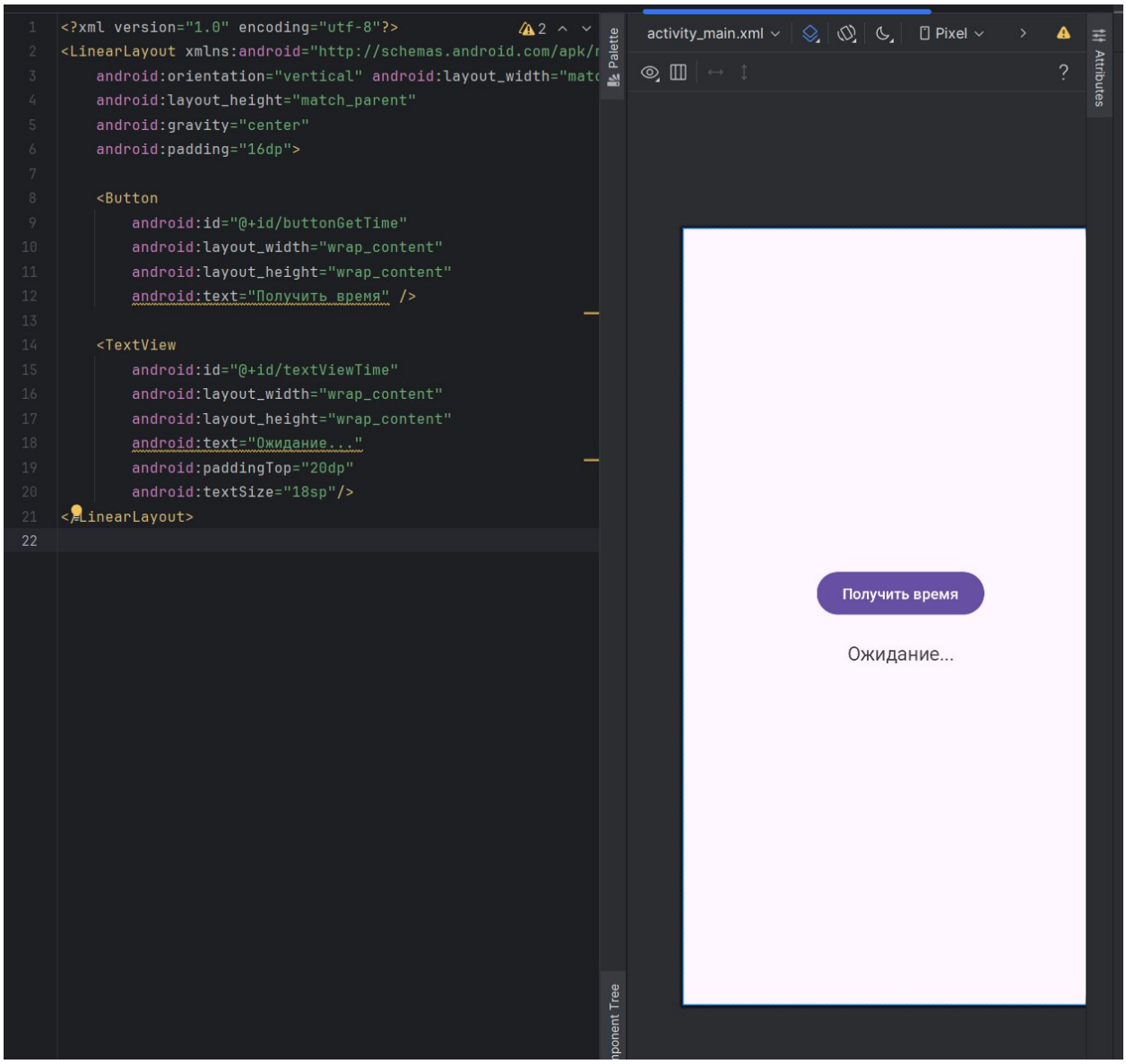
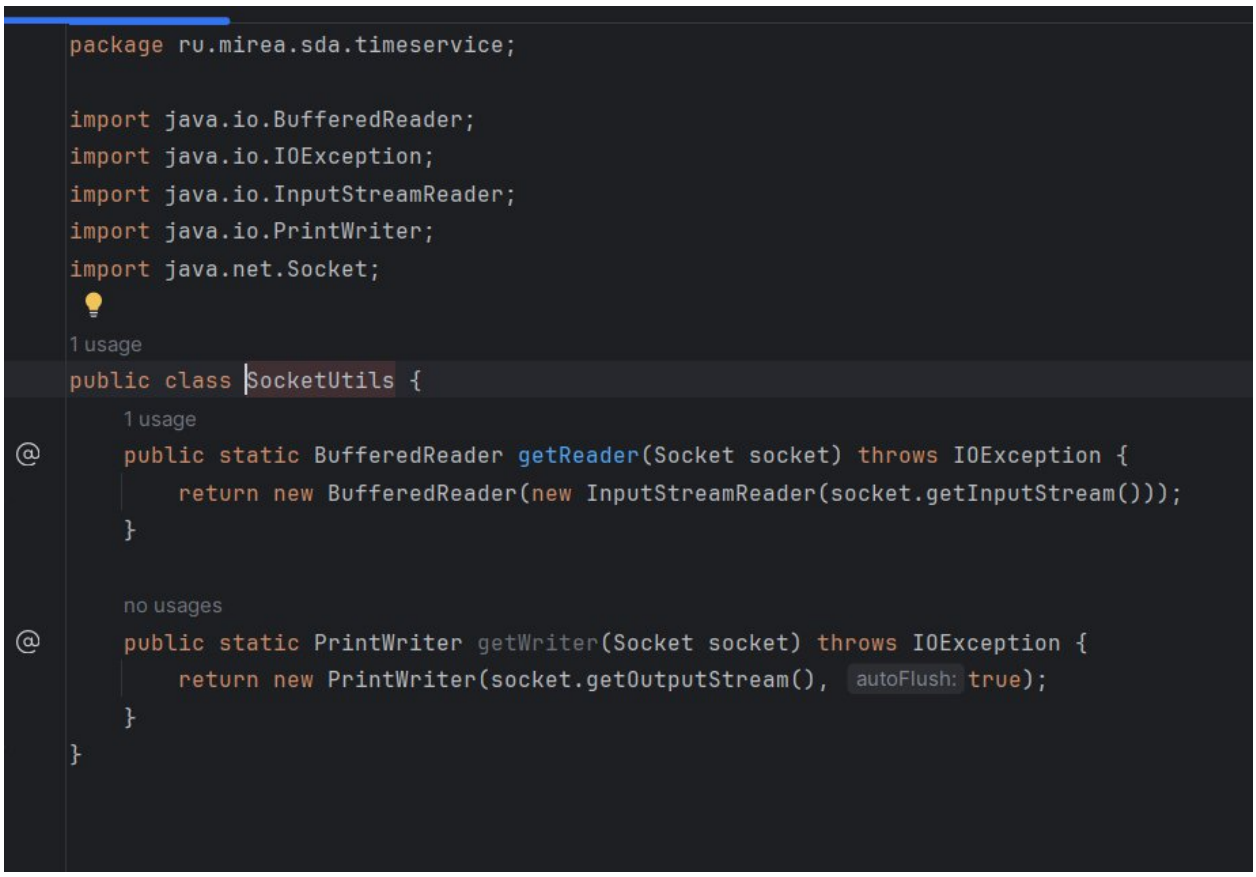
Практическая работа №7

