APP DEVELOPMENT FRAMEWORKS (L5)

PROGRAMACIÓN MOBILE

Profesor: ben Blanco
Año: 2020/2021
Alumno: Alejandro Sanchez Gimeno.

Index

1.- App Project

1.1.- Introduction

This mobile app concept is based on the Destiny 2 companion app. It's a project with educational purposes, all the images and references will be linked on the reference section.

1.2.- Characteristics

The user will be able to login and sign in via email and password. If the user doesn't have an existing account they can sign up with a brand new one.

Users will see a list of characters with their attributes: class, race, power and level. The users can sort this list based on the power of characters.

If the user clicks on any character, a new layout will pop up and show the equipment. The equipment is divided on primary, special and heavy weapons, and helmet armor. Each weapon has its stats: power and elemental damage (kinetic, solar, arc and void). Armor has power and armor type. Users can sort these items based on their power.

1.3.- Class diagram

This project it's pretty simple so there aren't many classes. The main ones are:

1. Character: main class that stores the data from the JSON. It has the power(Int) , class(String), race(String), level(String), url image(String), three array of weapon classes and one array of armor class. This class is parcelable in order to pass its data to the fragments.

2. Armor: Class for the armor data. Contains the power(int) and the url image(string). parcelable as well.

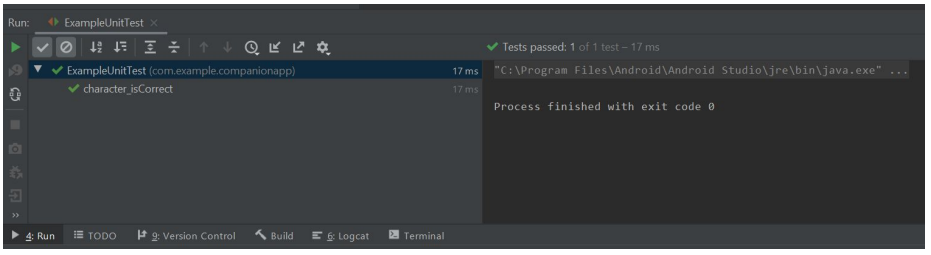
3. Weapon: Class for the weapon data. Contains the power)int), the url image(string) and the elemental damage(int).

All these classes have their respective adapters to create the views.

1.4.- Tests

The app has been tested with a basic unit test, in order to check possible nulls or future erros.

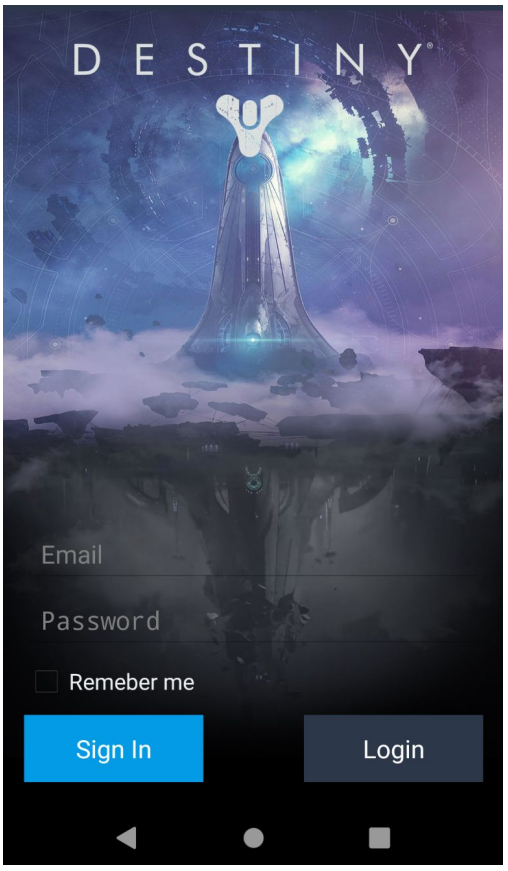
```
package com.example.companionapp

import com.example.companionapp.Classes.Armor
import com.example.companionapp.Classes.Character
import com.example.companionapp.Classes.Weapon
import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun character_isCorrect() {

        val primary = Array( size: 1) {Weapon( power: 1920, image: "URL", type: 1)}
        val special = Array( size: 1) {Weapon( power: 1920, image: "URL", type: 1)}
        val heavy = Array( size: 1) {Weapon( power: 1920, image: "URL", type: 1)}
        val helmet = Array( size: 1) {Armor( power: 1920, image: "URL")}

        var character = Character( power: 1942, characterclass: "Titan", race: "Exo", level: "level 30", image: "URL",primary,special,heavy,helmet)

        assertEquals( expected: 1942, character.power)
        assertEquals( expected: "Titan", character.characterclass)
        assertEquals( expected: "Exo", character.race)
        assertEquals( expected: "level 30", character.level)
        assertEquals( expected: "URL", character.image)

        assertEquals( expected: 1920, character.primaryweapon?.get(0)?.power)
        assertEquals( expected: "URL", character.primaryweapon?.get(0)?.image)
        assertEquals( expected: 1, character.primaryweapon?.get(0)?.type)

        assertEquals( expected: 1920, character.specialweapon?.get(0)?.power)
        assertEquals( expected: "URL", character.specialweapon?.get(0)?.image)
        assertEquals( expected: 1, character.specialweapon?.get(0)?.type)

        assertEquals( expected: 1920, character.heavyweapon?.get(0)?.power)
        assertEquals( expected: "URL", character.heavyweapon?.get(0)?.image)
        assertEquals( expected: 1, character.heavyweapon?.get(0)?.type)

        assertEquals( expected: 1920, character.helmet?.get(0)?.power)
        assertEquals( expected: "URL", character.helmet?.get(0)?.image)
    }
}
```



