ESAT INNOVATION SCHOOL | ESCUELA SUPERIOR DE ARTE Y TECNOLOGÍA

Mobile 2020-2021

KIBO

Name: Mobile programme
Professor: Rubén Blanco
Course: 2020/2021
Author: Alexandre Calabuig Langa

INDEX

1.- INTRODUCTION

The APP developed is called KIBO, which is a companion application that runs on Android devices and allows the user to access with an email and password, in order to check their ranking, game history and character items that theoretically would have a user who would use the application frequently.

The objective of the application is the retention of users of the hypothetical game, extending its market to mobile platforms.

Imagen 1: Splash screen with rotative logo

2.- FEATURES

2.1 Estructure

At the code level, each section is a predetermined activity, and in some of them, a container is added where a FragmentList is introduced, in which the content of the desired section can be visualized.

The project has had to access JSON located on the Internet, therefore, classes have had to be made to be able to handle the data obtained from JSON.

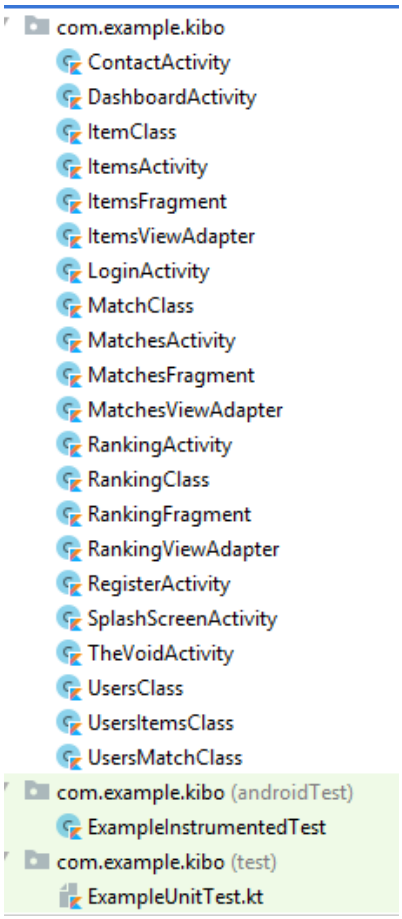


Imagen 2: Clases and Activity

2.2 Development patterns

At the beginning of the development, tests have been made to know how to collect JSON data, either via local files or via files located on the Internet, as well as other types of actions such as the use of fragment lists and how they can be filled in.

Once it is clear how all the things are going to be done, we start designing what the JSON class is going to be and what elements are going to be collected from them, in order to create them and upload them to an internet repository.

Then, you create the activities that will be necessary and start working by the user login. It is important to start here because the user's data will be needed to access the data during the application.

Once you start programming, you separate into functions what you want to pack, such as the functions of changing the background, or the creation of a Thread to access data online.

Afterwards, the Activities have been made in which Internet access and data collection through threads are not necessary, to have those parts already finished and working from one Activity to another.

Once the most basic activities have been completed, the activities with Internet access and JSON data collection are completed one by one.

By doing all these activities in a row, they become faster than doing them separately, since by getting used to knowing what kind of arrays to touch and what parameters to access, this part, which can be a bit messy, is done more easily.

When all the functionality is finished, all the code and design is reviewed, adjusting the possible problems and commenting everything properly (at least to the necessary functions) so that any developer who has to access the code, knows what each element is.