Работа с сетевыми запросами, сокетами, библиотекой Retrofit и аутентификацией Firebase

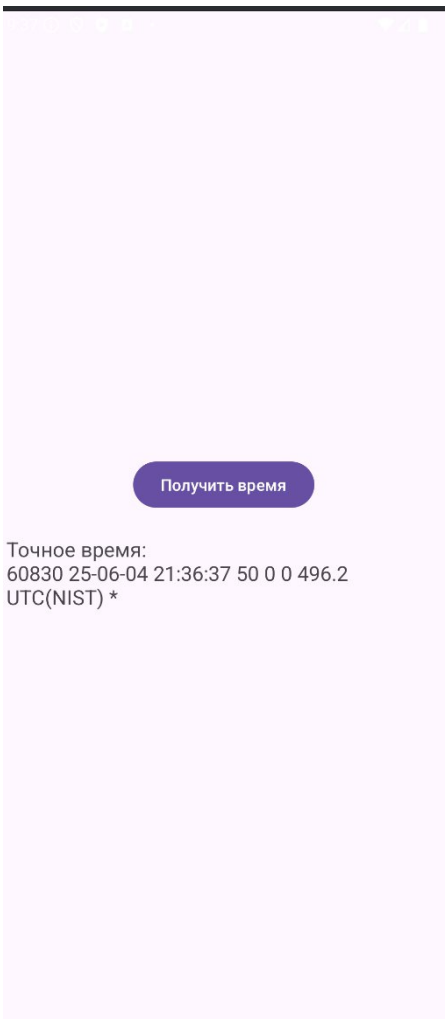
Задание 1. Сокет

Создан модуль timeservice. В интерфейсе размещены кнопка и TextView. Реализовано подключение к серверу time.nist.gov по протоколу TCP через сокет (порт 13). Использован AsyncTask для выполнения сетевого запроса в отдельном потоке. Полученное время отображается на экране.



