Ingeniería del Software.

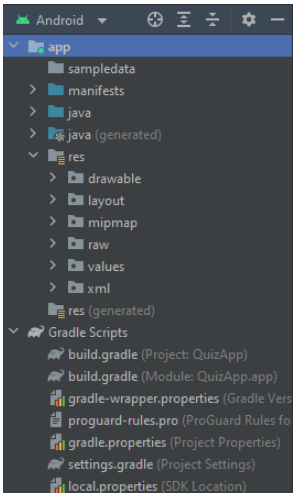
Documentación del código de la Aplicación móvil Kiwcha Yachay

Integrantes:

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

Para empezar, es muy importante mencionar que nuestra aplicación fue desarrollada sobre la plataforma de Android Studio, junto con java. Cuenta con un registro de actividades realizadas, además de una unidad con 3 lecciones adjuntas, donde, cada lección cuenta con 3 actividades. Las actividades varian entre sí, debido a que cada lección es sobre un tema en especifico.

Al momento de hablar del código de la aplicación es importante entender lo siguiente:



Nuestra aplicación cuenta con las siguientes carpetas:

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

Manifests

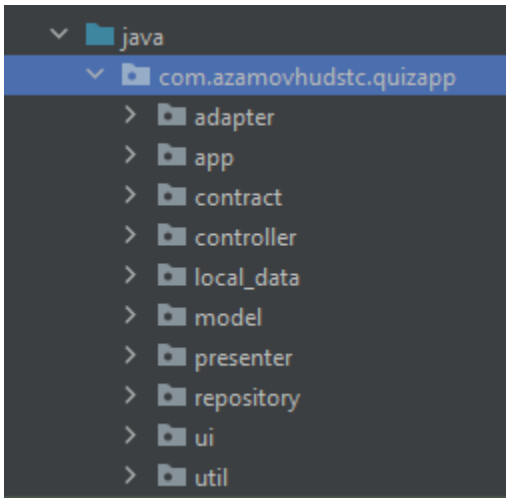
Donde “manifest” hace referencia a todos los scripts que nosotros vayamos a implementar en nuestro código. Un breve ejemplo se muestra en la siguiente figura:

```
        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" />
```

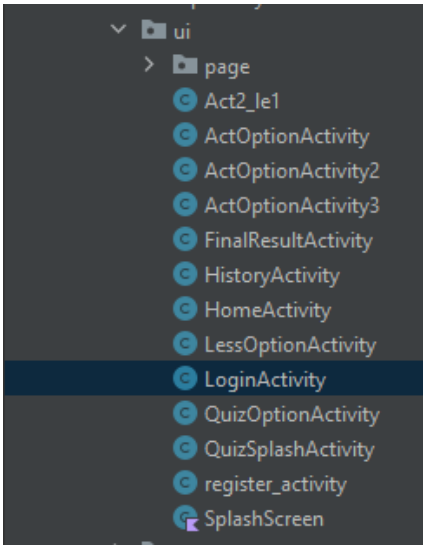
Se puede observar que debemos declarar la nueva actividad que creemos dentro de la "manifest" en orden que esta se compile con todos los archivos del programa, en orden que nosotros creemos otro archivo .java para definir el correcto funcionamiento de una interfaz creada, debemos declararla aquí.

Java

En la "carpeta" java se declara todos los controladores, adaptadores, entre otros ítems(scripts) que son muy necesarios para conectar el front-end con el back-end. Esto significa que, en esta carpeta debemos declarar las acciones de cada botón, o image view que nosotros tengamos en nuestra aplicación.



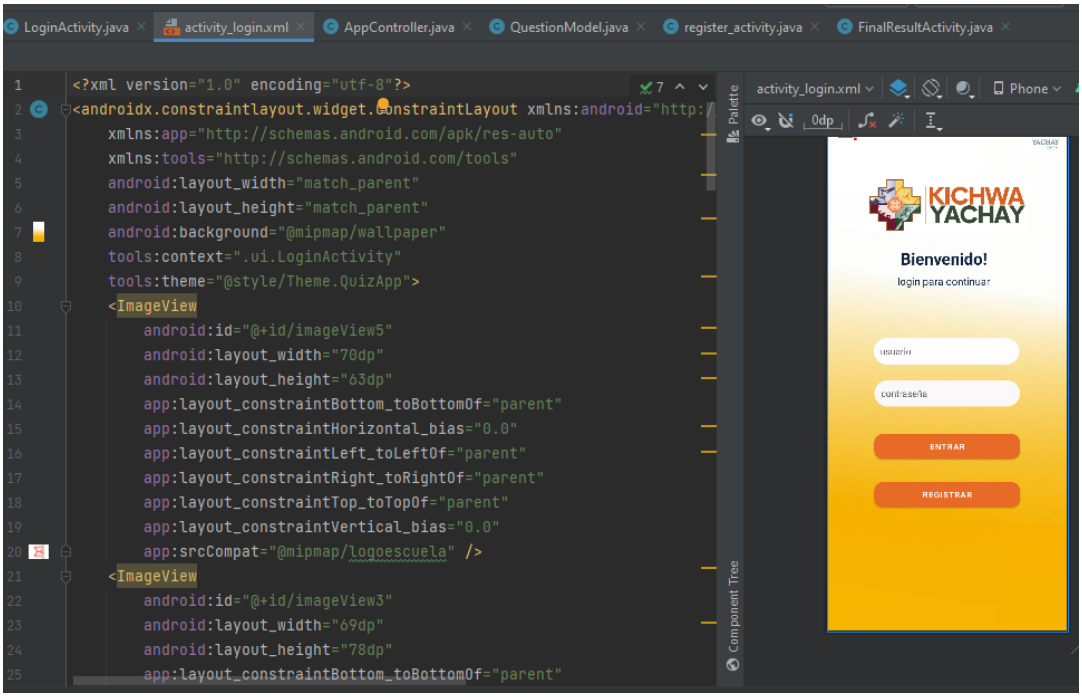
Donde, la carpeta "ui" almacena todos los controladores de interfaces de nuestra aplicación.



