fechaString)
dadesJugador.put ("Puntuacio",puntuacio)
dadesJugador.put ("Nivell", nivell)

```

A menu creem la variable

```

class Menu : AppCompatActivity() {
    //creem unes variables per comprovar usuari i autenticació
    lateinit var auth: FirebaseAuth
    var user:FirebaseUser? = null;
    lateinit var tancarSessio: Button
    lateinit var CreditsBtn: Button
    lateinit var PuntuacionsBtn: Button
    lateinit var jugarBtn: Button

    lateinit var miPuntuaciotxt: TextView
    lateinit var puntuacio: TextView
    lateinit var uid: TextView
    lateinit var correo: TextView
    lateinit var nom: TextView

    private var nivell ="1"

```

I quan busca el registre a la base de dades, capturem el valor del nivell

```

if (ds.child("Email").getValue().toString().equals(user?.email)){
    trobat=true
    //carrega els textview

    puntuacio.setText( ds.child("Puntuacio").getValue().toString())
    uid.setText( ds.child("Uid").getValue().toString())
    correo.setText( ds.child("Email").getValue().toString())
    nom.setText( ds.child("Nom").getValue().toString())
    nivell = ds.child("Nivell").getValue().toString()
}

```

Ara li donarem funcionalitat, primerament anem a menu, a que carregui aquest view quan es premi la tecla jugar

Hem de passar diversos valors a la activity SeleccioNivell, entre ells el nivell que determinarà quins botons es mostren

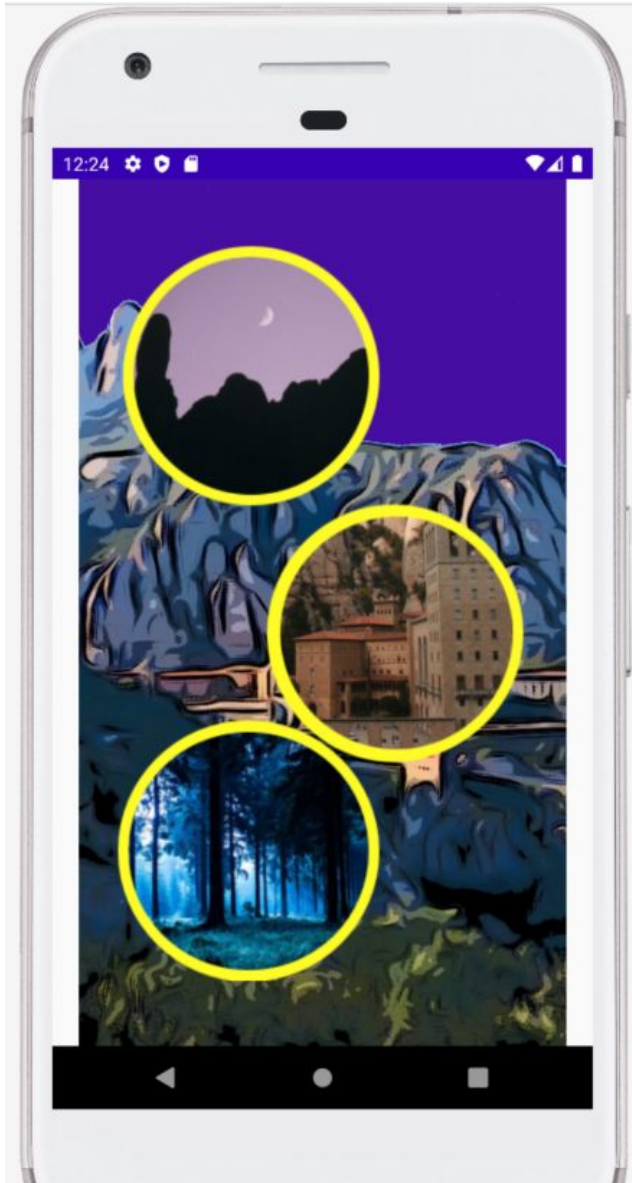
```
jugarBtn.setOnClickListener() {  
  
    //hem d'enviar el id, el nom i el contador, i el nivell  
    var Uids : String = uid.getText().toString()  
    var noms : String = nom.getText().toString()  
    var puntuacios : String = puntuacio.getText().toString()  
    var nivells : String = nivell  
  
    val intent= Intent(this, SeleccioNivell::class.java)  
    intent.putExtra("UID",Uids)  
    intent.putExtra("NOM",noms)  
    intent.putExtra("PUNTUACIO",puntuacios)  
    intent.putExtra("NIVELL",nivells)  
    startActivity(intent)  
    finish()  
  
}
```

**Els valors es recuperen a la pantalla seleccionivell**

```
private var NOM: String = ""  
private var PUNTUACIO: String=""  
private var UID: String=""  
private var NIVELL: String=""  
  
class SeleccioNivell : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_seleccio_nivell)  
        //ara recuperarem els valors  
        var intent:Bundle? = getIntent().extras  
        UID = intent?.get("UID").toString()  
        NOM = intent?.get("NOM").toString()  
        PUNTUACIO = intent?.get("PUNTUACIO").toString()  
        NIVELL = intent?.get("NIVELL").toString()  
    }  
}
```

```
}  
}
```

provem l'aplicació



Es veuen tres botons però a la pràctica només es veurà un d'ells, depenent del valor de nivell que tingui el jugador

Codifiquem perquè només surti el botó que volem, utilitzem GONE per a ocultar-lo i VISIBLE per a mostrar-lo

```
lateinit var imageButton1 :ImageButton  
lateinit var imageButton2 :ImageButton
```

```

lateinit var imageButton3 :ImageButton

class SeleccioNivell : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_seleccio_nivell)
        //ara recuperar els valors
        var intent:Bundle? = getIntent().extras
        UID = intent?.get("UID").toString()
        NOM = intent?.get("NOM").toString()
        PUNTUACIO = intent?.get("PUNTUACIO").toString()
        NIVELL = intent?.get("NIVELL").toString()

        //busco els 3 butons
        imageButton1 = findViewById(R.id.imageButton)
        imageButton2 = findViewById(R.id.imageButton2)
        imageButton3 = findViewById(R.id.imageButton3)

        //deshabilito els 3 buttons
        imageButton1.isEnabled=false
        imageButton2.isEnabled=false
        imageButton3.isEnabled=false

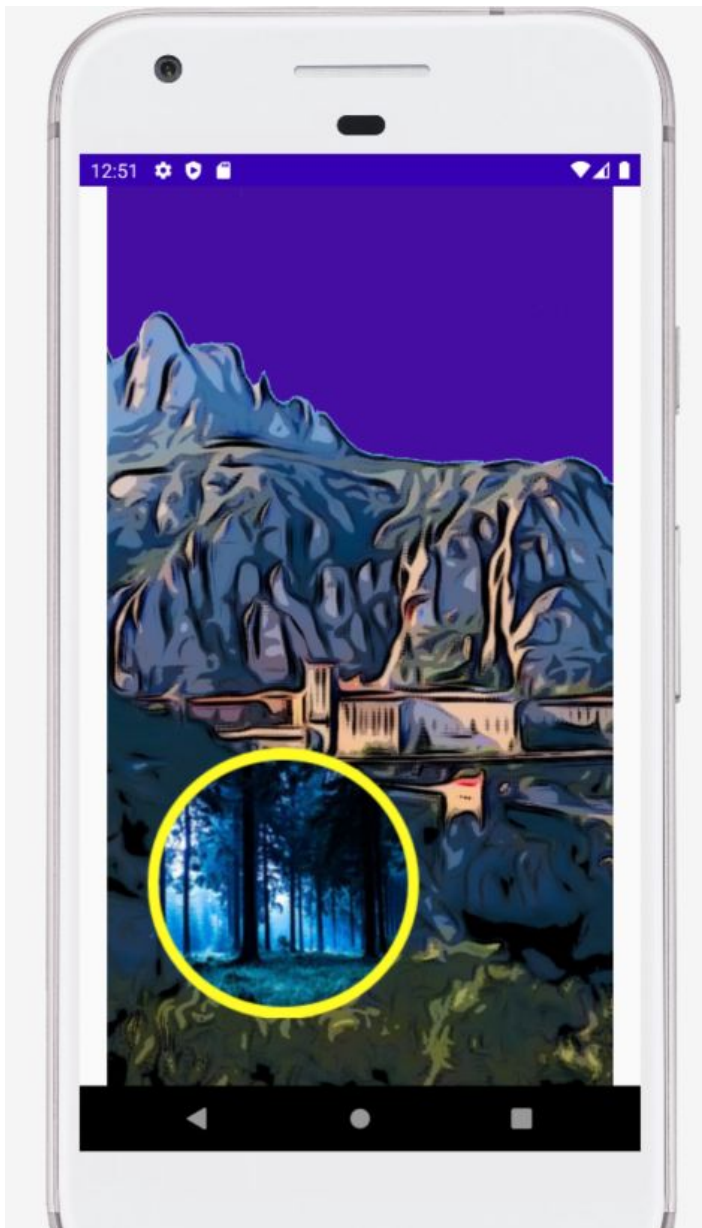
        imageButton1.visibility =View.GONE
        imageButton2.visibility =View.GONE
        imageButton3.visibility =View.GONE

        if (NIVELL == "1")    {
            Toast.makeText(this, "NIVELL 1", Toast.LENGTH_LONG).show()
            imageButton1.isEnabled=true
            imageButton1.visibility =View.VISIBLE
        }
        if (NIVELL == "2")
        {
            Toast.makeText(this, "NIVELL 2", Toast.LENGTH_LONG).show()
            imageButton2.isEnabled=true
            imageButton2.visibility =View.VISIBLE
        }
        if (NIVELL == "3") {
            Toast.makeText(this, "NIVELL 3", Toast.LENGTH_LONG).show()
            imageButton3.isEnabled=true
            imageButton3.visibility =View.VISIBLE
        }
    }
}

```

}

Ara quan comença, a nivell 1, la pantalla de selecció queda així:



Veiem que queden unes línies blanques al costat, aixó és perquè per defecte android no distorsiona la imatge, en aquest cas ens interessa que la imatge ocupi tota la pantalla sí o sí, tant se val si es deforma una mica, per aixó afegim **scaleType = "fitXY"** que expandirà tant x com y



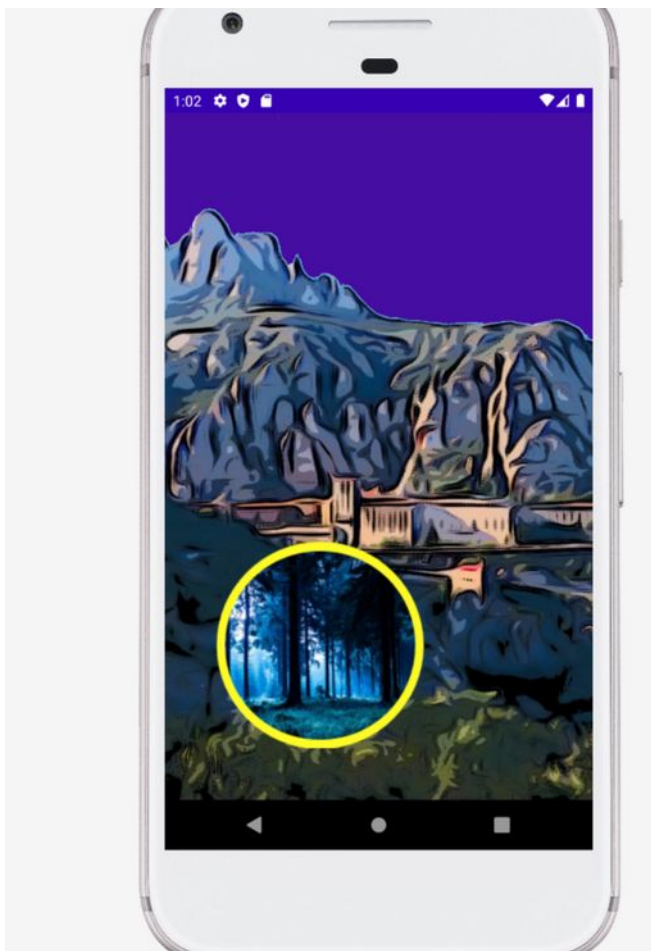
```