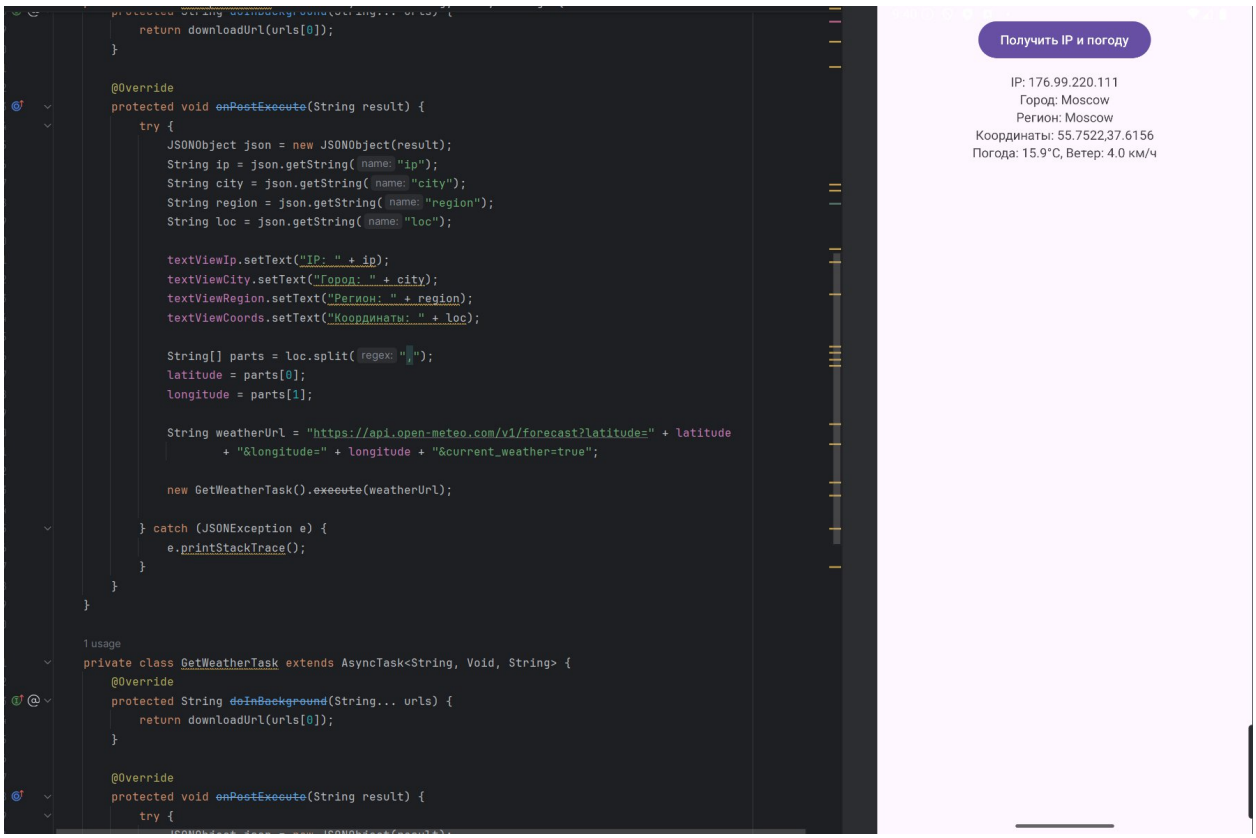

package ru.mirea.sda.timeservice;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
1 usage
public class SocketUtils {
1 usage
public static BufferedReader getReader(Socket socket) throws IOException {
return new BufferedReader(new InputStreamReader(socket.getInputStream()));
}
no usages
public static PrintWriter getWriter(Socket socket) throws IOException {
return new PrintWriter(socket.getOutputStream(), autoFlush: true);
}
}

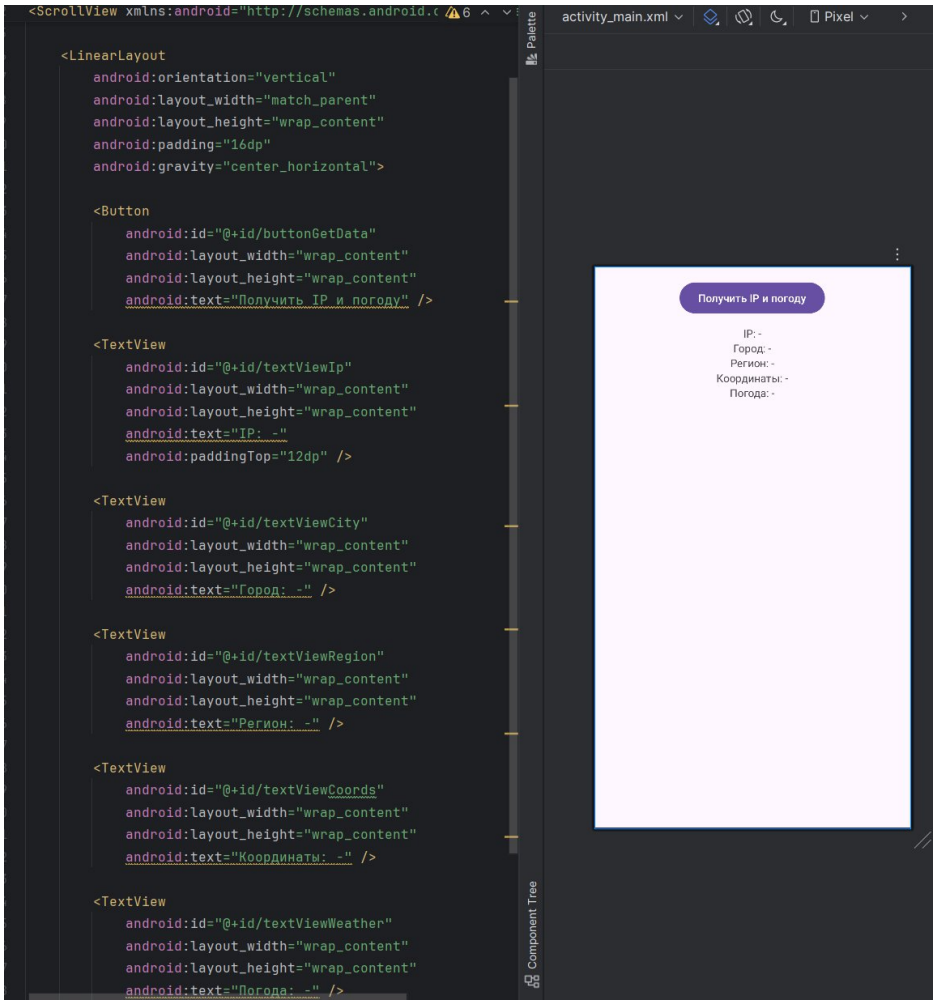
```
public class MainActivity extends AppCompatActivity {
    1 usage
    private static final String HOST = "time.nist.gov";
    1 usage
    private static final int PORT = 13;

    2 usages
    private TextView textViewTime;
    2 usages
    private Button buttonGetTime;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textViewTime = findViewById(R.id.textViewTime);
        buttonGetTime = findViewById(R.id.buttonGetTime);

        buttonGetTime.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { new GetTimeTask().execute(); }
        });
    }

    1 usage
    private class GetTimeTask extends AsyncTask<Void, Void, String> {
        @Override
        protected String doInBackground(Void... voids) {
            String result = "";
            try {
                Socket socket = new Socket(HOST, PORT);
                BufferedReader reader = SocketUtils.getReader(socket);
                reader.readLine();
                result = reader.readLine();
                socket.close();
            } catch (IOException e) {
                Log.e(TAG, msg: "Ошибка при получении времени", e);
                result = "Ошибка подключения";
            }
            return result;
        }

        @Override
        protected void onPostExecute(String time) {
            textViewTime.setText("Точное время:\n" + time);
```