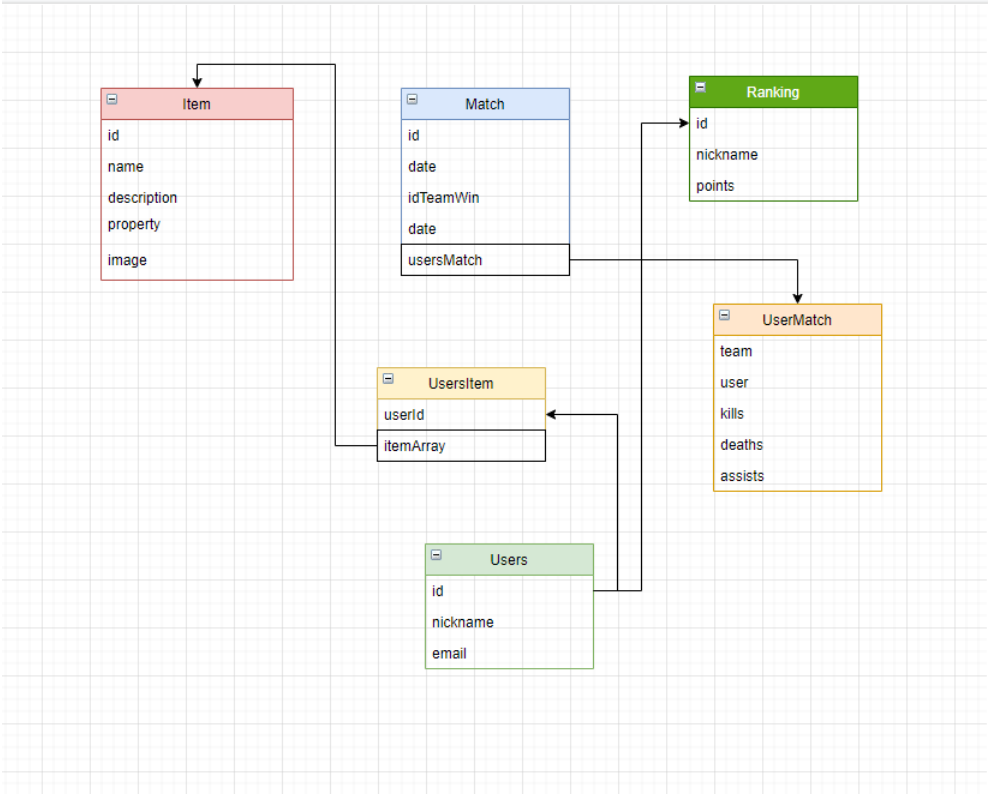
2.3 Tests

Tests have been carried out in the Unit Test, mainly on access to JSONs as mentioned above. To demonstrate that the classes are correctly formatted, as well as the JSON, they are placed at a specific address in the array to demonstrate that the information being sought is that which has been sought from the first moment.