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".SeleccioNivell">

  <ImageView
    android:id="@+id/fondomontse"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0"
    android:scaleType="fitXY"
    app:srcCompat="@drawable/fondomontse" />

```

Efectivament ara queda així, sense les bandes blanques a cada costat

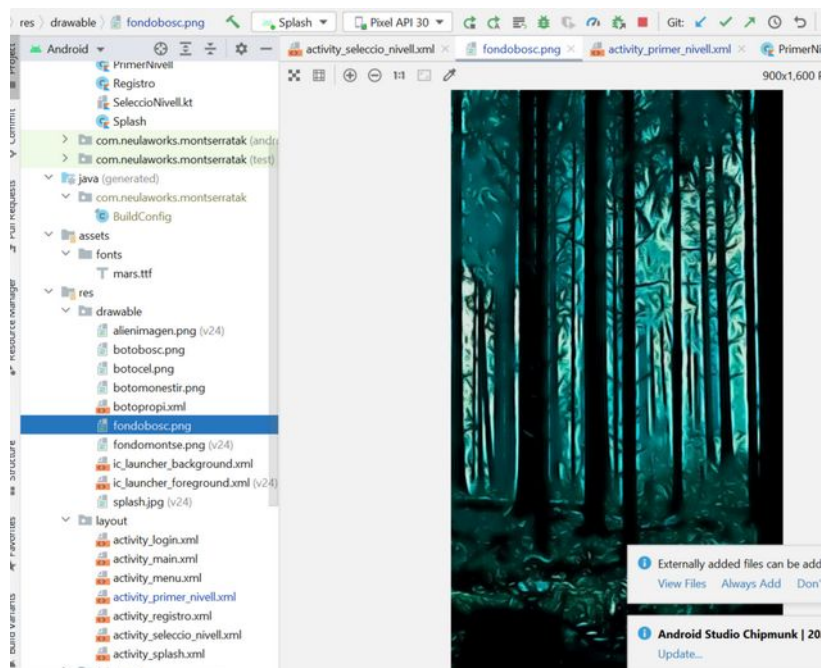


## Fase 6: Nivell 1

Farem 6 botons que cada un d'ells completa una part de la pantalla, la idea és que el contingut dels botons va canviant durant el temps; poden tenir res un humà o un et

Primer creem la pantalla que es diu primernivell

Pujem un arxiu que es diu fondobosc en format 16:9



I després treballarem sobre el layout: de entrada copiem el Imageview de seleccionivell i canviem la imatge per fondobosc

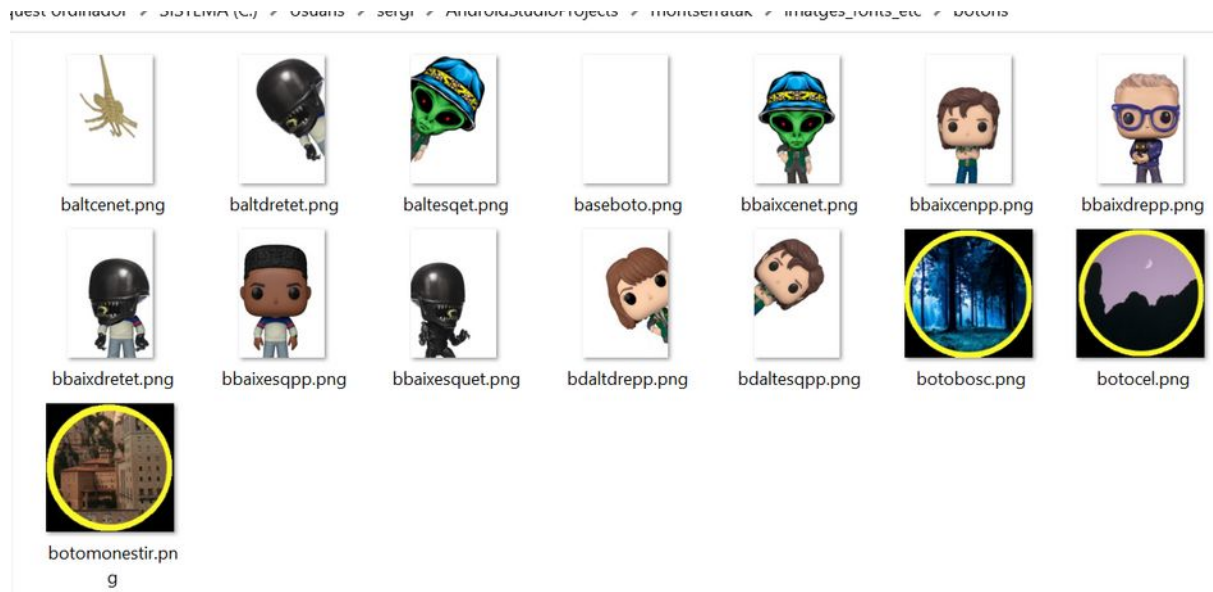
Ara col·locarem els botons aprofitant el constrained layout

Primerament hem de pujar els dibuixos dels botons, són aquests (atenció que baseboto.png és completament transparent, la resta tenen el fons com transparent)



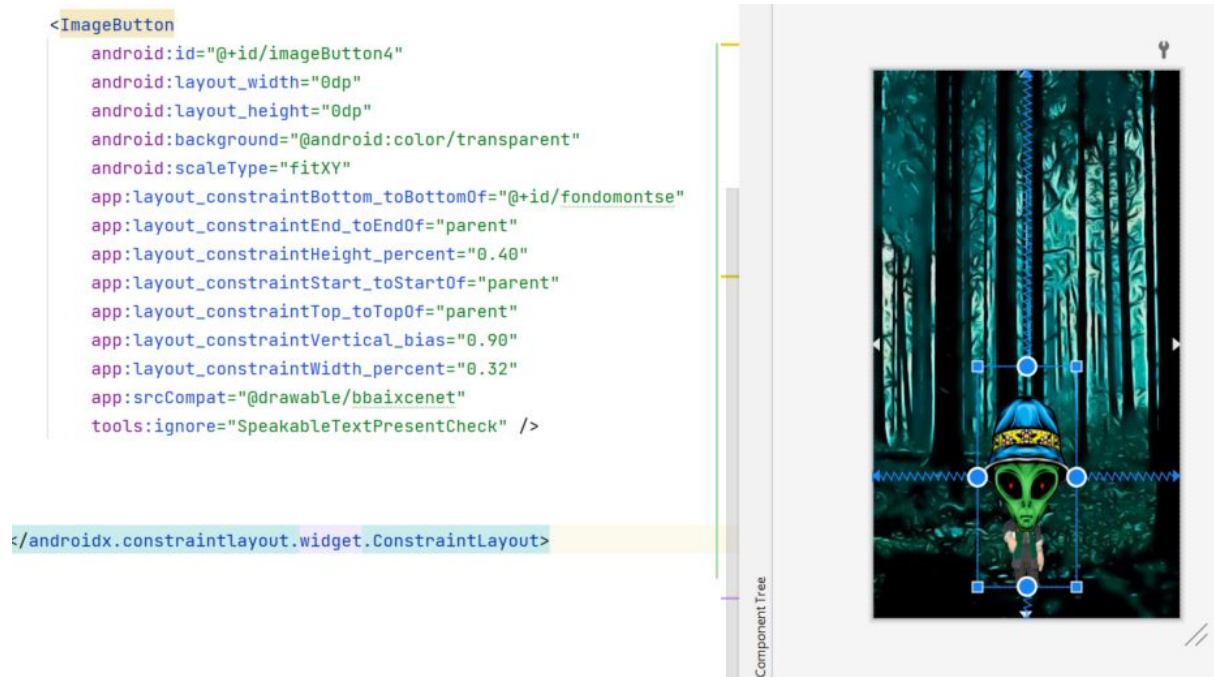
La carpeta de drawable ens queda així





- Drawable
- alienimagen.png (v24)
  - baltcenet.png
  - baltdretet.png
  - baltesqet.png
  - baseboto.png
  - bbaixcenet.png
  - bbaixcenpp.png
  - bbaixdrepp.png
  - bbaixdretet.png
  - bbaixesqpp.png
  - bbaixesquet.png
  - bdaltdrepp.png
  - bdaltesqpp.png
  - botobosc.png
  - botocel.png

Dibuixarem els primers botons com si tots ells tinguessin aliens (no hauria de passar durant el joc a no ser que el jugador sigui molt dolent)



Fem servir un primer botó com a patró i a partir d'aquest referenciem els altres.

Utilitzarem referència per % el que ens permetrà que els botons s'adaptin a tots els dispositius (l'únic inconvenient és que les imatges dels botons poden veure's afectades, s'aprimaran o s'engreixaran una mica depenent de la resolució de la pantalla)

El primer botó queda centrat horitzontalment al mig, i verticalment li diem

```
app:layout_constraintVertical_bias="0.90"
```

que correspon a un 90% de la pantalla

Pel que fa al ample i alt, utilitzem %:

```
app:layout_constraintHeight_percent="0.40"
```

```
app:layout_constraintWidth_percent="0.32"
```

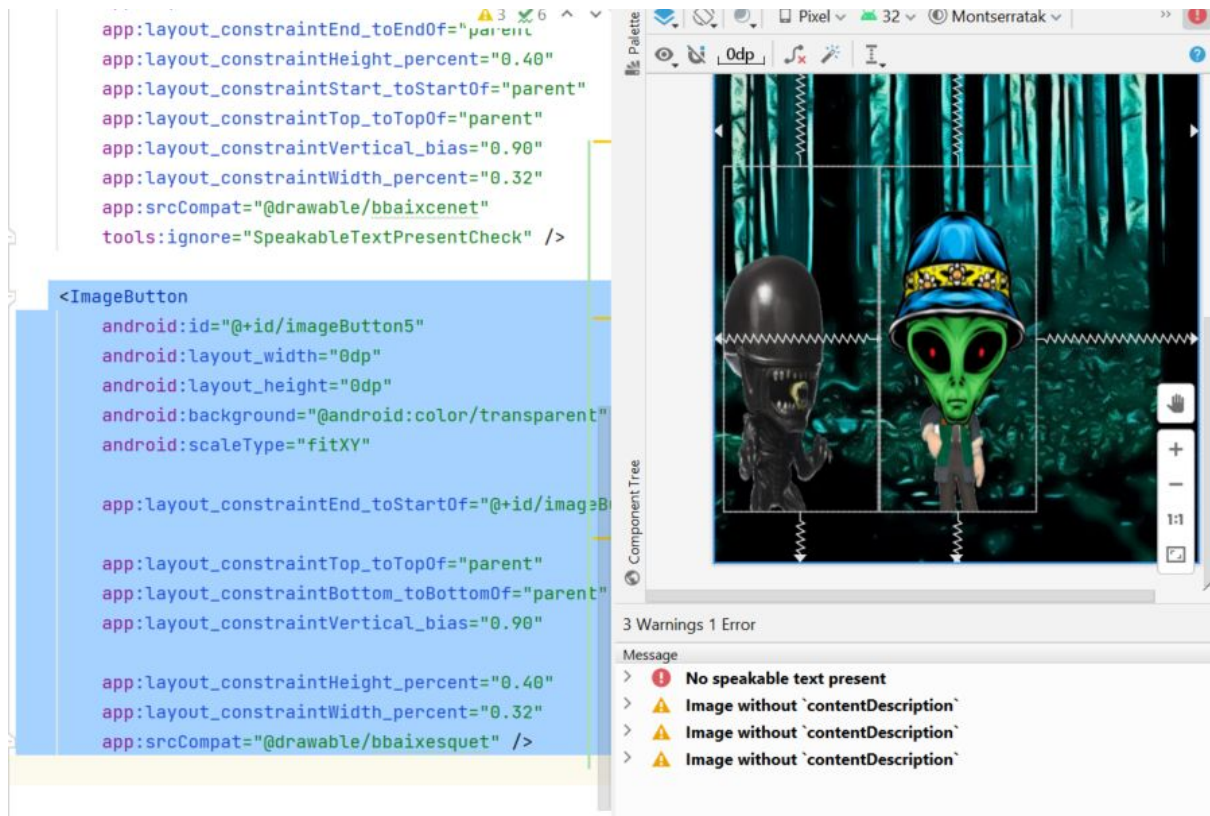
(són 2 fileres de 3 botons) l'ample total serà 96% de la pantalla i l'alt 80% de la pantalla

Aquests valors no es fan servir però és obligatori declarar-los

```
android:layout_width="0dp"
```

```
android:layout_height="0dp"
```

Veiem el següent botó per entendre com s'ha fet el constraint



Valors tamany (depenent de la mida del contenidor en %)

```

app:layout_constraintHeight_percent="0.40"
app:layout_constraintWidth_percent="0.32"

```

valors de posició vertical (entre mig del contenidor desplaçat 90%)

```

app:layout_constraintTop_toTopOf="parent"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintVertical_bias="0.90"

```

valors de posició horitzontal, respecte al botó 4 (el central)

```

app:layout_constraintEnd_toStartOf="@+id/imageButton4"

```

Al final els botons queden:

```

<?xml version="1.0" encoding="utf-8" ?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".PrimerNivell">

```

```

<ImageView
    android:id="@+id/fondomontse"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0"
    android:scaleType="fitXY"
    app:srcCompat="@drawable/fondobosc" />

<ImageButton
    android:id="@+id/imageButton4"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="@android:color/transparent"
    android:scaleType="fitXY"
    app:layout_constraintBottom_toBottomOf="@+id/fondomontse"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHeight_percent="0.40"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.98"
    app:layout_constraintWidth_percent="0.32"
    app:srcCompat="@drawable/bbaixcenet"
    tools:ignore="SpeakableTextPresentCheck" />

<ImageButton
    android:id="@+id/imageButton5"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="@android:color/transparent"
    android:scaleType="fitXY"

    app:layout_constraintEnd_toStartOf="@+id/imageButton4"

    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintVertical_bias="0.98"

    app:layout_constraintHeight_percent="0.40"
    app:layout_constraintWidth_percent="0.32"
    app:srcCompat="@drawable/bbaixesquet" />