2.- UI/UX Flow Chart

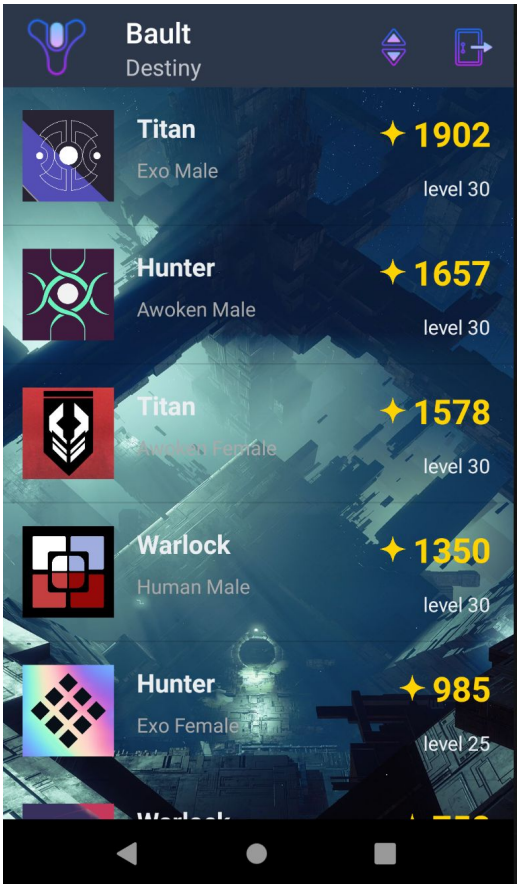
2.1.- Flow Chart

If it's the first time the user opens the app, the login activity is the first thing they will see. They can sign up with an email and password, or login with an already existing account. They can also check the remember me box to keep logged to the app. If the authentication fails, a message will appear to notify the user, if it’s successful the app continues.



Once the user is logged, they will see the main characters screen. It’s composed by a toolbar with two buttons, the logout button and the sort button

The logout button will send the user to the login screen and the sort button will sort the character list based on power



The user can scroll up and down to navigate and click any character they want. once the user clicks one character, the equipment for this character will be shown.

On this screen the user can scroll up and down to see the equipment and its statistics. They can also sort all categories by power clicking on the sort button mentioned above.

If the user wants to go back to the character list, they only need to press the back button or slide on configurations with no capacitive buttons.

2.2.- Development problems and solutions

The app receives all the data from an online JSON, it has an array of characters with all its data. I use the picasso library to load the images. Once all data is stored on the main activity, the first problem I encountered was to pass all this information to the fragments.

The main activity has a frame layout, one fragment for the character list and another one for the equipment, so i needed to pass that data in order to show it on the fragments.

Initially I thought of using a bundle, I discovered that bundles can only hold int, bool, string, ect… and parcelables. So I came up with the idea of making the data class parcelable. That solved the problem.

Another problem was to maintain the data once the application closed. Given that my application needs the internet in order to get the data from the JSON (including images) I decided that the app will always require the internet in order to work and I implemented the remember me function to prevent the user from logging in every time they open the app. I used the shared preferences to save the login and access directly to the main activity if the user is already logged.

3.- User Manual

3.1.- Authentication

Here the user can sign in with a new account or log in with an existing one. With either of these options the user can choose if they want to keep the session open or not.

3.2.- Characters

Here the user can see all the characters and its details, scroll up and down, sort the list and log out to return to the login screen. Also go to the equipment screen just by clicking on any character.

3.3.- Equipment

The user is able to see all the weapons and armors for a given character, scroll up and down, sort all the items , log out and return to the character list by clicking the back button.

4.- References

All the weapons, armor and emblems images are from:

https://d2.destinygamewiki.com/wiki/Destiny_2_Wiki

ToolBar Icons:

https://icons8.com/