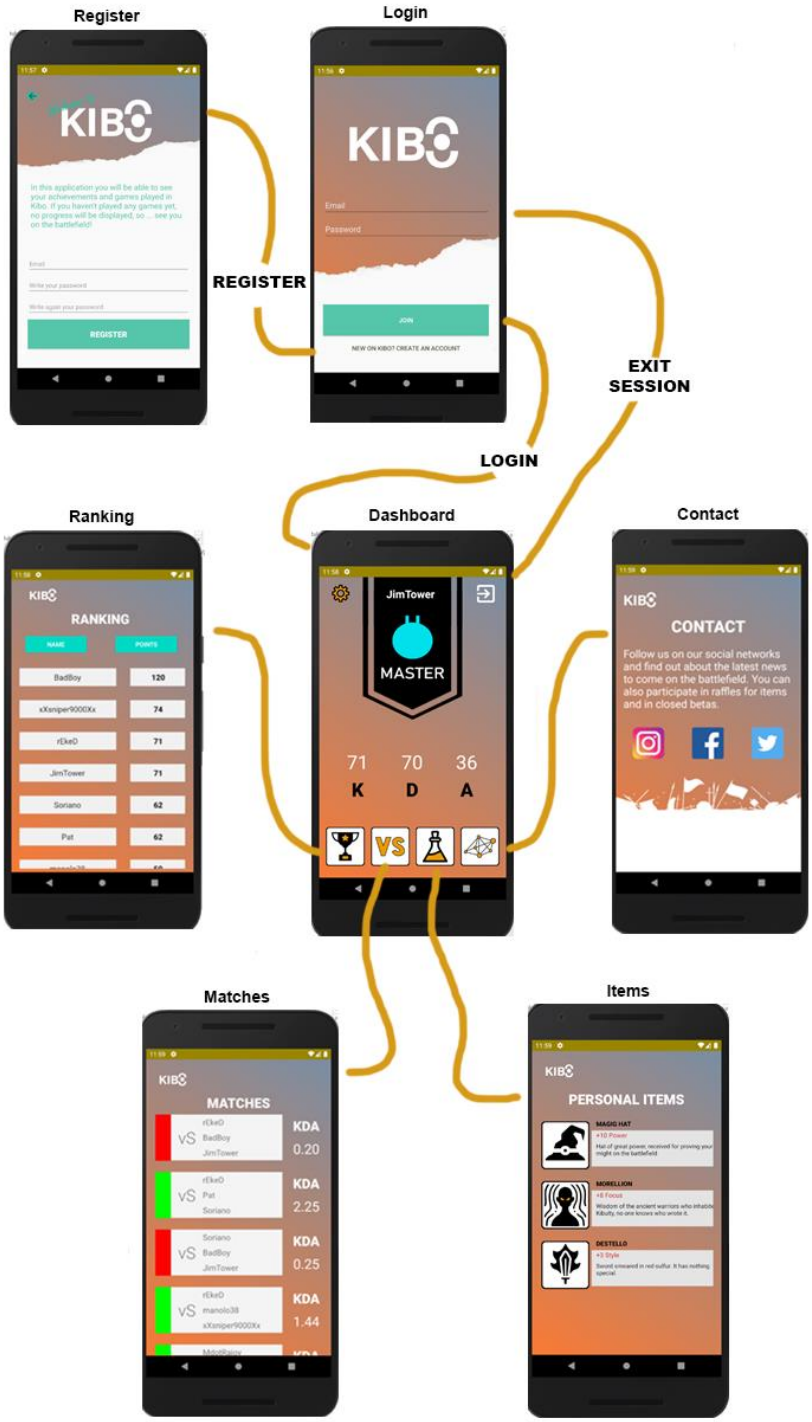
```
var userItemsArray: ArrayList<UsersItemsClass> = ArrayList()
var itemsFromUser: ArrayList<ItemClass> = arrayListOf()
@Test
fun getId_isCorrect() {

    var userArray = ("[{'id': 0,'nickname': 'rEkeD', 'email': 'reked@esat.es'}, "
        + "{'id': 1,'nickname': 'Soriano','email': 'soriano@esat.es'}, "
        + "{'id': 2,'nickname': 'BadBoy','email': 'badboy@esat.es'}]")

    val userListType: Type = object : TypeToken<ArrayList<UserClass?>?>() {}.type
    val usy: ArrayList<UserClass> = Gson().fromJson(userArray, userListType)

    assertEquals( expected: 0, usy[0].id)
}

@Test
fun getNickname_isCorrect(){
    var userArray = ("[{'id': 0,'nickname': 'rEkeD', 'email': 'reked@esat.es'}, "
            + "{'id': 1,'nickname': 'Soriano','email': 'soriano@esat.es'}, "
            + "{'id': 2,'nickname': 'BadBoy','email': 'badboy@esat.es'}]")

    val userListType: Type = object : TypeToken<ArrayList<UserClass?>?>() {}.type
    val usy: ArrayList<UserClass> = Gson().fromJson(userArray, userListType)

    assertEquals( expected: "Soriano",usy[1].nickname)
}
```

Imagen 3: Test example

2.4 Class diagram



3.- UI/UX FLOWCHART



4.- ABOUT THE DEVELOPMENT

4.1 Problems during practice

During the course of the practice, there have been problems on several occasions:

- The collection of internet data by JSON: In order to use internet data, you have to use a Thread to access that data, so you tried to access that data within the Fragment created for each section of the application. But you can't put a Thread as usual inside the fragment, and when you tried, there were errors and failures.

- Creating a menu for the selection of the background: In the application, the user can choose the color of the background, so it was thought to make a drop-down menu that shows the available options, but after several attempts, the menu did not go as expected.

4.2 Problem solving

For the above problems the following solutions have been made:

- The collection of internet data by JSON: As I had problems doing it from a fragment, I discovered that the fragment can be passed variables and arrays by parameter, so the data collection is done in the Activity, and once processed the necessary data that I need in the fragment, I pass those data through a commit in the container introduced in the Activity. By passing these data, you can also pass to the RecyclerView, so we can show the data that are needed.

- Creation of a menu for the selection of the background: As the creation of the menu did not come out correctly, I came up with a better idea; that it did not have a menu. Simply pressing the option button once changes the color. Thus, it is not necessary to enter a menu and you can quickly click on that button, so finally as there is no drop-down menu, the use of that function is faster and with less code to perform.

4.3 Possible improvements

As the code has progressed, it has been seen that at the end there are things that could have been improved, such as collecting all the necessary data at the beginning of the application and saving it locally until the logout, instead of loading it every time you enter the section that is needed. You could have done that so that it would only load once and the application would be fast the whole time.

You could also have put the background code in each activity if you had put it in a separate class and only needed to access it through a function call.

At the structure or functionality level, in the items section you could have put a second item view with all the items that exist, so that the user knows how many there are and how many he has.