<ImageButton

```

```

android:id="@+id/imageButton6"
android:layout_width="0dp"
android:layout_height="0dp"
android:background="@android:color/transparent"
android:scaleType="fitXY"

app:layout_constraintStart_toEndOf="@+id/imageButton4"

app:layout_constraintTop_toTopOf="parent"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintVertical_bias="0.98"

app:layout_constraintHeight_percent="0.40"
app:layout_constraintWidth_percent="0.32"

app:srcCompat="@drawable/bbaixdretet"
/>

```

```

<ImageButton
    android:id="@+id/imageButton7"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="@android:color/transparent"
    android:scaleType="fitXY"

    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintBottom_toTopOf="@+id/imageButton4"
    app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintHeight_percent="0.40"

    app:layout_constraintWidth_percent="0.32"
    app:srcCompat="@drawable/baltcenet" />

```

```

<ImageButton
    android:id="@+id/imageButton8"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="@android:color/transparent"
    android:scaleType="fitXY"

    app:layout_constraintBottom_toTopOf="@+id/imageButton5"
    app:layout_constraintEnd_toStartOf="@+id/imageButton7"

    app:layout_constraintHeight_percent="0.40"
    app:layout_constraintWidth_percent="0.32"
    app:srcCompat="@drawable/baltesqet" />

```

```

<ImageButton
    android:id="@+id/imageButton9"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="@android:color/transparent"
    android:scaleType="fitXY"

    app:layout_constraintBottom_toTopOf="@+id/imageButton6"
    app:layout_constraintHeight_percent="0.40"
    app:layout_constraintStart_toEndOf="@+id/imageButton7"
    app:layout_constraintWidth_percent="0.32"
    app:srcCompat="@drawable/baltdretet" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

Hem canviat el 0.90 a 0.98 per aprofitar la part baixa de la pantalla

Anem a veure com queda, primerament hem de fer un intent des de selecció de nivell a nivell1

```

imageButton1.setOnClickListener() {

    //hem d'enviar el id, el nom i el contador, i el nivell
    val intent= Intent(this, PrimerNivell::class.java)
    intent.putExtra("UID", UID)
    intent.putExtra("NOM", NOM)
    intent.putExtra("PUNTUACIO", PUNTUACIO)
    intent.putExtra("NIVELL", NIVELL)
    startActivity(intent)
    finish()

}

```



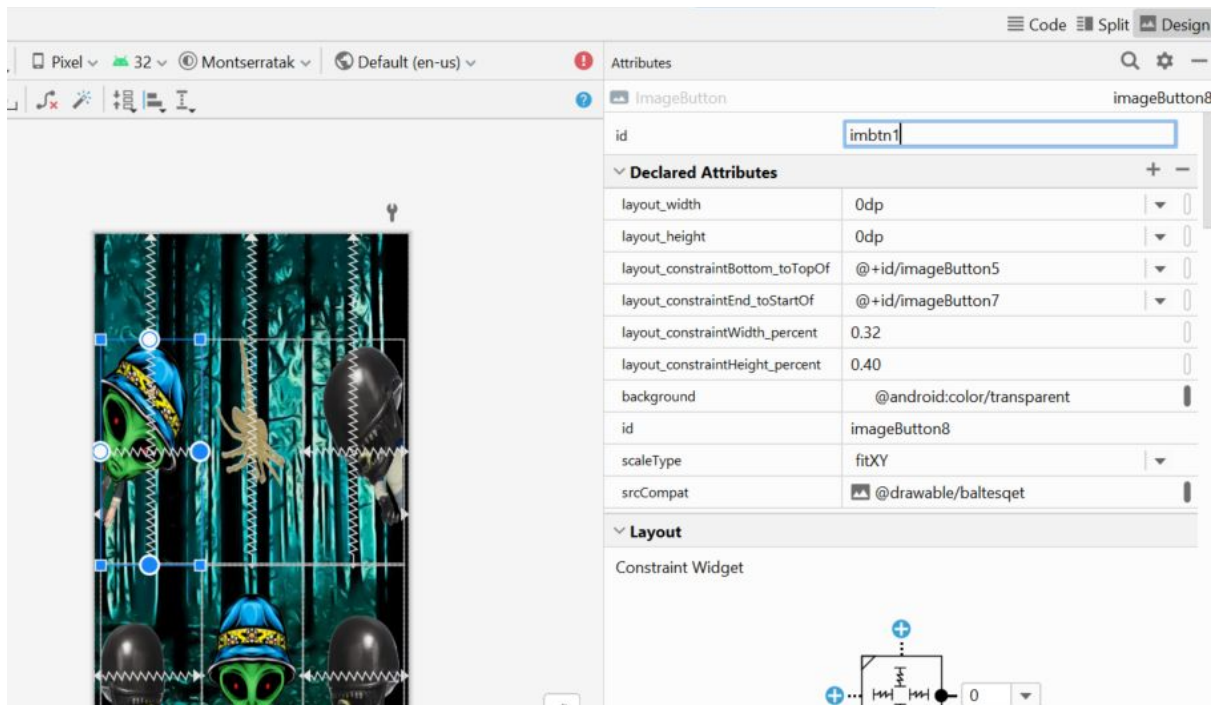




Ara renombrarem els botons de dalt a baix imbtn 1 2 3  
4 5 6

Ho fem a la finestra de Desing utilitzant refractor perquè s'actualitzin totes les referències





Si mirem el codi xml ja han canviat el valors i les referències

<ImageButton

```

    android:id="@+id/imbtn6"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="@android:color/transparent"
    android:scaleType="fitXY"

    app:layout_constraintStart_toEndOf="@+id/imbtn5"

    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintVertical_bias="0.98"

    app:layout_constraintHeight_percent="0.40"
    app:layout_constraintWidth_percent="0.32"

    app:srcCompat="@drawable/bbaixdretet"
/>

```

Ara farem que vagin canviant les imatges dels botons a PrimerNivell. Per fer-ho farem servir un countdown

El countdown funciona de manera molt fàcil. Es llança amb dos valors temps\_total i Interval.

Al codi a continuació podem veure que la variable time\_in\_mili\_seconds es un valor LONG que fa un compte enrere. Per exemple 100000L son 100 segons, i el valor de 1000 significa que cada 1000 miliseconds passarà per la funció onTick

```

countdown_timer = object: CountdownTimer (time_in_mili_seconds,1000)
{
    override fun onFinish() {
        //Ha acabat el comptador
        Log.i("DEBUG", "FINAL a temporitzador")
    }

    override fun onTick(millisUntilFinished: Long) {
        Log.i("DEBUG", "TIME")
        Log.i ("DEBUG",millisUntilFinished.toString())
    }
}
countdown_timer.start() //inicialitzo el procés

```

Quan acaba el llença la funcio onFinsih i a cada segon es llença la funció onTick

Anem a codificar el comptador:

Primer crearem un parell de variables per a configurar el comptador

```

lateinit var countdown_timer: CountdownTimer
var time_in_milli_seconds = 0L

```

I una array on guardarem el valor de cada botó (hi ha 6)

```

var contingut_botons = arrayOf(0,0,0,0,0,0) //poden ser 0 "res", 1
"alien", 2 "human"

```

Ara, des de onCreate, inicialitzem el comptador (aixó després ho canviarem i ho col·locarem a un botó)

```

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_primer_nivell)

    //busco els 3 butons
    imbtn1 = findViewById(R.id.imbtn1)
    imbtn2 = findViewById(R.id.imbtn2)
    imbtn3 = findViewById(R.id.imbtn3)
    imbtn4 = findViewById(R.id.imbtn4)
    imbtn5= findViewById(R.id.imbtn5)
    imbtn6 = findViewById(R.id.imbtn6)

```

```

time_in_milli_seconds = 100000L //100 segons
startTimer (time_in_milli_seconds)

```

```

canvio_imatges() //anem a una funció que canvia aleatoriament

```

```
les imatges
```

```
}
```

La funció que canvia les imatges és aquesta

```
private fun canvio_imatges() {  
    actualizto_array()  
    mostra_imatges()  
}
```

crida a mostra\_imatges

```
private fun mostra_imatges(){  
  
    var contingut: Int =0  
    for (position in contingut_botons.indices)  
    {  
        contingut= contingut_botons.get(position)  
        if (position==0)  
        {  
            //boto imbtn1  
            if (contingut==0)  
imbtn1.setImageResource(R.drawable.baseboto)  
            if (contingut==1)  
imbtn1.setImageResource(R.drawable.baltsetget)  
            if (contingut==2)  
imbtn1.setImageResource(R.drawable.bdaltesqpp)  
        }  
        if (position==1)  
        {  
            //boto imbtn2  
            if (contingut==0)  
imbtn2.setImageResource(R.drawable.baseboto)  
            if (contingut==1)  
imbtn2.setImageResource(R.drawable.baltcenet)  
        }  
        if (position==2)  
        {  
            //boto imbtn3  
            if (contingut==0)  
imbtn3.setImageResource(R.drawable.baseboto)  
            if (contingut==1)  
imbtn3.setImageResource(R.drawable.baltdretet)  
            if (contingut==2)
```

```

imbtn3.setImageResource(R.drawable.bdaldtrepp)
    }

    if (position==3)
    {
        //boto imbtn1
        if (contingut==0)
imbtn4.setImageResource(R.drawable.baseboto)
        if (contingut==1)
imbtn4.setImageResource(R.drawable.bbaixesquet)
        if (contingut==2)
imbtn4.setImageResource(R.drawable.bbaixesqpp)
    }

    if (position==4)
    {
        //boto imbtn1
        if (contingut==0)
imbtn5.setImageResource(R.drawable.baseboto)
        if (contingut==1)
imbtn5.setImageResource(R.drawable.bbaixcenet)
        if (contingut==2)
imbtn5.setImageResource(R.drawable.bbaixcenpp)
    }

    if (position==5)
    {
        //boto imbtn1
        if (contingut==0)
imbtn6.setImageResource(R.drawable.baseboto)
        if (contingut==1)
imbtn6.setImageResource(R.drawable.bbaixdrepp)
        if (contingut==2)
imbtn6.setImageResource(R.drawable.bbaixdrepp)
    }
}
}

```

Hi ha molt de codi però sempre fa el mateix, a partir d'un recorregut de l'array anem mirant cada posició i depenent del contingut pintem un dibuix o un altre al botó. Atenció perquè quan posem basebotó, com és transparent, no es veu cap imatge.

La funció `actualitzo_array` dona variables aleatòries a cada un dels valors de l'array, entre 0

```
private fun actualizto_array() {
    var r=Random()
    var retornarandom: Int
    for (position in contingut_botons.indices)
    {
        retornarandom=r.nextInt(3)
        //retornarà 0, 1 o 2
        contingut_botons.set(position,retornarandom)
    }
    //fa una comprovació perquè la posició tercera no pot ser mai
    humano (no hi ha dibuix)
    if (contingut_botons.get(2).equals(2)) contingut_botons.set(2,0)
}
```

Utilitzem un element Random, que cada vegada que es crida al mètode Random.nextInt(valor) retorna un número sencer entre 0 i valor-1

A continuació afegirem un botó de start i un parell de text, un per veure la puntuació i un de compte enrere

Primerament creem les strings a string.xml

```
<!-- strings PrimerNivell -->
<string name="btnStart">START</string>
<string name="puntuacioTxt"> -- </string>
<string name="countdownTxt"> -- </string>
```

Ara creem dos strings i un botó

```
<TextView
    android:id="@+id/puntuacioTxt"
    android:layout_width="0dp"
    android:textSize="30sp"
    android:textColor="@color/white"
    android:gravity="center_horizontal"
    app:layout_constraintWidth_percent="0.40"
    android:layout_height="wrap_content"
    android:background="@android:color/transparent"
    app:layout_constraintBottom_toBottomOf="@+id/fondomontse"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.10"
    android:text="@string/puntuacioTxt"
/>
```

```
<TextView
```

```

android:id="@+id/countdownTxt"
android:layout_width="0dp"
android:textSize="40sp"
android:textColor="@color/white"
android:gravity="center_horizontal"
app:layout_constraintWidth_percent="0.40"
android:layout_height="wrap_content"
android:background="@android:color/transparent"
app:layout_constraintBottom_toBottomOf="@+id/fondomontse"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.05"
android:text="@string/countdownTxt"
/>

```

<Button

```

android:id="@+id/btnStart"
android:layout_width="0dp"

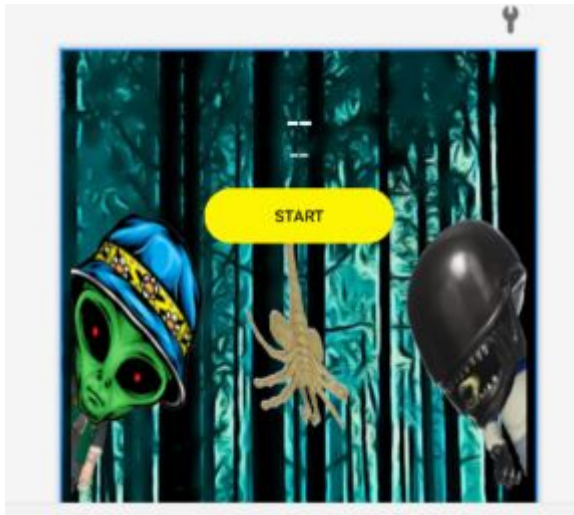
android:layout_height="wrap_content"
android:layout_marginTop="120dp"

android:background="@drawable/botopropi"
android:text="@string/btnStart"
android:textColor="@color/black"
app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintWidth_percent="0.40" />

