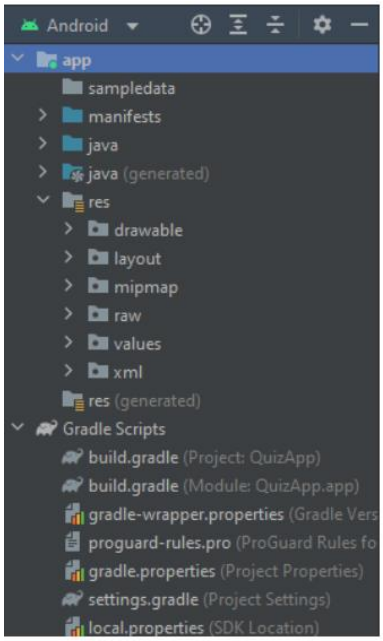
School of Mathematical and Computational Sciences
Career of Computer Science

Subject: Software Engineering

Content: Kiwcha Yachay mobile application code documentation

Members: - Wilter Jose Menoscal Saltos.
- Ricardo Joseph Farinango Romero.
- Christopher Hernan Almachi Loor.
- Emilio Ibañez.
- Anthony Salazar.

To begin with, it is very important to mention that our application was developed on the Android Studio platform, together with java. It has a log of activities performed, in addition to a unit with 3 lessons attached, where, each lesson has 3 activities. The activities vary among them, because each lesson is about a specific topic. When talking about the application code it is important to understand the following:



Our application has the following folders:

- Manifests.
- Java.
- Res.
- Gradle Scripts

Manifests

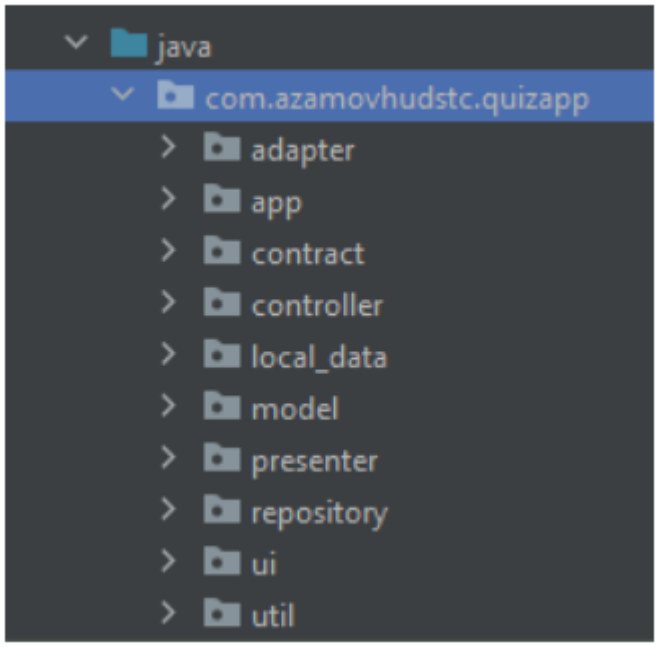
Where "manifest" refers to all the scripts that we are going to implement in our code. A brief example is shown in the following figure:

```
android:supportsRtl="true"
android:theme="@style/Theme.QuizApp"
android:usesCleartextTraffic="true"
tools:targetApi="31">
<activity
    android:name=".ui.page.act3_le1"
    android:exported="false" />
<activity android:name=".ui.Act2_le1" />
```