Задание 2. HttpURLConnection

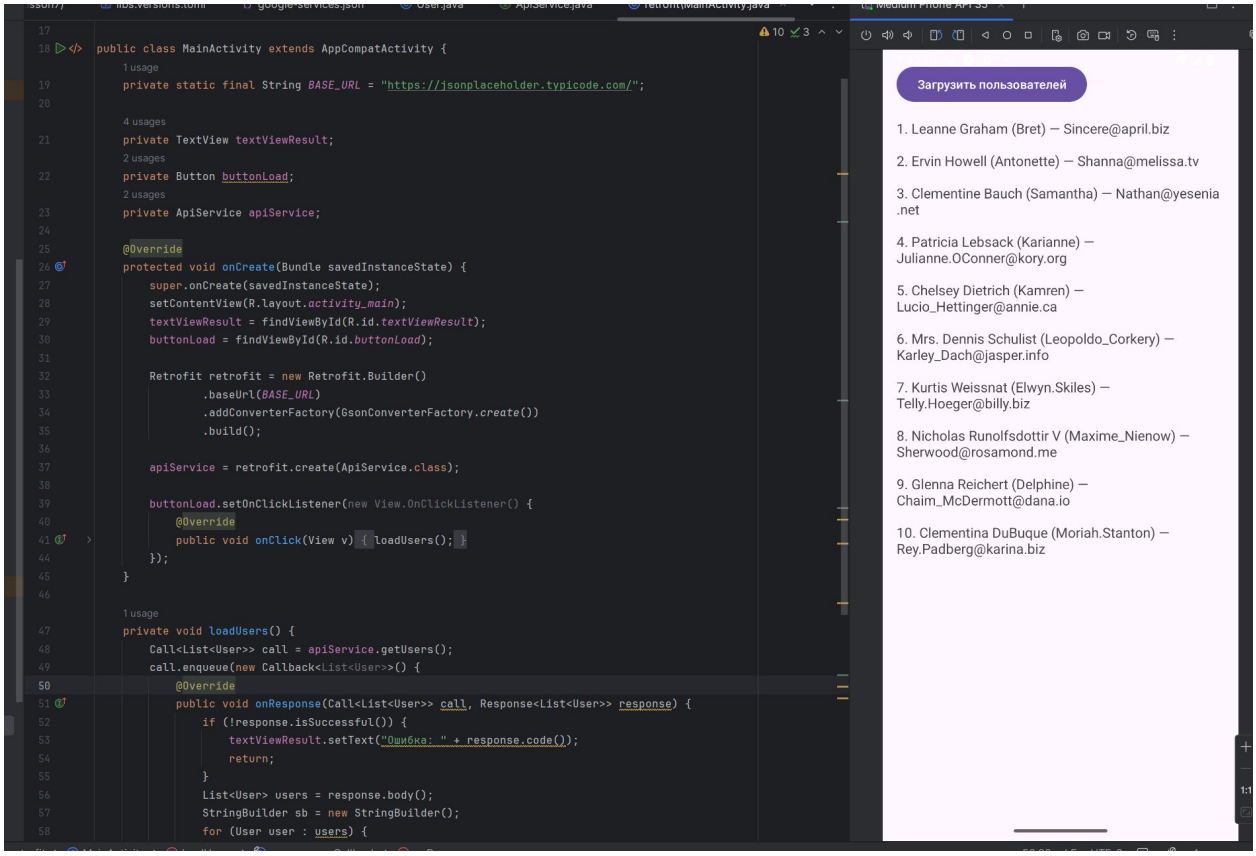
Создан модуль httpurlconnection. Приложение определяет внешний IP-адрес устройства и выводит город, регион и координаты, полученные из https://ipinfo.io/json. Далее координаты передаются в API open-meteo.com, и на экран выводится текущая погода. Работа реализована через AsyncTask с обработкой JSON-ответа.