```

Els podem ubicar amb constrain i tants per cent que sempre quedaran adequats a qualsevol pantalla



Queda una cosa així

Ara afegirem funcionalitat al botó (quan es premi desapareixerà de la pantalla i inicialitzarà el count down)

Primer creem les variables a PrimerNivell.kt

```
lateinit var btnStart: Button
lateinit var puntuacioTxt: TextView
lateinit var countdownTxt: TextView
```

Ara, a onCreate, farem que apuntin als elements corresponents

```
btnStart = findViewById(R.id.btnStart)
puntuacioTxt = findViewById(R.id.puntuacioTxt)
countdownTxt = findViewById(R.id.countdownTxt)
```

Al mateix mètode donem funcionalitat al botó, i onCreate queda així:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_primer_nivell)

    //busco els 3 butons
    imbtn1 = findViewById(R.id.imbtn1)
    imbtn2 = findViewById(R.id.imbtn2)
    imbtn3 = findViewById(R.id.imbtn3)
    imbtn4 = findViewById(R.id.imbtn4)
    imbtn5 = findViewById(R.id.imbtn5)
    imbtn6 = findViewById(R.id.imbtn6)

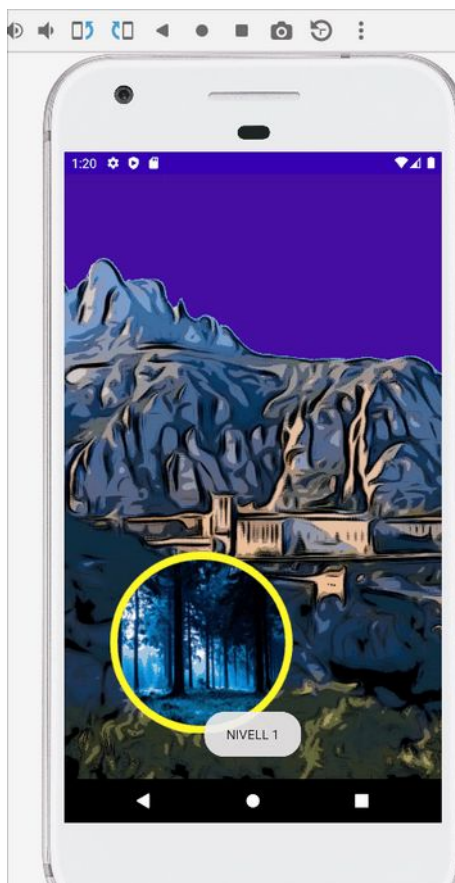
    btnStart = findViewById(R.id.btnStart)
    puntuacioTxt = findViewById(R.id.puntuacioTxt)
    countdownTxt = findViewById(R.id.countdownTxt)
```

```
time_in_milli_seconds = 100000L //100 segons  
  
mostra_imatges()
```

```
btnStart.setOnClickListener(){  
    // posem en marxa el timer  
    startTimer (time_in_milli_seconds)  
    // esborrem el botó  
    btnStart.visibility = View.INVISIBLE  
}
```

```
}
```

Provem la funcionalitat del botó



Ara donarem funcionalitat als botons imagebutton, de manera que en pulsar-los facin



diverses coses.

Primera: que mostrin una taca de sang (vermella si era humà o verda si era alien)

Primerament hem de crear dos pngs nous



sangalien.png



sanghuma.png

I els arrosseguem a drawable

Posem el comptador a 0

Primer creem una variable de classe

```
var puntsactuals: Int = 0 //els punts actuals del jugador
```

I al onCreate fem que puntuacioTxt mostri el valor d'aquesta variable

```
puntuacioTxt.setText(puntsactuals.toString())
```

I creem el listener del primer botó, el numero 1

```
imbtn1.setOnClickListener() {  
    //mirem si te un alien, un humà o res  
    if (contingut_botons.get(0).equals(1))  
    {  
        //es un alien, pinta sang d'alien  
        imbtn1.setImageResource(R.drawable.sangalien)  
        //incrementa el comptador en 10  
        contador(10)  
    }  
    if (contingut_botons.get(0).equals(2))  
    {  
        //es un humà, pinta sang d'humà  
        imbtn1.setImageResource(R.drawable.sanghuma)  
        //decrementa el comptador en -50  
        contador(-50)  
    }  
}
```

**I el mètode contador que incrementa o decrementa la puntuació i la mostra al textview corresponent**

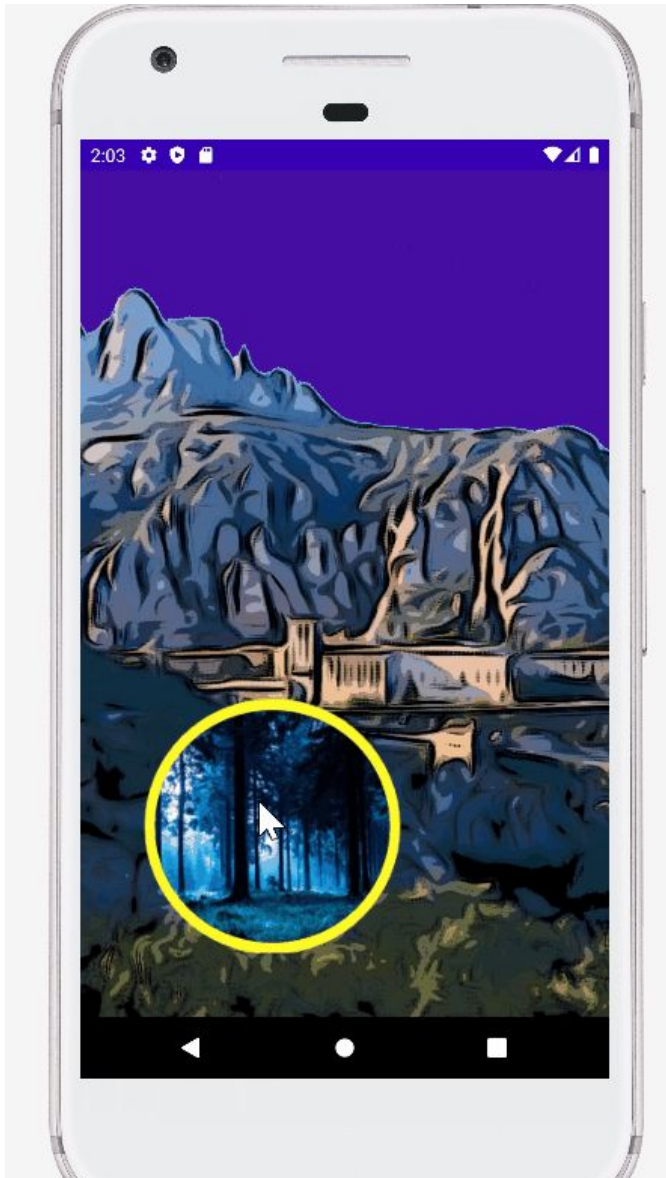
```
private fun contador(increment: Int) {  
    puntsactuals = puntsactuals + increment
```

```

    puntuacioTxt.setText(puntsactuals.toString())
}

```

Queda tal que així



Ara afegirem el contingut de l'altre textview, on hi ha el compte enrere, aixó ho fem dins del onTick del countdown (farem servir una variable intermitja per passar de milisegons a segons)

```

override fun onTick(millisUntilFinished: Long) {
    Log.i("DEBUG", "TIME")
    Log.i ("DEBUG",millisUntilFinished.toString())
    canvio_imatges()
    // mostro el valor a countdownTxt
    val segons:Long = millisUntilFinished/1000
}

```

```

        countdownTxt.setText(segons.toString())
    }

```

Canviem una mica el disseny: el compte enrere el col·loquem més a dalt amb un color groc i major tamany de lletra

```

android:textSize="80sp"
android:textColor="#fdfd04"

```

I al text de la puntuació afegim " punts" perquè s'entengui què és aquest valor, com que potser volem fer el programa en diversos llenguatges, anem a strings i afegim un valor

```

<string name="punts"> punts</string>

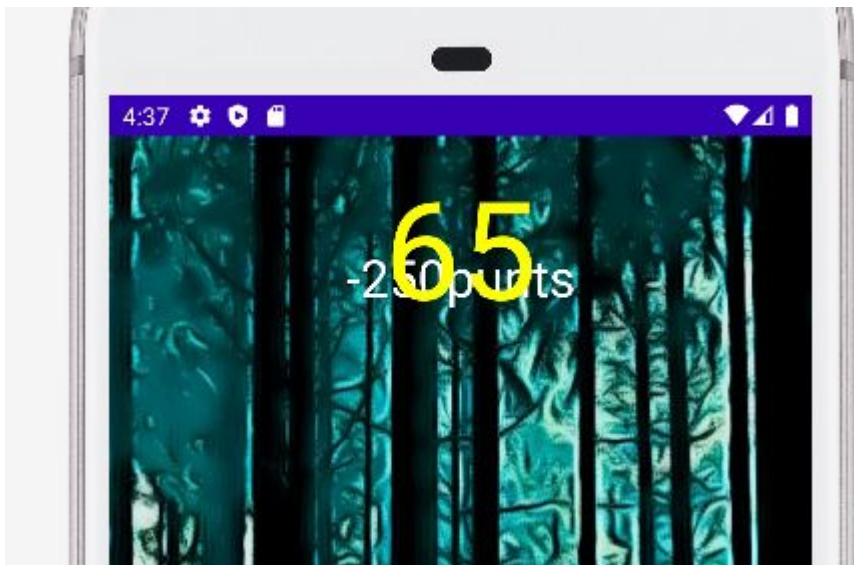
```

I des del codi cridarem a aquest name així

```

private fun contador(increment: Int) {
    puntsactuals = puntsactuals + increment
    var punts:String = getString(R.string.punts)
    puntuacioTxt.setText(puntsactuals.toString()+punts)
}

```



Canviem la posició dels números, el tamany, i al countdown li donem format amb la font mars.ttf

```

<TextView
    android:id="@+id/puntuacioTxt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="56dp"
    android:layout_marginEnd="16dp"
    android:background="@android:color/transparent"

```

```

    android:gravity="center_horizontal"
    android:text="@string/puntuacioTxt"
    android:textColor="@color/white"
    android:textSize="35sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintWidth_percent="0.40" />

```

```

<TextView
    android:id="@+id/countdownTxt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="20dp"
    android:layout_marginTop="30dp"
    android:background="@android:color/transparent"
    android:gravity="center_horizontal"
    android:text="@string/countdownTxt"
    android:textColor="#fdfd04"
    android:textSize="80sp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintWidth_percent="0.40" />

```

I a onCreate posem la font (aprofitem per posar la mateixa font al altre text i al botó)

```

val tf = Typeface.createFromAsset(assets, "fonts/mars.ttf")
countdownTxt.setTypeface(tf)
puntuacioTxt.setTypeface(tf)
btnStart.setTypeface(tf)

```

Provant l'aplicació veiem un problema:

Quan inicia el nivell hi ha punts de l'anterior jugador.

Si el nivell és el 1, llavors ha d'iniciar sempre a 0

```

if (NIVELL.compareTo("1")==0) puntsactuals=0 //si es el nivell 1
llavors possa els punts a 0

```



Ara que sembla que funciona, implementem els listeners del altres cinc botons (es podria fer amb una array de imagebuttons), però en el nostre cas farem copiar i enganxar.