Donde, usando el archivo “Login Activity” como ejemplo, tenemos que, primeramente, declarar los botones del interfaz, para darles un evento a estos botones(acciones), luego es muy importante declarar qué layout estamos usando, en este caso es el de “activity_login”, esto es para darle acciones a esta interfaz anteriormente creada. Luego se declara la función “Validar Usuario” en orden de establecer conexión el servicio de la base de datos anteriormente creado. Finalmente, la función “validar usuario” tiene una sintaxis en orden de establecer conexión y recibir una respuesta del servidor, para saber si se estableció la conexión correctamente.

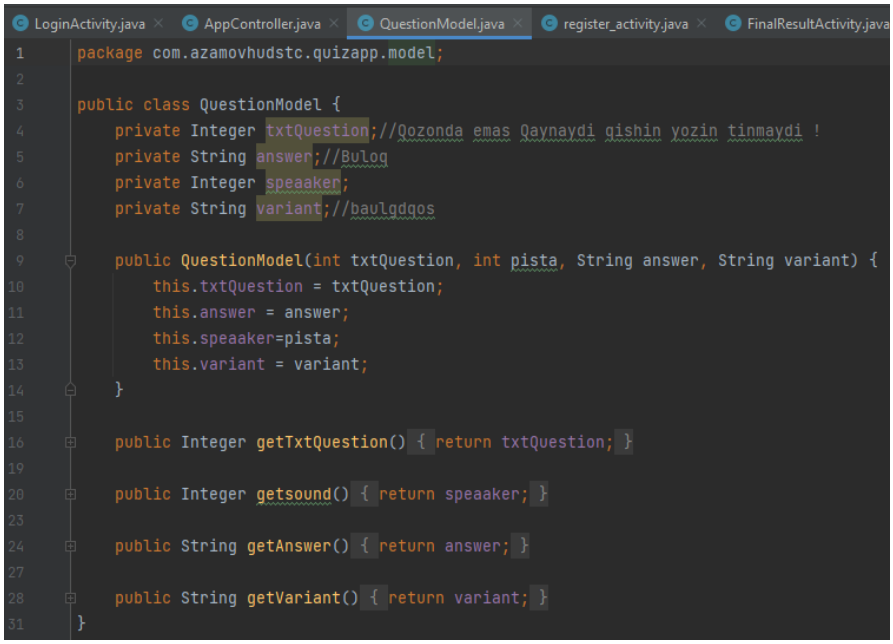
```java
package com.azamovhudstc.quizapp.ui;

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)

Luego, dentro de las carpetas “model” y “controller” encontramos archivos que tienen que ver con las preguntas y respuestas de las diferentes actividades, un breve ejemplo usaremos los archivos “app_controller” y “question model”.



Question model

```
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

Donde, en question model se define el diseño de la pregunta, donde esta tiene su respectiva pregunta, respuesta correcta, audio y la variante de la respuesta original. Además de tener las clases públicas en orden de retornar los valores finales hacia el script de manipulación de nuestra interfaz. En el archivo App Controller, básicamente usamos el diseño anteriormente creado en nuestro Question Model, y añadimos la correspondiente pregunta con su respuesta, audio y variante de la respuesta. Es importante mencionar que estos controladores corresponden para la actividad 2 de la lección 1.

RES

Finalmente tendremos la carpeta “res”, donde aquí tendremos todos los “layout” de nuestra aplicación, esto significa la interfaz de nuestra aplicación, más dedicada al front-end. Además de tener carpetas donde se almacenarán las imágenes usadas en nuestra aplicación, los archivos de audio, personalización de ciertos botones y flechas.

Gradle Scripts

Esta carpeta es muy importante, debido a que aquí importaremos todas las librerías necesarias para el correcto funcionamiento de nuestra aplicación(No eliminarla).