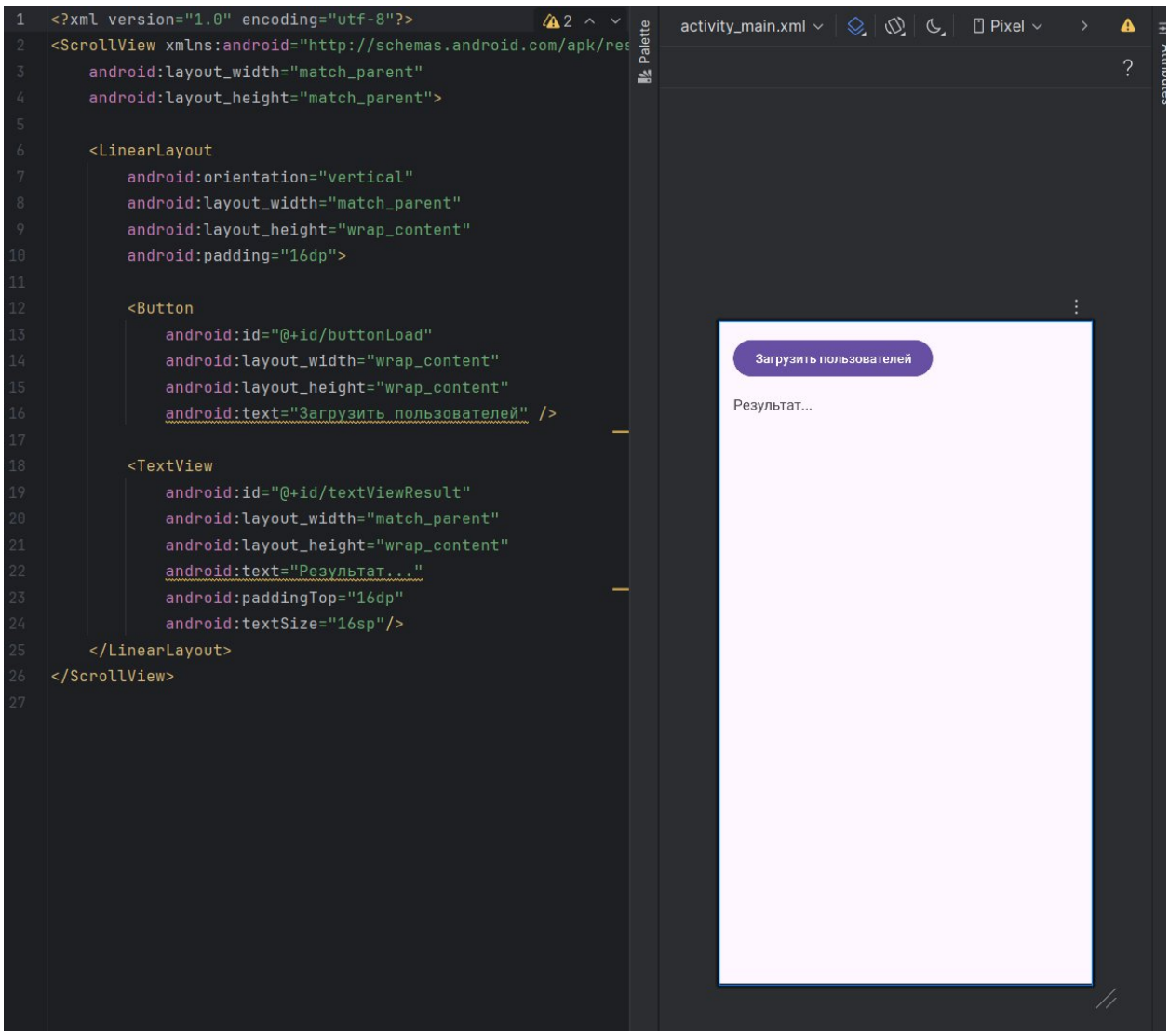

return downloadUrl(urls[0]);
}
@Override
protected void onPostExecute(String result) {
try {
JSONObject json = new JSONObject(result);
String ip = json.getString(name: "ip");
String city = json.getString(name: "city");
String region = json.getString(name: "region");
String loc = json.getString(name: "loc");
textViewIp.setText("IP: " + ip);
textViewCity.setText("Город: " + city);
textViewRegion.setText("Регион: " + region);
textViewCoords.setText("Координаты: " + loc);
String[] parts = loc.split(regex: ",");
latitude = parts[0];
longitude = parts[1];
String weatherUrl = "https://api.open-meteo.com/v1/forecast?latitude=" + latitude
+ "&longitude=" + longitude + "&current_weather=true";
new GetWeatherTask().execute(weatherUrl);
} catch (JSONException e) {
e.printStackTrace();
}
}
}
1 usage
private class GetWeatherTask extends AsyncTask<String, Void, String> {
@Override
protected String doInBackground(String... urls) {
return downloadUrl(urls[0]);
}
@Override
protected void onPostExecute(String result) {
try {
Получить IP и погоду
IP: 176.99.220.111
Город: Moscow
Регион: Moscow
Координаты: 55.7522,37.6156
Погода: 15.9°C, Ветер: 4.0 км/ч


<ScrollView xmlns:android="http://schemas.android.
<LinearLayout
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:padding="16dp"
android:gravity="center_horizontal">
<Button
android:id="@+id/buttonGetData"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Получить IP и погоду" />
<TextView
android:id="@+id/textViewIp"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="IP: -"
android:paddingTop="12dp" />
<TextView
android:id="@+id/textViewCity"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Город: -" />
<TextView
android:id="@+id/textViewRegion"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Регион: -" />
<TextView
android:id="@+id/textViewCoords"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Координаты: -" />
<TextView
android:id="@+id/textViewWeather"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Погода: -" />
Palette
activity_main.xml
Pixel
Получить IP и погоду
IP: -
Город: -
Регион: -
Координаты: -
Погода: -
Component Tree

