Experiment Report

Integrity Statement

Integrity Statement

I solemnly promise: This experimental report is the result of my independent work. To the best of my knowledge, the experimental report does not contain results that have been published or written by others, except where specifically noted. The contribution of any person who helped with the reported work is clearly stated and acknowledged in the lab report.

If there is plagiarism, I am willing to bear any consequences.

Hereby declare.

Sign: Jonas Gil Date: 2023.12.03

Topic: **ChatBot App**

Purpose: create a chatbot app whit the integration of OpenAI API

Experiment content and completion status:

1. Environment Configuration

Open Android Studio -> Create new project -> Select Kotlin

Writing dependencies:

Open build.gradle.kts (:app) go to -> dependencies -> write

```
implementation("com.squareup.okhttp3:okhttp:4.10.0")
```

all the dependencies:

```
dependencies {

    implementation("androidx.core:core-ktx:1.9.0")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.8.0")

implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")

androidTestImplementation("androidx.test.espresso:espresso-core:3.
```

```
5.1")
    implementation("com.squareup.okhttp3:okhttp:4.10.0")
}
```

2. Create the app

Open Activity_main.xml

- Create edit text to write question.

```
<EditText
    android:id="@+id/etQuestion"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Type your question here"
    android:textSize="20sp" />
```

- Create button to send question.

```
<Button
    android:id="@+id/btnSubmit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Submit"
    android:layout_marginTop="20dp" />
```

- Create TextView to show the answer

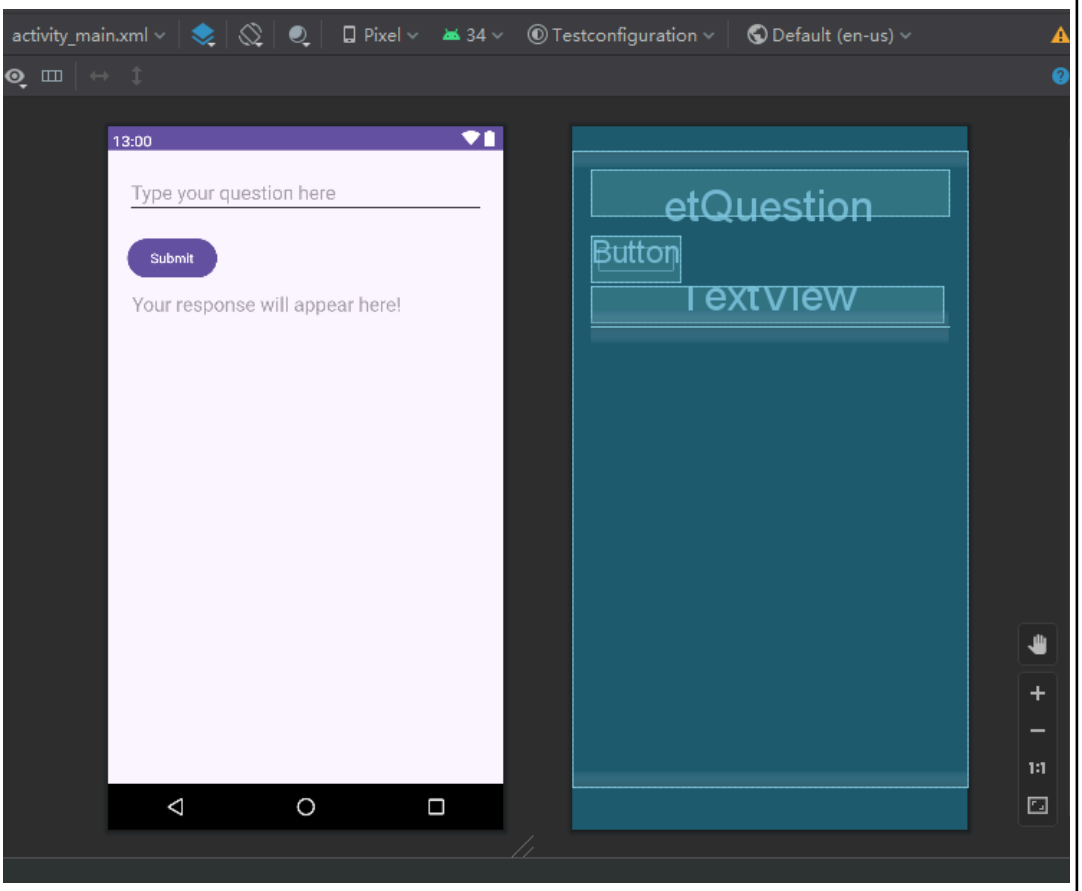
```
<TextView
    android:id="@+id/txtResponse"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Your response will appear here!"
    android:textSize="20sp"
    android:layout_marginTop="5dp"
    android:layout_marginEnd="5dp"
    android:layout_marginBottom="5dp"
    android:padding="5dp" />
```

All the code: Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
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:orientation="vertical"
    android:padding="20dp"
    tools:context=".MainActivity">

    <!-- EditText para escribir la pregunta -->
    <EditText
        android:id="@+id/etQuestion"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Type your question here"
        android:textSize="20sp" />

    <!-- Botón para enviar la pregunta -->
    <Button
        android:id="@+id/btnSubmit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:layout_marginTop="20dp" />

    <!-- TextView para mostrar la respuesta -->
    <TextView
        android:id="@+id/txtResponse"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Your response will appear here!"
        android:textSize="20sp"
        android:layout_marginTop="5dp"
        android:layout_marginEnd="5dp"
        android:layout_marginBottom="5dp"
        android:padding="5dp" />

    <!-- LinearLayout adicional -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"/>