```
imbtn1.setOnClickListener() {  
    //mirem si te un alien, un humà o res  
    if (contingut_botons.get(0).equals(1))  
    {  
        //es un alien, pinta sang d'alien  
        imbtn1.setImageResource(R.drawable.sangalien)  
        //incrementa el comptador en 10  
        contador(10)  
    }  
    if (contingut_botons.get(0).equals(2))  
    {  
        //es un humà, pinta sang d'humà  
        imbtn1.setImageResource(R.drawable.sanghumà)  
        //decrementa el comptador en -50  
        contador(-50)  
    }  
    //posem el contingut de boto a 0 (sinó si fes doble click  
    marcaria doble)  
    contingut_botons.set(0,0)  
}
```

```

imbtn2.setOnClickListener() {
    if (contingut_botons.get(1).equals(1))
    {
        imbtn2.setImageResource(R.drawable.sangalien)
        contador(10)
    }
    contingut_botons.set(1,0)
}

imbtn3.setOnClickListener() {
    if (contingut_botons.get(2).equals(1))
    {
        imbtn3.setImageResource(R.drawable.sangalien)
        contador(10)
    }
    if (contingut_botons.get(2).equals(2))
    {
        imbtn3.setImageResource(R.drawable.sanghuma)
        contador(-50)
    }
    contingut_botons.set(2,0)
}

imbtn4.setOnClickListener() {
    if (contingut_botons.get(3).equals(1))
    {
        imbtn4.setImageResource(R.drawable.sangalien)
        contador(10)
    }
    if (contingut_botons.get(3).equals(2))
    {
        imbtn4.setImageResource(R.drawable.sanghuma)
        contador(-50)
    }
    contingut_botons.set(3,0)
}

imbtn5.setOnClickListener() {
    if (contingut_botons.get(4).equals(1))
    {
        imbtn5.setImageResource(R.drawable.sangalien)
        contador(10)
    }
    if (contingut_botons.get(4).equals(2))
    {
        imbtn5.setImageResource(R.drawable.sanghuma)
        contador(-50)
    }
    contingut_botons.set(4,0)
}

```

```

imbtn6.setOnClickListener() {
    if (contingut_botons.get(5).equals(1))
    {
        imbtn6.setImageResource(R.drawable.sangalien)
        contador(10)
    }
    if (contingut_botons.get(5).equals(2))
    {
        imbtn6.setImageResource(R.drawable.sanghuma)
        contador(-50)
    }
    contingut_botons.set(5,0)
}

```

També fem que el canvi d'imatges sigui cada dos segons

```

override fun onTick(millisUntilFinished: Long) {
    // mostro el valor a countdownTxt
    val segons:Long = millisUntilFinished/1000
    countdownTxt.setText(segons.toString())

    //cada 2 segons canvio_imatges
    if ((segons.toFloat() % 2f)==0f) canvio_imatges()
}

```

Això ho fem calculant la divisió sencera; només quan el valor de segons dividit entre 2 sigui divisió sencera, canviarem les imatges.

Ja només ens resta desar els valors a Firebase i fer un canvi de nivell si hem fet més de 500 punts

Primerament hem de crear una serie de variables que ens envien des de SeleccioNivell

```

private var NOM: String = ""
private var PUNTUACIO: String=""
private var UID: String=""
private var NIVELL: String=""

```

I capturar-les a onCreate així: (com vam fer a SeleccioNivell)

```

var intent:Bundle? = intent.extras
UID = intent?.get("UID").toString()
NOM = intent?.get("NOM").toString()
PUNTUACIO = intent?.get("PUNTUACIO").toString()
NIVELL = intent?.get("NIVELL").toString()

```

Ara quan arriba a onFinish del timer, creem un nou mètode que es diu finalnivell

```

countdown_timer = object: CountdownTimer (time_in_mili_seconds,1000)
{
    override fun onFinish() {
        //Ha acabat el comptador
        Log.i("DEBUG", "FINAL a temporitzador")
        finalNivell()
    }
}

```

Creem també un boto de continuar que serà invisible al principi

Com sempre primer un string name

```
<string name="continuarBtn">Continuar</string>
```

I el botó

```

<Button
    android:id="@+id/continuarBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/continuarBtn"
    android:background="@drawable/botopropi"
    android:textColor="@color/black"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/fondomontse" />

```

Ara a PrimerNivell.kt

```
lateinit var continuarBtn: Button
```

I a onCreate

```

continuarBtn = findViewById(R.id.continuarBtn)
continuarBtn.setTypeface(tf)
continuarBtn.visibility = View.INVISIBLE

```

finalnivell ha de fer:

- guardar les dades de puntuació, la data i el nivell
- incrementar el nivell si cal
- Mostrar un fons per nivell superat i un altre per no superat
- mostrar un botó per sortir (indicant si s'ha superat o no)

Fem els dos fons en format 16:9





defeat.png



victory.png

I els arroseguem a drawable

Botó de sortir

Anar al menú de l'aplicació

El Mètode final de nivell queda així:

```
private fun finalNivell(){
    var database: FirebaseDatabase =
    FirebaseDatabase.getInstance("https://montserratak-76f14-default-
    rtdb.europe-west1.firebaseio.com/")
    var reference: DatabaseReference = database.getReference("DATA
    BASE JUGADORS")

    //captura la data
    val date = Calendar.getInstance().time
    val formatter = SimpleDateFormat.getDateInstance()
    val formattedDate = formatter.format(date)

    var nivell:String ="1"

    var fondo:ImageView = findViewById(R.id.fondomontse)
    if (puntsactuals>500){
        Log.i ("DEBUG","mostra victory")
        //canvia la imatge per victory
        fondo.setImageResource(R.drawable.victory)
        nivell="2"
    }
    else{
        Log.i ("DEBUG","mostra defeat")
        fondo.setImageResource(R.drawable.defeat)
        nivell="1"
        //canvia la imatge per defeat
    }

    //grava les dades del jugador (puntuació, nivell i Data)
    //accedint directament al punt del arbre de dades que volem anar,
```

podem modificar

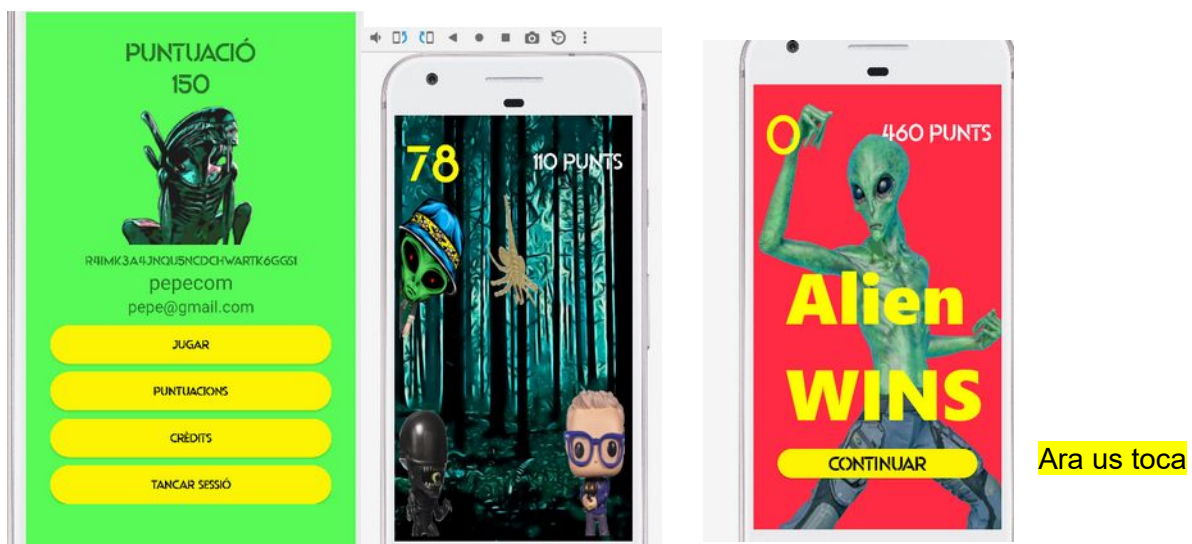
```
//només una de les dades sense que calgui canviar tots els camps:  
nom, email...
```

```
reference.child(UID).child("Puntuacio").setValue(puntsactuals.toString()  
)  
reference.child(UID).child("Nivell").setValue(nivell)  
reference.child(UID).child("Data").setValue(formatedDate)  
  
//posa tots els valors dels imagebuttons a 0  
for (position in contingut_botons.indices)  
contingut_botons.set(position,0)  
  
//crida al mètode de mostra imatges  
mostra_imatges()  
  
//ja hem gravat les dades  
//ara mostrem el botó de continuar  
Log.i ("DEBUG","mostra boto continuar")  
continuarBtn.visibility = View.VISIBLE  
}
```

I el botó de sortir torna a la pantalla principal

```
continuarBtn.setOnClickListener() {  
    //va a la pantalla inicial  
    val intent= Intent(this, MainActivity::class.java)  
    startActivity(intent)  
    finish()  
}
```

Provant l'aplicació veiem que ja guarda la puntuació i la mostra a la pantalla del menú i si el resultat és menor de 500 guanyen aliens:



afegir so utilitzant SoundPool

## Fase 7: Implementació de Nivells 2 i 3

Els nivells 2 i 3 seran el nivell 1 amb algunes diferències: el fons i la dificultat

Per a implementar-los farem servir el mateix nivell 1 que d'entrada canviarà només el fons depenent si el nivell és el 1, el 2 o el 3

Hem creat dos dibuixos nous, el fondoncel i el fondonestir



fondocel.png



fondonestir.p  
ng

Els arrosseguem a drawable

Ara ja podem canviar el codi del joc. Primerament anem a selecció de nivell i farem que a qualsevol nivell, faci un intent a PrimerNivell.ktç

Teníem:

```
imageButton1.setOnClickListener(){ it: View!  
  
    //hem d'enviar el id, el nom i el contador, i el nivell  
    val intent= Intent( packageContext: this, PrimerNivell::class.java)  
  
    intent.putExtra( name: "UID", UID)  
    intent.putExtra( name: "NOM", NOM)  
    intent.putExtra( name: "PUNTUACIO", PUNTUACIO)  
    intent.putExtra( name: "NIVELL", NIVELL)  
    Log.i( tag: "DEBUG", msg: "UID enviat:")  
    Log.i( tag: "DEBUG", UID)  
    startActivity(intent)  
    finish()  
  
}
```

Com els tres imageButtons portaran a PrimerNivell, en comptes de repetir el codi per a cada imageButton.setOnClickListener, el que fem és fer un mètode i que tots vagin a aquest mètode. D'aquesta manera:

```

        imageButton1.setOnClickListener(){ it: View!
            canviaNivell()
        }
        imageButton2.setOnClickListener(){ it: View!
            canviaNivell()
        }
        imageButton3.setOnClickListener(){ it: View!
            canviaNivell()
        }
    }

    private fun canviaNivell() {
        //hem d'enviar el id, el nom i el contador, i el nivell
        val intent= Intent( packageContext: this, PrimerNivell::class.java)
        intent.putExtra( name: "UID", UID)
        intent.putExtra( name: "NOM", NOM)
        intent.putExtra( name: "PUNTUACIO", PUNTUACIO)
        intent.putExtra( name: "NIVELL", NIVELL)
        Log.i( tag: "DEBUG", msg: "UID enviat:")
        Log.i( tag: "DEBUG", UID)
        startActivity(intent)
        finish()
    }
}

```

Ara ja podem canviar el fons a PrimerNivell depenent del nivell on es trobi l'usuari, i ho fem així, al onCreate

```

time_in_milli_seconds = 100000L //100 segons

var fondo:ImageView = findViewById(R.id.fondomontse)
if (NIVELL.compareTo("2")==0) {
fondo.setImageResource(R.drawable.fondomonestir) }
if (NIVELL.compareTo("3")==0) {
fondo.setImageResource(R.drawable.fondocel) }

if (NIVELL.compareTo("1")==0) {
    puntsactuals = 0
    //si es el nivell 1 llavors possa els punts a 0
} else {
    puntsactuals=PUNTUACIO.toInt()
}
}

```

També incrementarem la dificultat del joc en aquest nivell, amb la velocitat de canvi dels personatges (al nivell 1 canviara cada 3 segons, al nivell 2 cada 2 segons i al nivell 1 cada segon) Aixó ho fem amb una variable float que es diu segonsCanvi i que varia depenent de a quin nivell estem

```

override fun onTick(millisUntilFinished: Long) {
    var segonsCanvi =3f
}

```

```

val segons:Long = millisUntilFinished/1000
countdownTxt.setText(segons.toString())
Log.i("DEBUG", "COUNTDOWN")

if (NIVELL.compareTo("2")==0) { segonsCanvi=2f}
if (NIVELL.compareTo("3")==0) { segonsCanvi=1f}
//cada x segons canvioimatges
if ((segons.toFloat() % segonsCanvi)==0f) canvioimatges()
}

```

Establim els requisits per a canviar de nivell. Utilitzem una variable boolean i la puntuació total per decidir si passem de nivell o no

```

var nivell:String = "1"
var guanya: Boolean =false //true si es guanya
var fondo:ImageView = findViewById(R.id.fondomontse)

if (NIVELL.toInt()==1 && puntsactuals>200){ guanya=true}
if (NIVELL.toInt()==2 && puntsactuals>500){ guanya =true}
if (NIVELL.toInt()==3 && puntsactuals>800){guanya =true}

if (guanya){
    Log.i ("DEBUG","mostra victory")
    //canvia la imatge per victory
    fondo.setImageResource(R.drawable.victory)
    if (NIVELL.toInt()==1) {nivell="2"}
    if (NIVELL.toInt()==2) {nivell="3"}
    if (NIVELL.toInt()==3) {nivell="1"} //Torna a començar
}
else{
    Log.i ("DEBUG","mostra defeat")
    //canvia la imatge per defeat
    fondo.setImageResource(R.drawable.defeat)
    nivell=NIVELL //no avança de nivell
    //(NIVELL es la variable que hem passat amb
    Bundle )
}
}

```

Podem ajustar la dificultat del joc amb els valors mínims de puntuació per a passar de nivell:

```

if (NIVELL.toInt()==1 && puntsactuals>200){ guanya=true}
if (NIVELL.toInt()==2 && puntsactuals>500){ guanya =true}
if (NIVELL.toInt()==3 && puntsactuals>800){guanya =true}

```

I la durada amb la variable  
`time_in_milli_seconds = 60000L //60 segons`

## Fase 8: Imatges amb Picasso i Intents implícits

En aquesta fase veurem com utilitzar una llibreria d'imatges com Picasso, fer Intents implícits, l'ús de AlertDialog i com funcionen els permisos de android studio. També utilitzarem una eina que ens permet guardar documents a Firebase: Storage.

D'entrada volem afegir més dades: població, edat, i una imatge

Al registre afegim tres dades més, primerament anem a string i afegim la població i l'edat