Задание 3. Retrofit

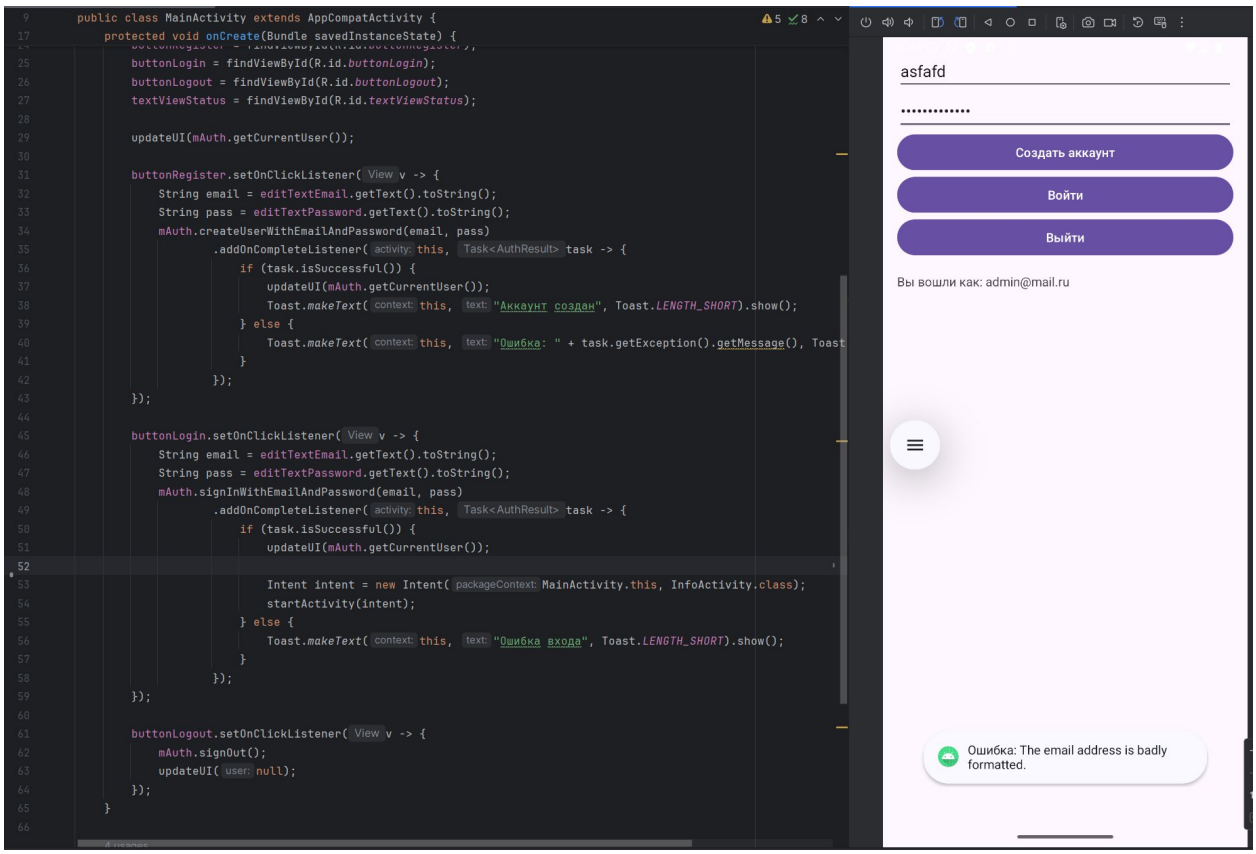
Создан модуль retrofit. С помощью библиотеки Retrofit выполнен запрос к API времени (https://timeapi.io). Данные выводятся в TextView. Использован GsonConverterFactory для парсинга ответа. Инициализация Retrofit и вызов API выполнены в onCreate.