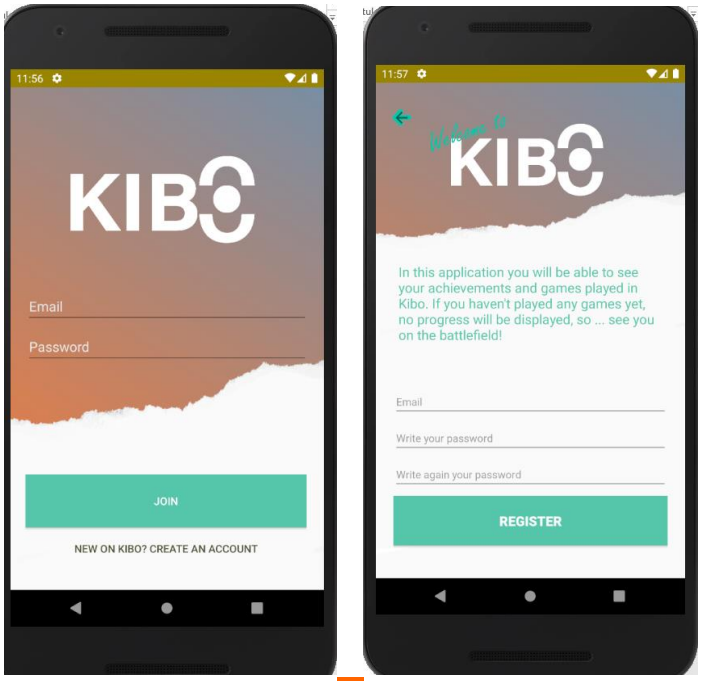
I also would have liked to improve the Contact section, so that it was more interactive and had more content, and also that it could be re-stamped and already access to some predetermined data and load the new data into a JSON.

5.- USER MANUAL

Hay dos formas de poder ejecutar la aplicación:

- Si se quiere instalar el APK en el teléfono móvil, vaya a la sección APP→APK y coja el archivo para descargarlo en el dispositivo móvil. Una vez tengas el archivo en el dispositivo, podrás ejecutarlo e instalarlo.
- Si se desea ejecutar desde Android Studio, abra la aplicación de Android y desde “Open”, seleccione el archivo de ejecución de APP→CODE. Una vez abierto, espere a que se haga la build y una vez finalizada, en la parte superior derecha se hace clic en el botón de “Play” para que se abra un dispositivo virtual, y ya podrá disfrutar de la aplicación.

5.1 Demo use

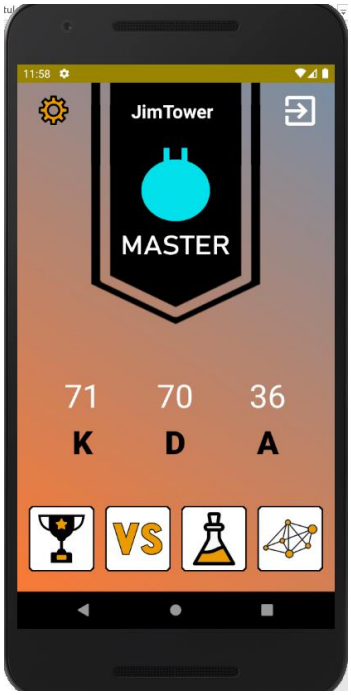


At the beginning of the application you will be able to login or register. As the registration would currently lead to a screen that tells you that you would have to have games played, you will be logged in with one of the following emails and passwords:

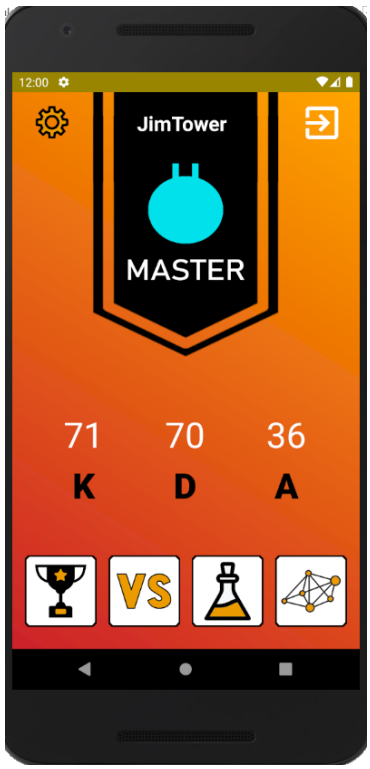
reked@esat.es
sniper@esat.es
emedot@esat.es
jimtower@esat.es
badboy@esat.es