```
<!-- Activitat registre -->
<string name="correoEt">E-mail</string>
<string name="nombreEt">Nom</string>
<string name="passEt">Password</string>
<string name="fechaTxt">Data</string>
<string name="Registrar">REGISTRAR</string>
<string name="edatEt">Edat</string>
<string name="poblacioEt">Població</string>
```

Ara anem a Activity\_registro.xml i afegim 2 edit text, edat i població

```
<!-- ENTREM EL NOM DEL JUGADOR -->
<EditText
    android:id="@+id/nombreEt"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/nombreEt" />

<!-- ENTREM L'edat -->
<EditText
    android:id="@+id/edatEt"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/edatEt" />

<!-- ENTREM LA POBLACIO -->
<EditText
    android:id="@+id/poblacioEt"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/poblacioEt" />
```

Afegirem una imatge amb PICASSO

Picasso és una biblioteca popular d'Android de codi obert per a la càrrega d'imatges locals i



remotes.

<https://rubentejera.com/picasso-libreria-gestion-imagenes-para-android/>

<https://square.github.io/picasso/#download>



The screenshot shows the Picasso library website. At the top, there's a red header with the word "Picasso" on the left and "Download Latest" with GitHub and Android icons on the right. Below the header, there are two code blocks. The first block shows XML dependency code: 

```
<dependency>
<groupId>com.squareup.picasso3</groupId>
<artifactId>picasso</artifactId>
<version>(insert latest version)</version>
</dependency>
```

 The second block is titled "GRADLE" and shows the corresponding Gradle implementation: 

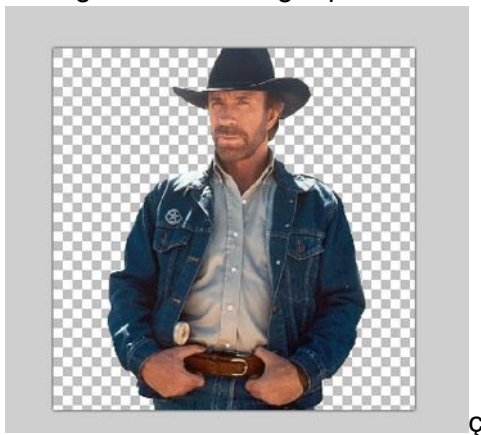
```
implementation 'com.squareup.picasso:picasso:(insert latest version)'
```

Hem d'afegir al gradle:

```
//base de dades
//implementation 'com.google.firebase:firebase-database-ktx'
implementation 'com.google.firebase:firebase-database-ktx:20.0.5'
```

```
//picasso
implementation 'com.squareup.picasso:picasso:2.71828'
```

Ara agafem una imatge que serà l'usuari per defecte (300x300 pixels)



I l'arroseguem a drawable

I a activity registro.xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="30dp">
```

```

<!-- IMATGE -->
<androidx.appcompat.widget.AppCompatImageView
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:id="@+id/jugador"
    android:src="@drawable/jugador" />

```

Ara a la pantalla registro hem d'afegir edat i poblacio

```

//Definim les variables que farem servir
//lateinit ens permet no inicialitzar-les encara
lateinit var correoEt :EditText
lateinit var passEt :EditText
lateinit var nombreEt :EditText
lateinit var fechaTxt :TextView
lateinit var Registrar : Button

```

```

lateinit var edatEt :EditText
lateinit var poblacioEt :EditText

```

Connexió amb la vista (R)

```

// Busquem a R els elements als que apunten les variables
correoEt =findViewById<EditText>(R.id.correoEt)
passEt =findViewById<EditText>(R.id.passEt)
nombreEt =findViewById<EditText>(R.id.nombreEt)
fechaTxt =findViewById<TextView>(R.id.fechaEt)
Registrar =findViewById<Button>(R.id.Registrar)
edatEt =findViewById<EditText>(R.id.edatEt)
poblacioEt =findViewById<EditText>(R.id.poblacioEt)

```

Al mètode de registrar el jugador updateUI fem la connexió amb les strings

```

fun updateUI(user:FirebaseUser?) {
    //hi ha un interrogant perquè podria ser null
    if (user!=null)
    {
        var puntuacio: String = "0"
        // Quan et registres es crea una clau única, la guardem per a
        identificar perfils
        var uidString: String = user.uid
        var correoString: String = correoEt.getText().toString()
        var passString: String = passEt.getText().toString()
        var nombreString: String = nombreEt.getText().toString()
        var fechaString: String= fechaTxt.getText().toString()
        var nivell: String = "1"

        var edatString = edatEt.getText().toString()
        var poblacioString = poblacioEt.getText().toString()
    }
}

```

I els enviem a firebase a través de put

```
//AQUI GUARDA EL CONTINGUT A LA BASE DE DADES
// Utilitza un HashMap
var dadesJugador : HashMap<String,String> = HashMap<String, String>
()
dadesJugador.put ("Uid",uidString)
dadesJugador.put ("Email",correoString)
dadesJugador.put ("Password",passString)
dadesJugador.put ("Nom",nombreString)
dadesJugador.put ("Data",fechaString)
dadesJugador.put ("Edat",edatString)
dadesJugador.put ("Poblacio",poblacioString)
dadesJugador.put ("Imatge","")
dadesJugador.put ("Puntuacio",puntuacio)
dadesJugador.put ("Nivell", nivell)
```

Hem creat també un camp Imatge que el farem servir més endavant

Ara anem al menú on afegirem aquests dos camps, primerament anem a Strings

```
<!-- Activity Menu -->
<string name="tancarSessio">Tancar Sessió</string>
<string name="miPuntuaciotxt">Puntuació</string>
<string name="puntuacio"> -- </string>
<string name="uid"> -- </string>
<string name="correo"> -- </string>
<string name="nom"> -- </string>
<string name="MenuTxt">MENÚ</string>
<string name="jugarBtn">Jugar</string>
<string name="PuntuacionsBtn">Puntuacions</string>
<string name="CreditsBtn">Crèdits</string>
<string name="edat"> -- </string>
<string name="poblacio"> -- </string>
```

Ara des de activity\_menu.xml afegim els camps que volem veure, que sera la imatge de l'usuari, l'edat i la població

```
<!-- correu -->
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/correo"
    android:textSize="20sp"
    android:text="@string/correo"
    android:gravity="center"/>
```

```
<!-- edat -->
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/edat"
    android:textSize="15sp"
    android:text="@string/edat"
    android:gravity="center"/>
```

```
<!-- Població -->
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/poblacio"
    android:textSize="15sp"
    android:text="@string/poblacio"
    android:gravity="center"/>
```

La imatge utilitzarem la que feiem servir del alien una mica més petita

```
<!-- Imatge alien -->
<androidx.appcompat.widget.AppCompatImageView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:id="@+id/imatgePerfil"
    android:src="@drawable/jugador" />
```

Canviem <!--imatge alien --> per imatgePerfil per recordar que és la imatge del jugador



Afegim també un botó de Editar (en aquest cas només podrem canviar l'imatge)

Primer a strings

```
<!-- Activituy Menu -->
<string name="tancarSessio">Tancar Sessió</string>
<string name="miPuntuaciotxt">Puntuació</string>
<string name="puntuacio"> -- </string>
<string name="uid"> -- </string>
<string name="correo"> -- </string>
<string name="nom"> -- </string>
<string name="MenuTxt">MENÚ</string>
<string name="jugarBtn">Jugar</string>
<string name="editarBtn">Canviar Imatge</string>
```

I al xml

```
<!-- BOTO DE jugar -->
<Button
    android:background="@drawable/botopropi"
    android:textColor="@color/black"
    android:id="@+id/jugarBtn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/jugarBtn"
    android:layout_marginTop="10dp"
/>
```

```
<!-- Boto d'Editar -->
<Button
    android:background="@drawable/botopropi"
```

```

    android:textColor="@color/black"
    android:id="@+id/editarBtn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/editarBtn"
    android:layout_marginTop="10dp"
/>

```

I quedarà així



Ara afegim el botó nou, la imatge del perfil, i les dades de edat i població a Menu.kt

```

lateinit var tancarSessio: Button
lateinit var CreditsBtn: Button
lateinit var PuntuacionsBtn: Button
lateinit var jugarBtn: Button
lateinit var editarBtn: Button

lateinit var miPuntuaciotxt: TextView
lateinit var puntuacio: TextView
lateinit var uid: TextView
lateinit var correo: TextView
lateinit var nom: TextView
lateinit var edat: TextView
lateinit var poblacio: TextView

lateinit var imatgePerfil: ImageView

```

### A on create

```
editarBtn = findViewById<Button>(R.id.editarBtn)
```

```
edat=findViewById(R.id.edat)
```

```
poblacio=findViewById(R.id.poblacio)
```

```
imatgePerfil=findViewById(R.id.imatgePerfil)
```

//Assignem tipus de lletra al botó

```
editarBtn.setTypeface(tf)
```

I col·loquem un listener del botó

```
editarBtn.setOnClickListener() {  
    Toast.makeText(this, "EDITAR", Toast.LENGTH_SHORT).show()  
}
```

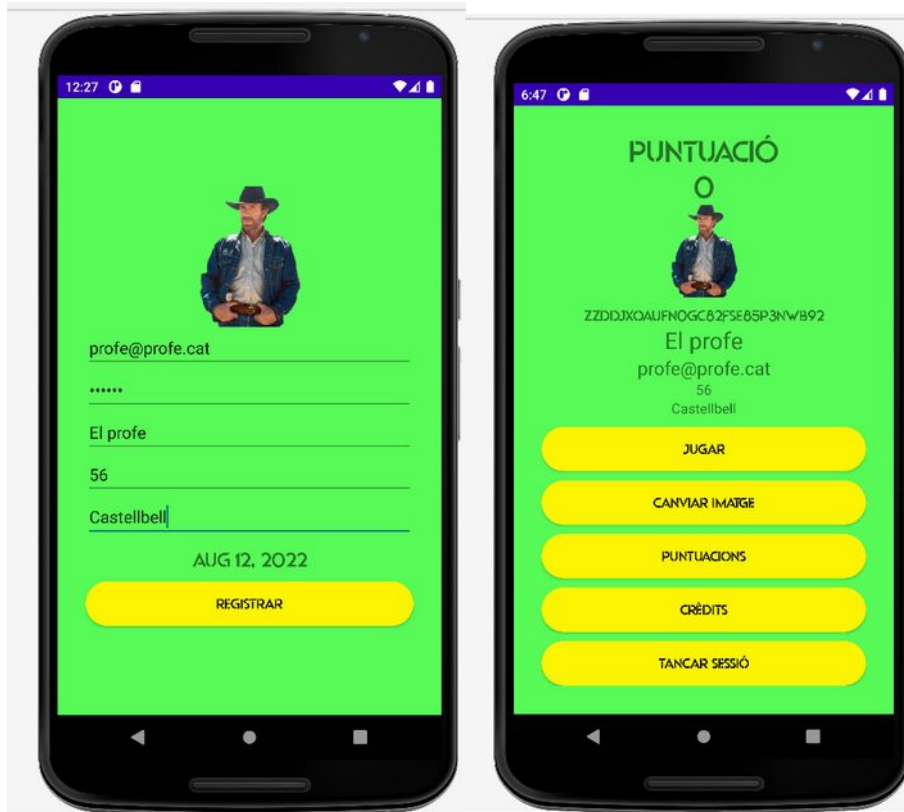
I al mètode consulta

```
puntuacio.setText(ds.child("Puntuacio").getValue().toString())  
uid.setText(ds.child("Uid").getValue().toString())  
correo.setText(ds.child("Email").getValue().toString())  
nom.setText(ds.child("Nom").getValue().toString())  
nivell = ds.child("Nivell").getValue().toString()  
poblacio.setText(ds.child("Poblacio").getValue().toString())  
edat.setText(ds.child("Edat").getValue().toString())  
var imatge: String = ds.child("Imatge").getValue().toString()  
  
Picasso.get().load(imatge).into(imatgePerfil);
```

com és possible que no hi hagi imatge, perquè quan es carrega la primer no hi ha res, fem un catch, si no hi ha imatge, col·locarem la del chuck

```
try {  
    Picasso.get().load(imatge).into(imatgePerfil)  
} catch (e:Exception){  
    Picasso.get().load(R.drawable.jugador).into(imatgePerfil)  
}
```