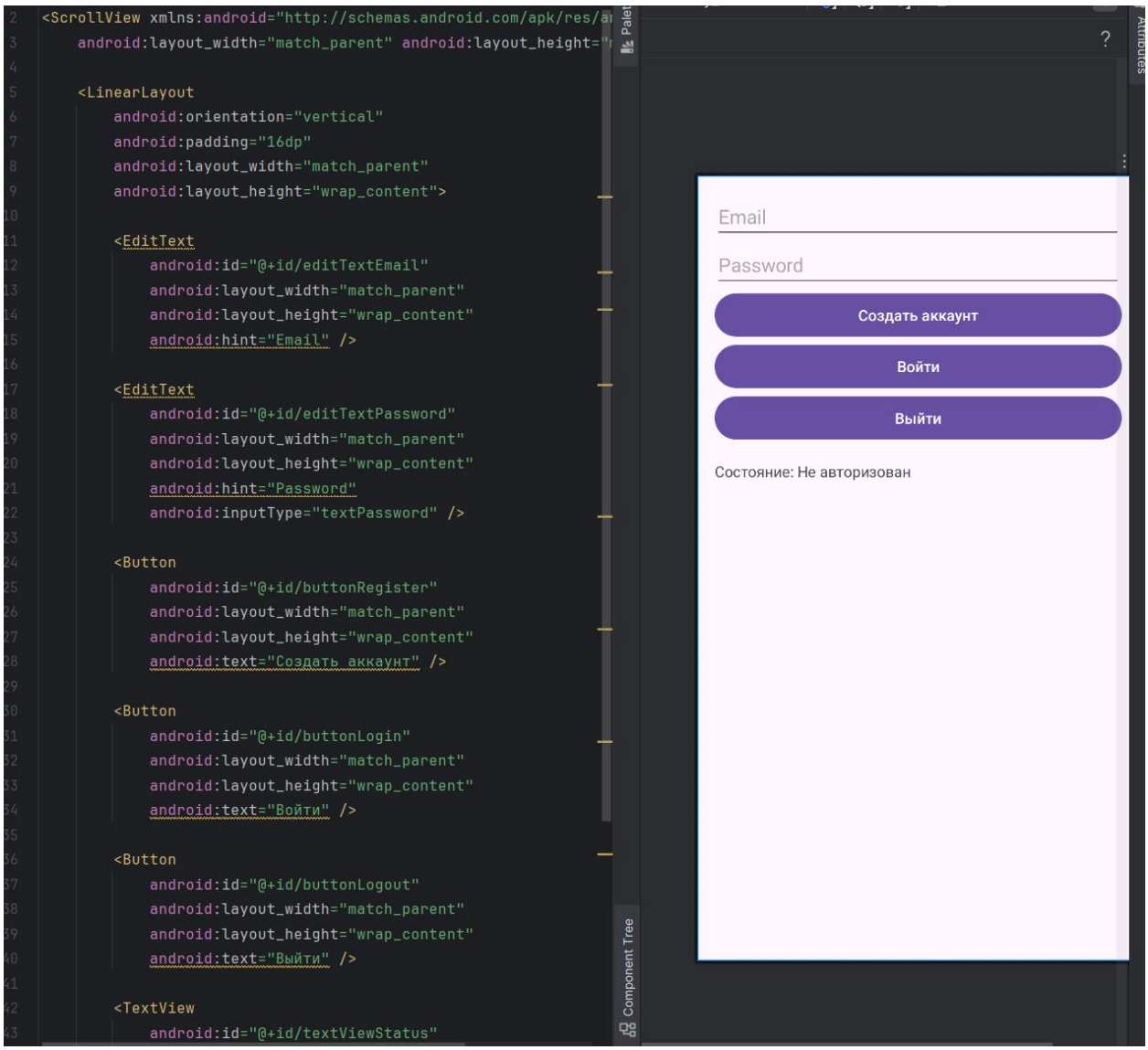




Задание 4. Firebase Authentication

Создан модуль firebaseauth. Приложение позволяет зарегистрироваться и войти в систему с помощью электронной почты и пароля. После авторизации отображаются UID и email пользователя. Добавлена верификация почты и возможность выхода из учётной записи. Используется FirebaseAuth и FirebaseUser.


public class MainActivity extends AppCompatActivity {
protected void onCreate(Bundle savedInstanceState) {
buttonLogin = findViewById(R.id.buttonLogin);
buttonLogout = findViewById(R.id.buttonLogout);
textViewStatus = findViewById(R.id.textViewStatus);
updateUI(mAuth.getCurrentUser());
buttonRegister.setOnClickListener(View v -> {
String email = editTextEmail.getText().toString();
String pass = editTextPassword.getText().toString();
mAuth.createUserWithEmailAndPassword(email, pass)
.addOnCompleteListener(activity: this, Task<AuthResult> task -> {
if (task.isSuccessful()) {
updateUI(mAuth.getCurrentUser());
Toast.makeText(context: this, text: "Аккаунт создан", Toast.LENGTH_SHORT).show();
} else {
Toast.makeText(context: this, text: "Ошибка: " + task.getException().getMessage(), Toast
}
});
});
buttonLogin.setOnClickListener(View v -> {
String email = editTextEmail.getText().toString();
String pass = editTextPassword.getText().toString();
mAuth.signInWithEmailAndPassword(email, pass)
.addOnCompleteListener(activity: this, Task<AuthResult> task -> {
if (task.isSuccessful()) {
updateUI(mAuth.getCurrentUser());
Intent intent = new Intent(packageContext: MainActivity.this, InfoActivity.class);
startActivity(intent);
} else {
Toast.makeText(context: this, text: "Ошибка входа", Toast.LENGTH_SHORT).show();
}
});
});
buttonLogout.setOnClickListener(View v -> {
mAuth.signOut();
updateUI(user: null);
});
}
asfafd
Создать аккаунт
Войти
Выйти
Вы вошли как: admin@mail.ru
Ошибка: The email address is badly formatted.