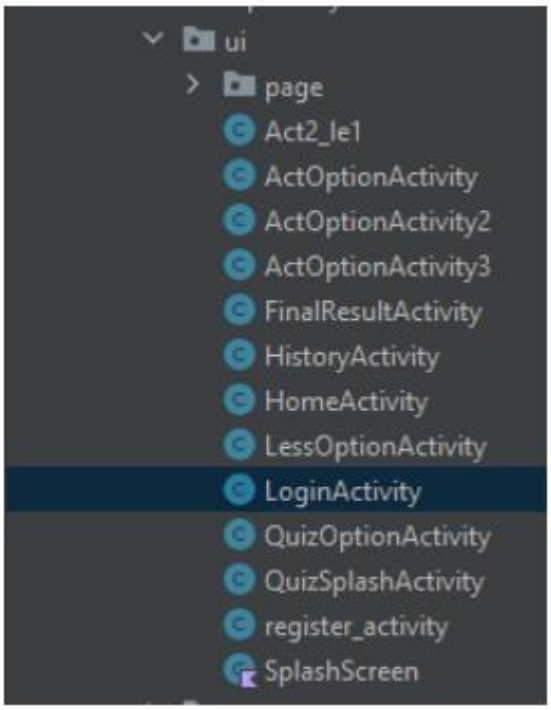
It can be observed that we must declare the new activity that we create inside the "manifest" in order that this is compiled with all the files of the program. "manifest in order for it to be compiled with all the files of the program, in order for us to create another .java file to we create another .java file to define the correct functioning of a created interface, we must declare it here. created, we must declare it here.

Java

In the java "folder" we declare all the drivers, adapters, among other items (scripts) that are very necessary to connect the frontend to the frontend. that are very necessary to connect the front-end with the back-end. This means that, in this folder we must declare the actions of each button, or image view that we have in our application. our application.



Where, the "ui" folder stores all the interface drivers of our application.



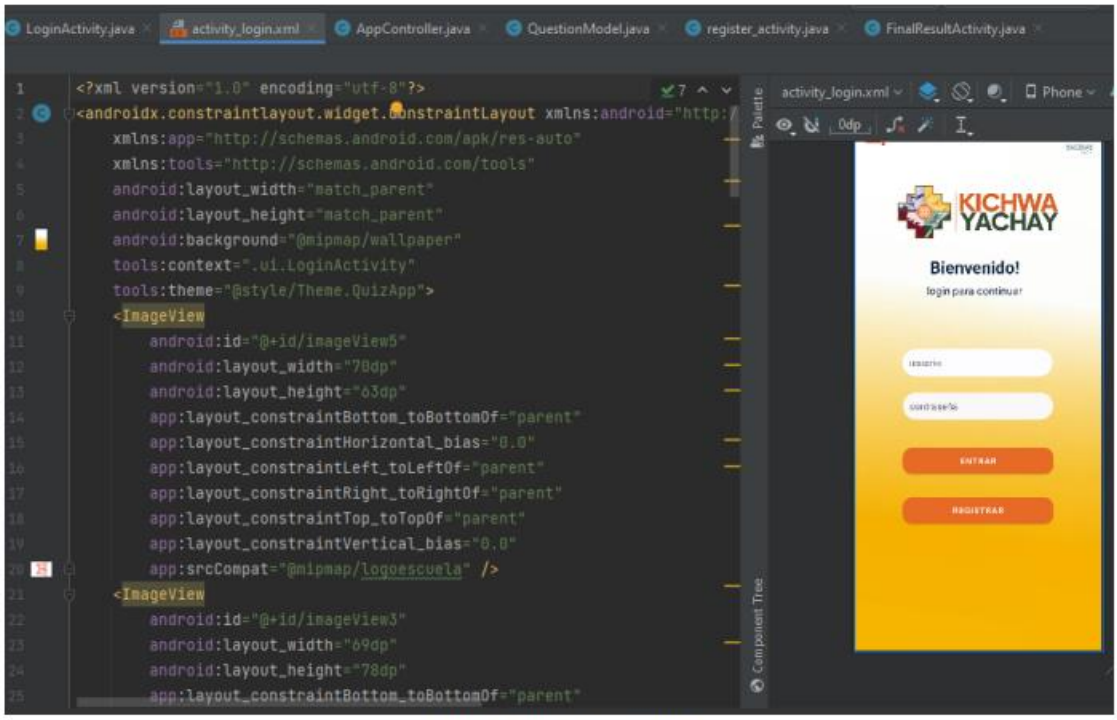
Where, using the "Login Activity" file as an example, we have to, first of all, declare the interface buttons, to give an event to these buttons (actions), then it is very important to declare which layout we are using, in this case it is the "activity_login", this is to give actions to give actions to this interface previously created. Then the function "Validate User" is declared in order to establish connection to the database service previously created. Finally, the function "validate user" has a syntax in order to establish connection and receive a response from the server, to know if the connection was established correctly.

```java
package com.azamovhudstc.quizapp.ui;

import ...


public class LoginActivity extends AppCompatActivity {
    Button registrar;
    EditText edtUser,edtpass;
    Button btnlog;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        edtUser=findViewById(R.id.user);
        edtpass=findViewById(R.id.password);
        btnlog = findViewById(R.id.btnLogin);
        btnlog.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                validarUsuario( URL: "http://192.168.174.33:2001/validar_usuario.php");}
                //Intent intent = new Intent(getApplicationContext(), HomeActivity.class);
                //startActivity(intent);
        });
```