Provem el programa fins aquí

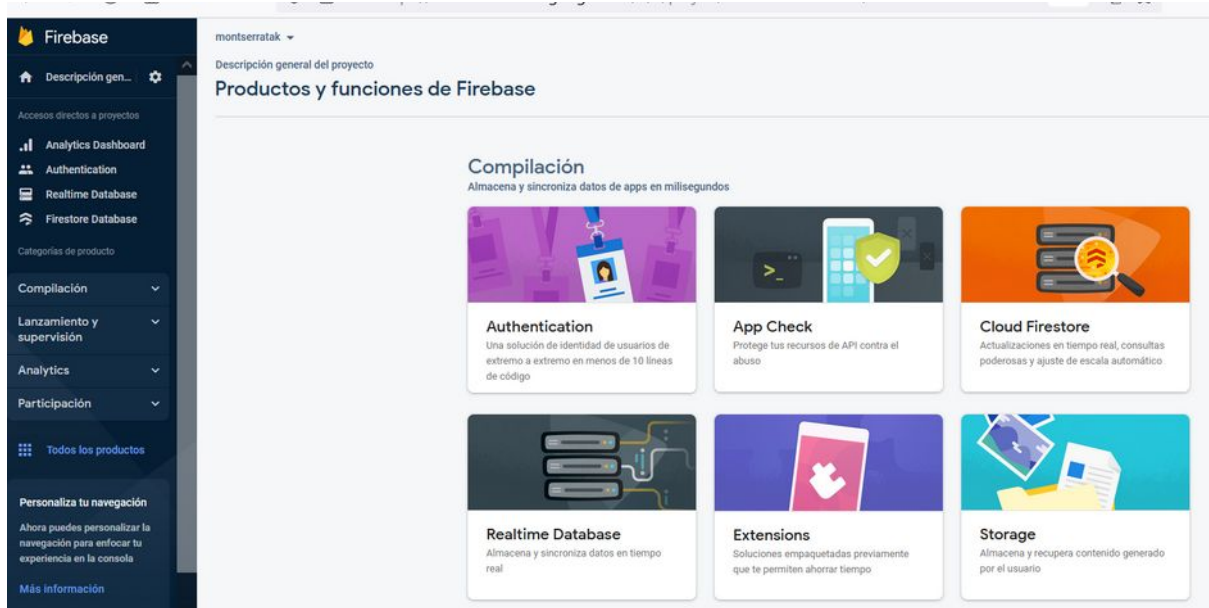


Anem a treballar amb el botó de Editar (canvi d'imatge)

Les imatges de tots els jugadors les guardarem ordenadament a un servei de Firebase que es diu Storage. Això ens servirà perquè quan llistem tots els jugadors poguem veure les seves imatges.

No perdem de vista que l'imatge que ha seleccionat el jugador de la galeria segueix estant en local, no al núvol.





## Selecionem Storage



Comenzar en modo de prueba

## Configura Cloud Storage

1 Reglas de seguridad para Cloud Storage — 2 Configura la ubicación de Cloud Storage

Después de definir la estructura de tus datos, **debes crear reglas para protegerlos**.  
[Más información](#)

**Iniciar en modo de producción**

Tus datos son privados de forma predeterminada. El acceso de lectura/escritura de los clientes solo se otorgará como se indica en tus reglas de seguridad.

**Comenzar en modo de prueba**

Para permitir una configuración rápida, los datos se abren de forma predeterminada. Sin embargo, debes actualizar las reglas de seguridad dentro de 30 días a fin de habilitar el acceso de lectura/escritura a largo plazo para los clientes.

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if
        request.time < timestamp.date(2022, 9, 12);
    }
  }
}
```

**1** Las reglas de seguridad predeterminadas del modo de prueba permiten que cualquier usuario con acceso a tu referencia de bucket de almacenamiento pueda ver, editar y borrar todos los datos que este contenga durante los siguientes 30 días.

Cancelar **Siguiente**

## Configura Cloud Storage

✓ Reglas de seguridad para Cloud Storage — 2 Configura la ubicación de Cloud Storage

El parámetro de configuración de la ubicación determina en qué lugar se almacenarán tu bucket de Cloud Storage predeterminado y sus datos.

**⚠ No podrás cambiar la ubicación después de configurarla. Este parámetro de configuración de la ubicación también será el predeterminado para Cloud Firestore.**

[Más información](#)

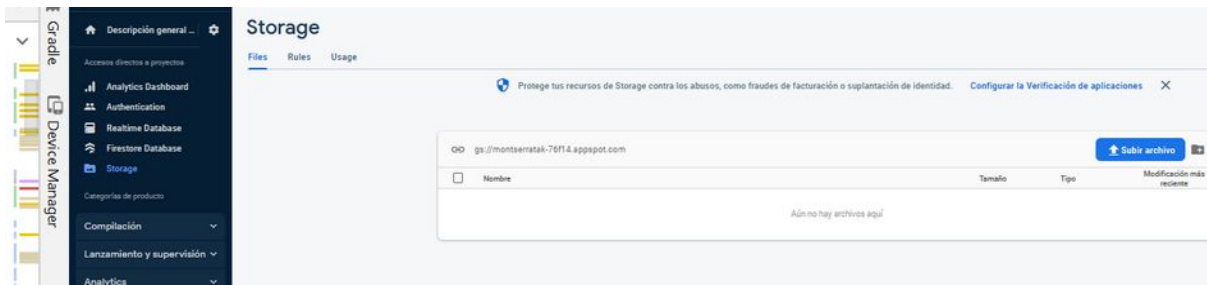
Ubicación de Cloud Storage

eur3 (europe-west) ▾

Los clientes del plan Blaze pueden elegir otras ubicaciones para los buckets adicionales.

Cancelar **Listo**

I ja ho tenim!



Ara mirem el tutorial

<https://firebase.google.com/docs/storage/android/start>

Per a aquesta part de la pràctica haurem de saber:

1. Fer un intent a la càmera o a la galeria i recuperar la imatge (tenint en compte els permisos de l'aplicació)

<https://handyopinion.com/pick-image-from-gallery-in-kotlin-android/>

Com demanar si l'usuari ha acceptat els permisos

<https://handyopinion.com/ask-runtime-permission-in-kotlin-android/>

2. pujar la imatge a storage, buscar-la i recuperar-la <https://www.youtube.com/watch?v=GmpD2DqQYVk>

Abans de fer res pujem el minlevel a 23 perquè sinó no ens funcionarà la acceptació de permisos (sinó hauríem de fer diferents mètodes segons la versió) Ho fem al gradle.

```

android {
    compileSdk 32

    defaultConfig {
        applicationId "com.neuLaworks.montserratak"
        minSdk 23
        targetSdk 32
    }
}

```

Ara hem de donar permisos al manifest perquè la aplicació pugui accedir a l'emmagatzematge intern

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.neulaworks.montserratak">

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

```



Afegirem una sèrie de mètodes per a gestionar els permisos

Els copiem com un bloc de <https://handyopinion.com/ask-runtime-permission-in-kotlin-android/>

```

//-----Permisos-----
fun isPermissionsAllowed(): Boolean {
    return if
    (ContextCompat.checkSelfPermission(this, android.Manifest.permission.
    READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
        false
    } else true
}

fun askForPermissions(): Boolean {
    val REQUEST_CODE=201
    if (!isPermissionsAllowed()) {
        if
    (ActivityCompat.shouldShowRequestPermissionRationale(this , android.M
    anifest.permission.READ_EXTERNAL_STORAGE) ) {
            showPermissionDeniedDialog()
        } else {
            ActivityCompat.requestPermissions(this
, arrayOf(android.
Manifest.permission.READ_EXTERNAL_STORAGE), REQUEST_CODE)
        }
        return false
    }
    return true
}

override fun onRequestPermissionsResult(requestCode:
Int, permissions: Array<String>, grantResults: IntArray) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
    val REQUEST_CODE=201
    when (requestCode) {
        REQUEST_CODE -> {
            if (grantResults.size > 0 && grantResults[0] ==

```

```

PackageManager.PERMISSION_GRANTED) {
    // permission is granted, you can perform your
operation here
    } else {
    // permission is denied, you can ask for
permission again, if you want
    // askForPermissions()
    }
    return
    }
    }
}

private fun showPermissionDeniedDialog() {
    AlertDialog.Builder(this)
        .setTitle("Permission Denied")
        .setMessage("Permission is denied, Please allow
permissions from App Settings.")
        .setPositiveButton("App Settings",
            DialogInterface.OnClickListener { dialogInterface, i -
>
                // send to app settings if permission is denied
permanently

                val intent = Intent()
                intent.action =
Settings.ACTION_APPLICATION_DETAILS_SETTINGS
                val uri = Uri.fromParts("package",
getPackageName(), null)
                intent.data = uri
                startActivity(intent)
            })
        .setNegativeButton("Cancel", null)
        .show()
    }
//-----

```

Ho copiem directament, podem personalitzar el missatge en anglès que demana el permís

Ara anem al listener del botó

```

editarBtn.setOnClickListener() {
    Toast.makeText(this, "EDITAR", Toast.LENGTH_SHORT).show()
    canviaLaImatge()
}

```

I cridem al procediment canviaLaImatge

Aquest procediment per ara fa poca cosa, llença un alertDialog on l'usuari pot escollir anar a la galeria o a la càmera, i dins de la galeria, per ara només mira si tenim els permisos necessaris.

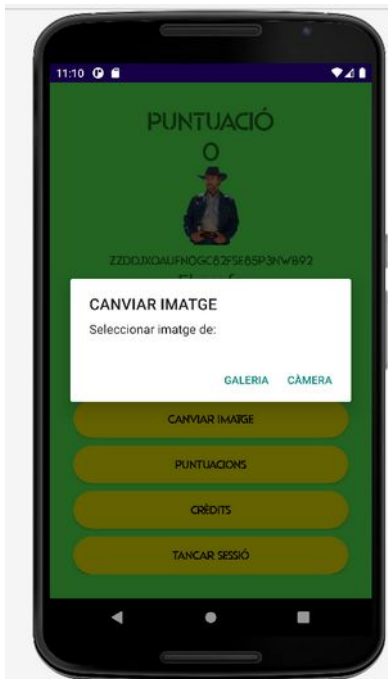
```

private fun canviaLaImatge() {
    //utilitzarem un alertDialog que seleccionara de galeria o agafar
    una foto
    // Si volem fer un AlertDialog amb més de dos elements (amb una
    llista),
    // Aixó ho farem amb fragments (que veurem més endavant)
    // Aquí hi ha un tutorial per veure com es fa:
    // https://www.codevscolor.com/android-kotlin-list-alert-dialog
    //Veiem com es crea un de dues opcions (habitualment acceptar o
    cancel·lar:

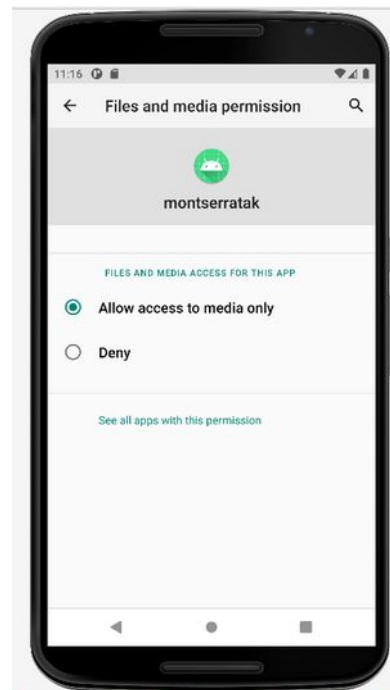
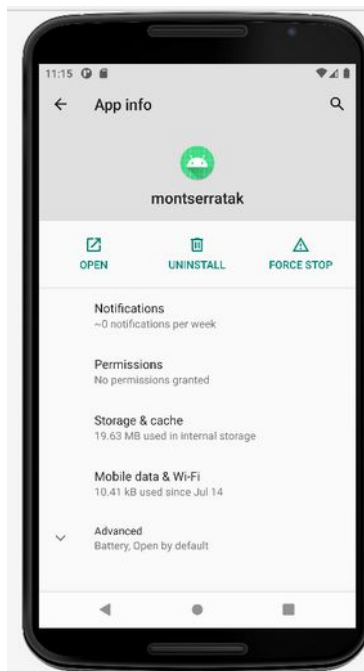
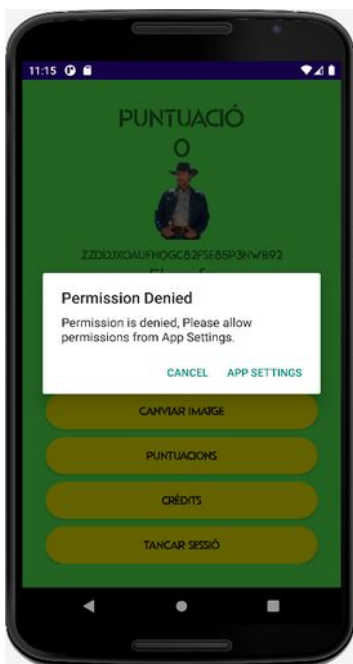
    val dialog = AlertDialog.Builder(this)
        .setTitle("CANVIAR IMATGE")
        .setMessage("Seleccionar imatge de: ")
        .setNegativeButton("Galeria") { view, _ ->
            Toast.makeText(this, "De galeria",
Toast.LENGTH_SHORT).show()
            //mirem primer si tenim permisos per a accedir a Read
External Storage
            if (askForPermissions()) {
                // Permissions are already granted, do your stuff
                // Aquí agafarem de la galeria la foto que ens calgui
            }
            else{
                Toast.makeText(this, "ERROR PERMISOS",
Toast.LENGTH_SHORT).show()
            }
        }
        .setPositiveButton("Càmera") { view, _ ->
            Toast.makeText(this, "A IMPLEMENTAR PELS ALUMNES",
Toast.LENGTH_LONG).show()
            view.dismiss()
        }
        .setCancelable(false)
        .create()
    dialog.show()
}