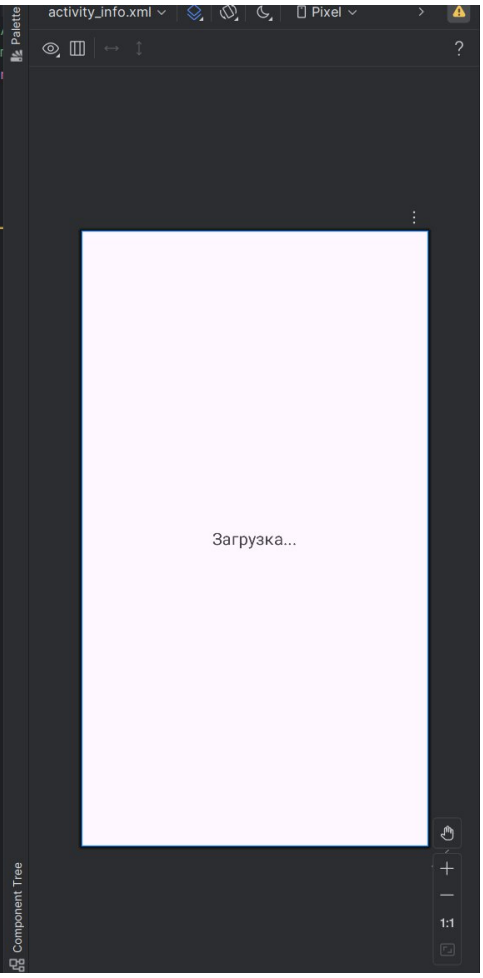


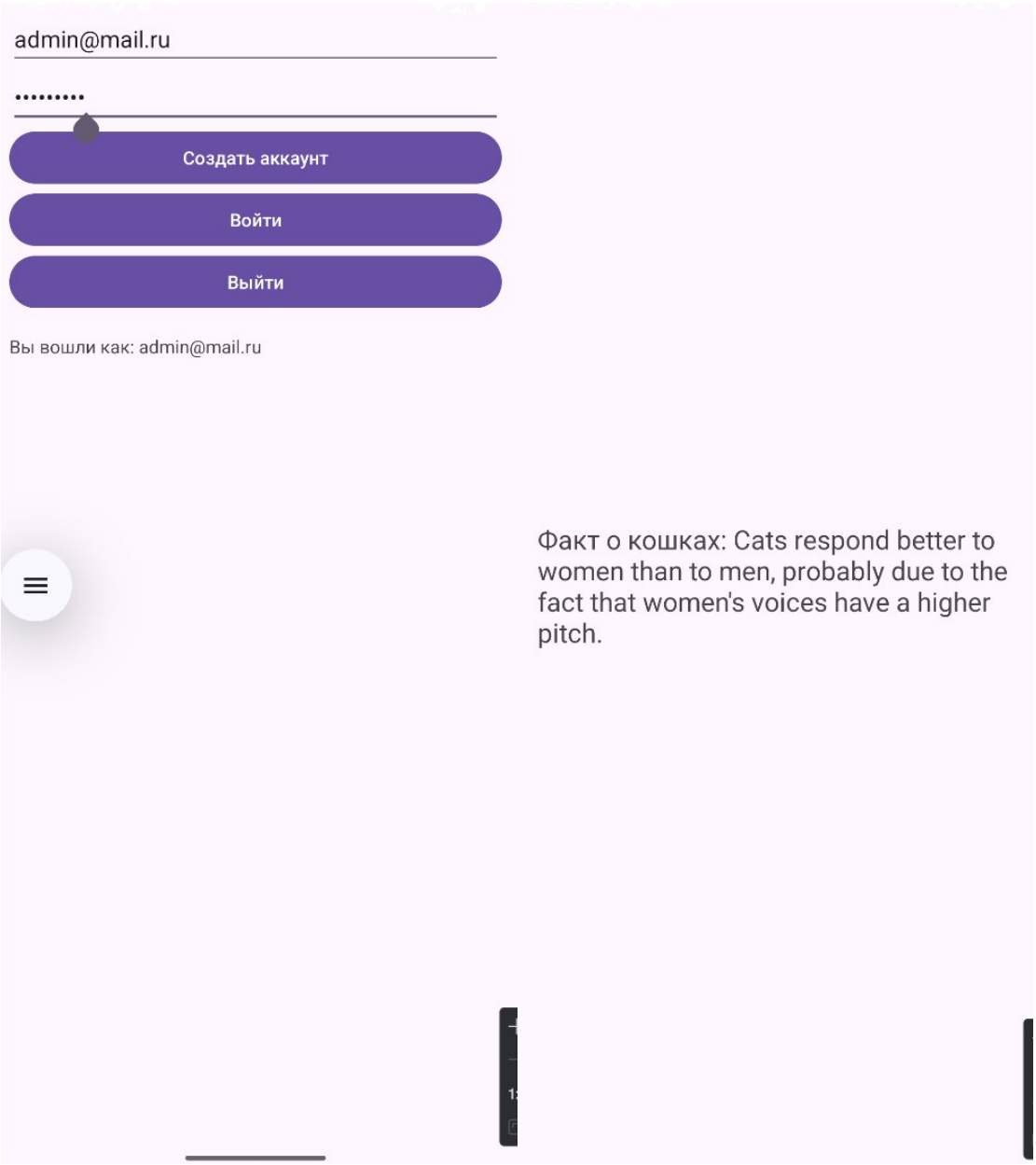
Контрольное задание

В рамках контрольного задания в модуле firebaseauth реализован экран входа в приложение с использованием Firebase Authentication. После успешной авторизации отображается дополнительная информация, полученная из внешнего API — факт о кошках. Данные загружаются по сети и выводятся на экран в TextView. Таким образом, выполнены оба условия задания: авторизация и отображение данных из сетевого источника.

```java
import androidx.appcompat.app.AppCompatActivity;
import retrofit2.*;
import retrofit2.converter.gson.GsonConverterFactory;

public class InfoActivity extends AppCompatActivity {

    4 usages
    private TextView textViewFact;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_info);

        textViewFact = findViewById(R.id.textViewTime);

        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl("https://catfact.ninja/")
                .addConverterFactory(GsonConverterFactory.create())
                .build();

        CatApi catApi = retrofit.create(CatApi.class);

        catApi.getCatFact().enqueue(new Callback<CatFactResponse>() {
            @Override
            public void onResponse(Call<CatFactResponse> call, Response<CatFactResponse> response) {
                if (response.isSuccessful() && response.body() != null) {
                    textViewFact.setText("Факт о кошках: " + response.body().fact);
                } else {
                    textViewFact.setText("Ошибка: " + response.code());
                }
            }

            @Override
            public void onFailure(Call<CatFactResponse> call, Throwable t) {
                textViewFact.setText("Ошибка запроса: " + t.getMessage());
            }
        });
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
    android:layout_width="match_parent" android:layout_height="
    android:orientation="vertical" android:gravity="center" andr

    <TextView
        android:id="@+id/textViewTime"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Загрузка..."
        android:textSize="20sp" />
</LinearLayout>
```





При нажатии на кнопку “Войти” подгружается какой-то факт о кошках на английском языке