Login Activity



Activity_login(Layout)

Then, inside the folders "model" and "controller" we find files that have to do with the questions and answers of the different activities, as a brief example we will use the files "app_controller" and "question model". files "app_controller" and "question model".

LoginActivity.java | AppController.java | QuestionModel.java | register_activity.java | FinalResultActivity.java

```java
package com.azamovhudstc.quizapp.model;

public class QuestionModel {
    private Integer txtQuestion;//Qozonda emas Qaynaydi qishin yozin tinmaydi !
    private String answer;//Bulog
    private Integer speaaker;
    private String variant;//baulgdqos

    public QuestionModel(int txtQuestion, int pista, String answer, String variant) {
        this.txtQuestion = txtQuestion;
        this.answer = answer;
        this.speaaker=pista;
        this.variant = variant;
    }

    public Integer getTxtQuestion() { return txtQuestion; }

    public Integer getsound() { return speaaker; }

    public String getAnswer() { return answer; }

    public String getVariant() { return variant; }
}
```

Question model

```java
package com.azamovhudstc.quizapp.controller;

import ...

public class AppController {
    private List<QuestionModel> questionModelList;
    private int level = 0;
    MediaPlayer mp;
    public AppController() { this.questionModelList = new ArrayList<>(); }
    public void loadQuestion() {
//        questionModelList.add(new QuestionModel(R.drawable.uzbekistan, "", "daspeyodszea"));
        questionModelList.add(new QuestionModel(R.drawable.ques1,R.raw.pista1, answer: "es negro", variant: "emrngeoksn s"));
        questionModelList.add(new QuestionModel(R.drawable.ques2,R.raw.pista2, answer: "es rojo", variant: "xoeqjsao yri"));
        questionModelList.add(new QuestionModel(R.drawable.ques3,R.raw.pista3, answer: "es blanco", variant: "oncarb slaed"));

        shuffle();
    }

    public void shuffle() { Collections.shuffle(questionModelList); }

    public int getQuestionCount() { return questionModelList.size(); }

    public QuestionModel getQuestion(int level) { return questionModelList.get(level); }
}
```

App controller

Where, in question model the design of the question is defined, where this one has its respective question, correct answer, audio and the variant of the original answer. Besides having the public classes in order to return the final values to the manipulation script of our interface. In the App Controller file, basically we use the design previously created in our Question Model, and we add the corresponding question with its answer, audio and variant of the answer. It is important to mention that these controllers correspond to activity 2 of lesson 1.

RES

Finally we will have the "res" folder, where here we will have all the "layout" of our application, this means the interface of our application, more dedicated to the front-end. Besides having folders where will be stored the images used in our application, the audio files, customization of certain buttons and arrows.

Gradle Scripts

This folder is very important, because here we will import all the necessary libraries for the correct operation of our application (Do not delete it).