```

Provem la aplicació a veure si ens demana permisos per accedir a la llibreria



Si hem denegat la permisió a aquesta app surt aquest missatge



Ara que hem comprovat que tenim permisos, ja podem obrir la galeria. Ho fem amb un intent i un codi de tipus sencer que ens servirà per a identificar quin intent ha sigut.

```

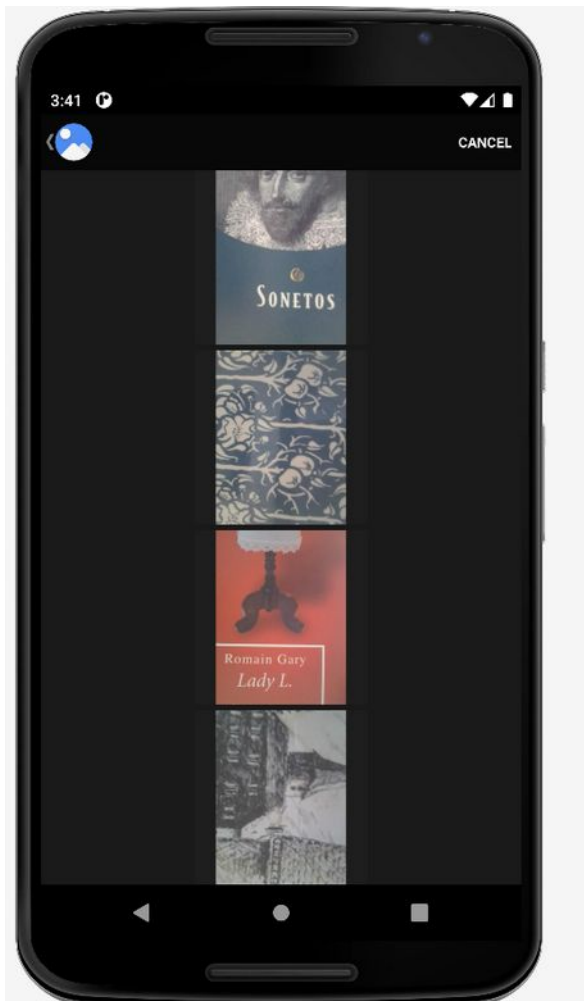
if (askForPermissions()) {
    // Permissions are already granted, do your stuff
    // Ara agafarem una imatge de la galeria
    val intent = Intent(Intent.ACTION_PICK)
    val REQUEST_CODE=201 //Aquest codi és un número que farem
servir per
  
```

```

// a identificar el que hem recuperat
del intent
// pot ser qualsevol número
intent.type = "image/*"
startActivityForResult(intent, REQUEST_CODE)
}

```

Provem l'aplicació i veiem que efectivament s'obre la galeria i ens deixa escollir una de les imatges



Ara hem de reescriure `onActivityResult` que és on ens retornarà la imatge el intent que hem llençat

Necessitarem abans de tot tenir una variable que ens recuperi la Uri

```
lateinit var imatgeUri: Uri
```

Ara sí, ja podem sobrescriure `onActivityResult`

```

override fun onActivityResult(requestCode: Int, resultCode: Int,
data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
}

```



```

val REQUEST_CODE=201
if (requestCode==REQUEST_CODE && resultCode== RESULT_OK){
    imatgeUri=data?.data!!
    imatgePerfil.setImageURI (imatgeUri)
}
else{
    Toast.makeText(this, "Error recuperant imatge de galeria",
Toast.LENGTH_SHORT).show()
}

```

Si el codi és el mateix que hem enviat i el resultat és Ok, llavors podem recuperar amb data la Uri (una Uri és un identificador de recursos, és quelcom similar a una Uri) i assignar-la a la imatge de perfil



Ara farem servir Storage de Firebase per a guardar les imatges

Primerament afegirem la llibreria Storage a les dependències de gradle

implementation 'com.google.firebase:firebase-storage-ktx:20.0.1' (actualment no sé si és necessari incloure aquesta dependència)

```

//base de dades
//implementation 'com.google.firebase:firebase-database-ktx'
implementation 'com.google.firebase:firebase-database-ktx:20.0.5'
implementation 'com.google.firebase:firebase-storage-ktx:20.0.1'

```

A continuació preparem un mètode que guardi la imatge

```

if (requestCode==REQUEST_CODE && resultCode== RESULT_OK) {
    imatgeUri=data?.data!!
    imatgePerfil.setImageURI (imatgeUri)
    pujarFoto (imatgeUri)
}

```

Primerament però necessitem unes variables

```

// Variables per a gravar a Storage
lateinit var storageReference: StorageReference
lateinit var imatgeUri: Uri

```

La StorageReference la inicialitzarem al onCreate

```

//Inicialitza el StorageReference
storageReference = FirebaseStorage.getInstance().getReference()

```

Ara fem el mètode de pujar la foto

Crearem a dins de Storage un child (o una carpeta) anomenada FotosPerfil

Dins d'aquesta carpeta hi trobarem tot de carpetes, una per a cada jugador. Aquest segon nivell s'identifica per la Uid del jugador

Per a pujar la foto a Storage copiem descaradament el procediment que ens recomanen a la documentació de Fitrebase

```

private fun pujarFoto (imatgeUri: Uri){
    var folderReference: StorageReference =
storageReference.child("FotosPerfil")
    var Uids: String = uid.getText().toString()

    //Podriem fer:
    //folderReference.child(Uids).putFile(imatgeUri)
    //Pero utilitzem el mètode recomanat a la documentació
    // https://firebase.google.com/docs/storage/android/upload-
files

    // Get the data from an ImageView as bytes
    imatgePerfil.isDrawingCacheEnabled = true

```

```

imatgePerfil.buildDrawingCache()
val bitmap = (imatgePerfil.drawable as BitmapDrawable).bitmap
val baos = ByteArrayOutputStream()
bitmap.compress(Bitmap.CompressFormat.JPEG, 100, baos)
val data = baos.toByteArray()

var uploadTask = folderReference.child(Uids).putBytes(data)
uploadTask.addOnFailureListener {
    // Handle unsuccessful uploads
    Toast.makeText(this, "Error enviand imatge a Storage",
Toast.LENGTH_SHORT).show()

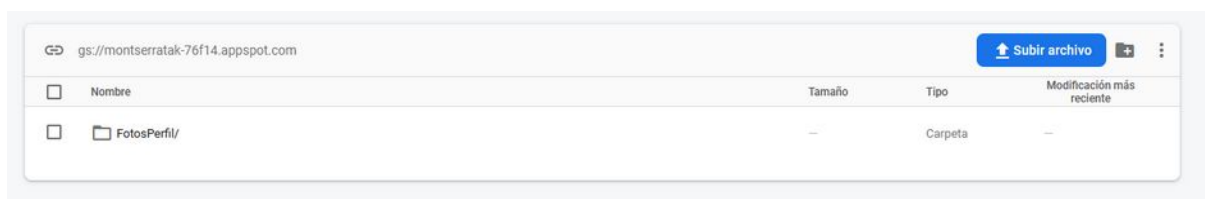
}.addOnSuccessListener { taskSnapshot ->
    // taskSnapshot.metadata contains file metadata such as size,
content-type, etc.
    // ...
}

```

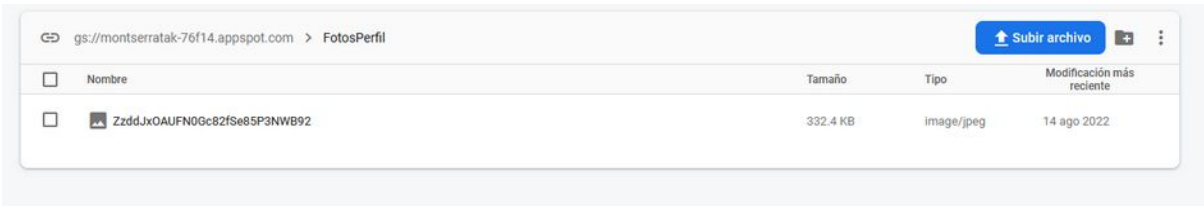
Anem a provar-ho



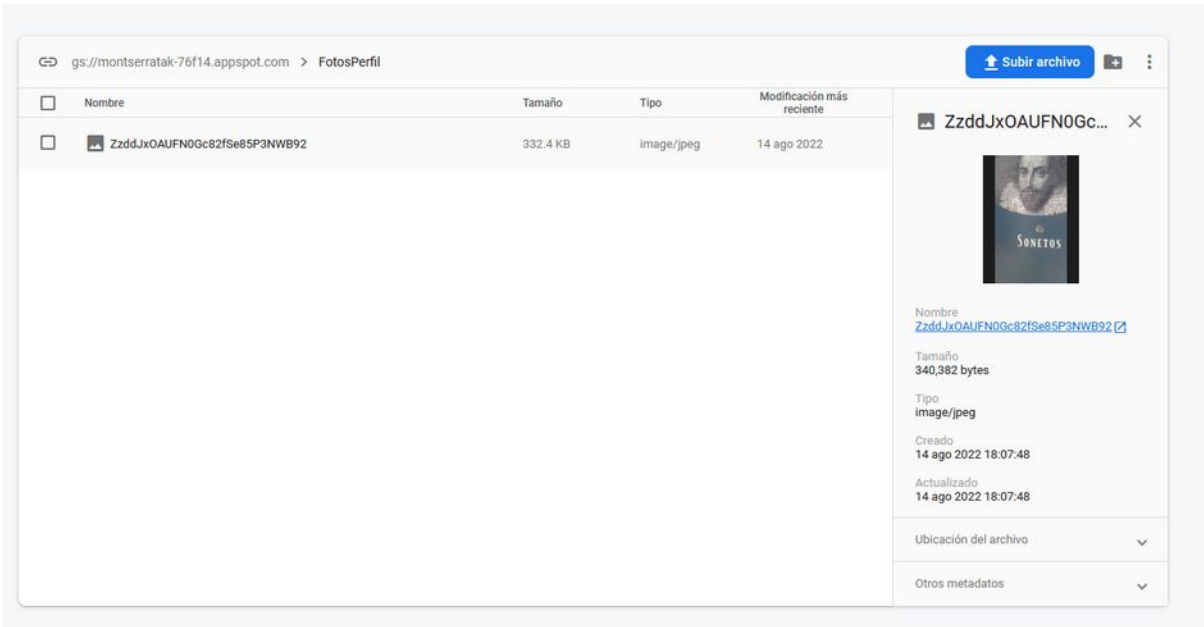
Si anem a firebase



I si obrim FotosPerfil



Tenim una carpeta amb el identificador de l'usuari  
I si piquem a sobre podem fins i tot veure la foto



Tenir la foto de l'usuari a storage ens servirà pel següent capítol, quan llistarem tots els jugadors i voldrem veure les seves fotos, els seus noms i les seves puntuacions.

**Heu d'implementar com agafar una imatge de la càmera del mòbil**

## Fase 9: Llistar jugadors

Volem veure un llistat de jugadors. De cada un d'ells hem de recuperar: nom, puntuació i una foto que hem desat a Storage

La foto l'agafarem de Firebase Storage: <https://firebase.google.com/docs/storage>

Per descarregar de Storage, ens recomana

```
var islandRef = storageRef.child("images/island.jpg")

val ONE_MEGABYTE: Long = 1024 * 1024
islandRef.getBytes(ONE_MEGABYTE).addOnSuccessListener {
    // Data for "images/island.jpg" is returned, use this as needed
}.addOnFailureListener {
    // Handle any errors
}
```

<https://firebase.google.com/docs/storage/android/download-files>

Utilitzarem un recyclerView per mostrar jugadors

<https://www.youtube.com/watch?v=k3zoVAMuW5w>

Podem veure un programa similar a:

<https://www.youtube.com/watch?v=hSKbpaKq0TU&list=PLhcYacorV7U6HHVsfXnWodwEPI0E38BXD&index=20>

I la continuació:

<https://www.youtube.com/watch?v=e-kzTv9FGo8&list=PLhcYacorV7U6HHVsfXnWodwEPI0E38BXD&index=21>

## Fase 10: Pantalla de Crèdits

Crearem una pantalla de crèdits utilitzant Fragments.

Veiem un vídeo sobre la utilització de Fragments

<https://www.develou.com/android-aplicaciones-fragmento/>

La pantalla de crèdits ha de carregar dos fragments, 1 d'ells ha de tenir el logo del centre, amb el nom del centre i el nom del cicle (Grau Superior de ....) i l'altre fragment ha de mostrar el nom dels alumnes que heu fet el programa, (i unes fotos vostres si voleu) i la data de finalització.

Un temporitzat fa que es mostri un fragment després de 2 o 3 segons es mostri l'altre

Hi ha un botó que retorna a la pantalla menú

Ens faltará finalment afegir a la pantalla menú una opció per a canviar la contrasenya del jugador

I ja haurem acabat el joc, ens faltará crear la apk i llestos

**ENHORABONA!!!!**