rita@esat.es
manolo38@esat.es
pat@esat.es
soriano@esat.es

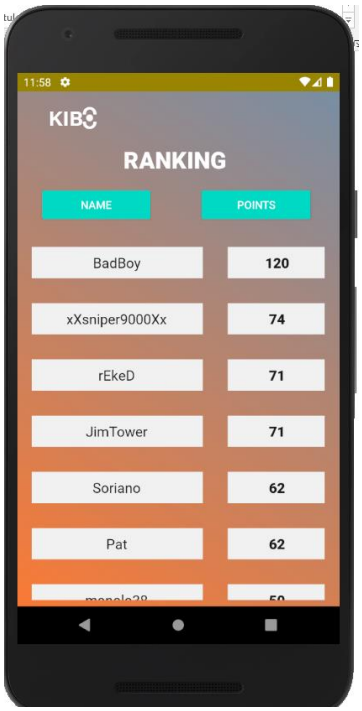
Password of all the acounts: 123456



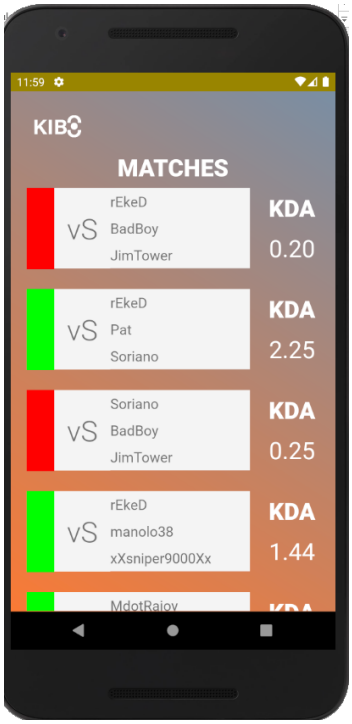
This is the dashboard in which you can see the total number of deaths and assists, besides being able to change the background color to the user's liking.



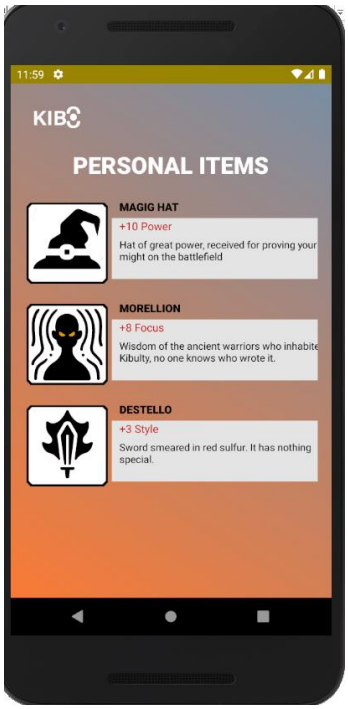
At the top left you can change the color by clicking once, and at the top right you will log out and return to the login. At the bottom you can access the Ranking, Game History, Items and Contact.



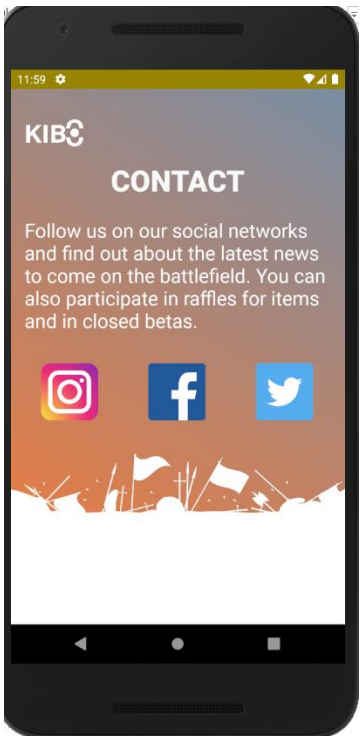
In the ranking you will see the points of the users, and you can organize it by points or by username.



In the match history, you can see your results, who you have played against and the KDA.



In the section of personal items you will be able to see the player's own items, with their description and attributes.



Finally, in the contact section you can access the developer's acounts, if you click on the images, it will take you out of the application.