</LinearLayout>
```

Interface design:



Coding the class Main_Activity.kt

- Getting view references

```
val etQuestion = findViewById<EditText>(R.id.etQuestion)
val btnSubmit = findViewById<Button>(R.id.btnSubmit)
val txtResponse = findViewById<TextView>(R.id.txtResponse)
```

- Defining the event whit a click

```
btnSubmit.setOnClickListener {
    // Getting the question from the EditText and displaying it in
a Toast
    val question = etQuestion.text.toString().trim()
    Toast.makeText(this, question, Toast.LENGTH_SHORT).show()
```

- Verifying if the question is not empty

```
if (question.isNotEmpty()) {
            // Calling the function to get a response and updating
the view with the response
```

```
            getResponse(question) { response ->
                runOnUiThread {
                    txtResponse.text = response
                }
            }
        }
    }
}
```

- Creating the function to get the question of the GPT-3 model

```
fun getResponse(question: String, callback: (String) -> Unit) {
    val apiKey = "YOUR_API_KEY" // Reemplaza con tu clave de API
de OpenAI
    val url =
"https://api.openai.com/v1/engines/text-davinci-003/completions"

    // Creating the request body in JSON format
    val requestBody = """
        {
            "prompt": "$question",
            "max_tokens": 500,
            "temperature": 0
        }
    """.trimIndent()
```

- Building th HTTP request using OkHttp

```
val request = Request.Builder()
    .url(url)
    .addHeader("Content-Type", "application/json")
    .addHeader("Authorization", "Bearer $apiKey")

.post(requestBody.toRequestBody("application/json".toMediaTypeOrNu
ll()))
    .build()
```

- Realizing the asynchrony call to the OpenAI API

```
client.newCall(request).enqueue(object : Callback {
    override fun onFailure(call: Call, e: IOException) {
        // Manejando fallos en la llamada a la API
        Log.e("error", "API failed", e)
```

```
    }

    override fun onResponse(call: Call, response: Response) {
        // Processing the API response
        val body = response.body?.string()

        if (body != null) {
            Log.v("data", body)
        } else {
            Log.v("data", "empty")
        }
```

Parsing the question to JSON

```
val jsonObject = JSONObject(body)
val jsonArray: JSONArray = jsonObject.getJSONArray("choices")
val textResult = jsonArray.getJSONObject(0).getString("text")
```

Call the callback whit the answer to update the user interface

```
callback(textResult)
```

All the code: Main_Activity.kt

```
package com.example.gpt_chat

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.inputmethod.EditorInfo
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import com.google.android.material.textfield.TextInputEditText
import okhttp3.*
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.RequestBody.Companion.toRequestBody
import org.json.JSONArray
import org.json.JSONObject
import java.io.IOException

class MainActivity : AppCompatActivity() {
    private val client = OkHttpClient()
```

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

        // Getting references to views
        val etQuestion = findViewById<EditText>(R.id.etQuestion)
        val btnSubmit = findViewById<Button>(R.id.btnSubmit)
        val txtResponse = findViewById<TextView>(R.id.txtResponse)

        // Defining the button click event
        btnSubmit.setOnClickListener {
            // Getting the question from the EditText and
displaying it in a Toast.
            val question = etQuestion.text.toString().trim()
            Toast.makeText(this, question,
Toast.LENGTH_SHORT).show()

            // Checking if the question is not empty
            if (question.isNotEmpty()) {


                // Calling the function to get a response and
updating the view with the response.

                getResponse(question) { response ->
                    runOnUiThread {
                        txtResponse.text = response
                    }
                }
            }
        }
    }

    // Function to obtain the response of the GPT-3 model
    fun getResponse(question: String, callback: (String) -> Unit)
{
        val apiKey = "YOUR_API_KEY" // Reemplaza con tu clave de
API de OpenAI
        val url =
"https://api.openai.com/v1/engines/text-davinci-003/completions"

        // Creando el cuerpo de la solicitud en formato JSON
        val requestBody = """
            {
```

```
                "prompt": "$question",
                "max_tokens": 500,
                "temperature": 0
            }
        """.trimIndent()

        // Building HTTP request using OkHttp
        val request = Request.Builder()
            .url(url)
            .addHeader("Content-Type", "application/json")
            .addHeader("Authorization", "Bearer $apiKey")

.post(requestBody.toRequestBody("application/json".toMediaTypeOrNu
ll()))
            .build()

        // Making the asynchronous call to the OpenAI API
        client.newCall(request).enqueue(object : Callback {
            override fun onFailure(call: Call, e: IOException) {
                // Handling API call failures
                Log.e("error", "API failed", e)
            }

            override fun onResponse(call: Call, response:
Response) {
                // Processing the API response
                val body = response.body?.string()

                if (body != null) {
                    Log.v("data", body)
                } else {
                    Log.v("data", "empty")
                }

                // Parseando la respuesta JSON
                val jsonObject = JSONObject(body)
                val jsonArray: JSONArray =
jsonObject.getJSONArray("choices")
                val textResult =
jsonArray.getJSONObject(0).getString("text")

                // Calling the callback with the response to
update the UI.
```

```
                callback(textResult)
            }
        })
    }
}
```

Problems: The program has some inconveniences with the Grade and the Android version.

Solutions (list problems encountered and solutions, list unsolved problems):

Experiment summary: In total, a project was created using Android Studio and the Kotlin programming language, an application was created, always with a button to send the text, a text viewer to show the answer and a text editor to write the question, the request was implemented HTTP via OkHttp, and the OpenAI API is called to get the response from the GPT-3 model.

References:

Acknowledgments: