**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего образования «Уральский федеральный университет

имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РТФ

Центр ускоренного обучения

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 20**

по дисциплине «Разработка мобильных приложений»

**Тема: Unit 3 Работа в фоновом режиме “AsyncTask и AsyncTaskLoader”**

Студент группы РИЗ-300028у: А.С. Севостьянов

Преподаватель: С.И.Тимошенко,

доц., к.т.н.

**Екатеринбург 2022**

содержание

[1 Постановка задачи 3](#_Toc118041527)

[2 Результаты проделанной работы 4](#_Toc118041528)

[3 Анализ полученных результатов 5](#_Toc118041529)

## 1 Постановка задачи

Использовать Google API Explorer, чтобы узнать об API книг. Создать приложение "Кто это написал?", Которое запрашивает API Books с помощью рабочего потока и отображает результат в пользовательском интерфейсе. Изменить приложение "Кто это написал?", чтобы использовать AsyncTaskLouder вместо AsyncTask.

.

## Результаты проделанной работы

Программный код проекта, файл MainActivity.

package com.example.task71;  
  
import android.content.Context;  
import android.net.ConnectivityManager;  
import android.net.NetworkInfo;  
import android.os.Bundle;  
import android.view.View;  
import android.view.inputmethod.InputMethodManager;  
import android.widget.EditText;  
import android.widget.TextView;  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.loader.app.LoaderManager;  
import androidx.loader.content.Loader;  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
*/\*\*  
 Приложение WhoWroteIt запрашивает API поиска книг на основе  
 при поиске пользователя. Он использует AsyncTask для выполнения задачи поиска в  
 фоновом режиме.  
 \*/*public class MainActivity extends AppCompatActivity  
 implements LoaderManager.LoaderCallbacks<String> {  
  
 private EditText mBookInput;  
 private TextView mTitleText;  
 private TextView mAuthorText;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 mBookInput = findViewById(R.id.*bookInput*);  
 mTitleText = findViewById(R.id.*titleText*);  
 mAuthorText = findViewById(R.id.*authorText*);  
  
 if (getSupportLoaderManager().getLoader(0) != null) {  
 getSupportLoaderManager().initLoader(0, null, this);  
 }  
 }  
  
 */\*\*  
 \* Обработчик onClick для кнопки "Поиск книг".  
 \*/* public void searchBooks(View view) {  
 // Получите строку поиска из поля ввода.  
 String queryString = mBookInput.getText().toString();  
 // Скрывайте клавиатуру при нажатии кнопки.  
 InputMethodManager inputManager = (InputMethodManager)  
 getSystemService(Context.*INPUT\_METHOD\_SERVICE*);  
 if (inputManager != null) {  
 inputManager.hideSoftInputFromWindow(view.getWindowToken(),  
 InputMethodManager.*HIDE\_NOT\_ALWAYS*);  
 }  
  
// Проверьте состояние сетевого подключения.  
 ConnectivityManager connMgr = (ConnectivityManager)  
 getSystemService(Context.*CONNECTIVITY\_SERVICE*);  
 NetworkInfo networkInfo = null;  
 if (connMgr != null) {  
 networkInfo = connMgr.getActiveNetworkInfo();  
 }  
  
// Если сеть доступна, подключена и поле поиска  
// не пусто, запустите загрузчик книг AsyncTask.  
 if (networkInfo != null && networkInfo.isConnected()  
 && queryString.length() != 0) {  
  
 Bundle queryBundle = new Bundle();  
 queryBundle.putString("queryString", queryString);  
 getSupportLoaderManager().restartLoader(0, queryBundle, this);  
  
 mAuthorText.setText("");  
 mTitleText.setText(R.string.*loading*);  
 }  
// В противном случае обновите TextView, чтобы сообщить пользователю, что нет соединения  
 // или нет поискового запроса.  
 else {  
 if (queryString.length() == 0) {  
 mAuthorText.setText("");  
 mTitleText.setText(R.string.*no\_search\_term*);  
 } else {  
 mAuthorText.setText("");  
 mTitleText.setText(R.string.*no\_network*);  
 }  
 }  
 }  
  
  
 @NonNull  
 @Override  
 public Loader<String> onCreateLoader(int id, @Nullable Bundle args) {  
 String queryString = "";  
  
 if (args != null) {  
 queryString = args.getString("queryString");  
 }  
  
 return new BookLoader(this, queryString);  
 }  
  
 @Override  
 public void onLoadFinished(@NonNull Loader<String> loader, String data) {  
 try {  
 // Преобразуйте ответ в объект JSON.  
 JSONObject jsonObject = new JSONObject(data);  
// Получить JSONArray элементов книги.  
 JSONArray itemsArray = jsonObject.getJSONArray("items");  
  
 // Инициализируйте поля итератора и результа// Преобразуйте ответ в объект JSON.тов.  
 int i = 0;  
 String title = null;  
 String authors = null;  
  
 // Ищите результаты в массиве элементов, завершая работу, когда найдены как заголовок, так и автор  
 // или когда все элементы были проверены.  
 while (i < itemsArray.length() &&  
 (authors == null && title == null)) {  
 // Получить текущую информацию о товаре.  
 JSONObject book = itemsArray.getJSONObject(i);  
 JSONObject volumeInfo = book.getJSONObject("volumeInfo");  
  
// Попробуйте получить автора и название из текущего элемента,  
// поймайте, если какое-либо поле пусто, и двигайтесь дальше.  
 try {  
 title = volumeInfo.getString("title");  
 authors = volumeInfo.getString("authors");  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
 // Перейдите к следующему пункту.  
 i++;  
 }  
  
 // Если найдено и то, и другое, отобразите результат.  
 if (title != null && authors != null) {  
 mTitleText.setText(title);  
 mAuthorText.setText(authors);  
 //mBookInput.setText("");  
 } else {  
 // Если ничего не найдено, обновите пользовательский интерфейс, чтобы отобразить неудачные результаты.  
 mTitleText.setText(R.string.*no\_results*);  
 mAuthorText.setText("");  
 }  
  
 } catch (Exception e) {  
// Если onPostExecute не получает правильную строку JSON,  
// обновите пользовательский интерфейс, чтобы показать неудачные результаты.  
 mTitleText.setText(R.string.*no\_results*);  
 mAuthorText.setText("");  
 e.printStackTrace();  
 }  
  
 }  
  
 @Override  
 public void onLoaderReset(@NonNull Loader<String> loader) {  
 }  
}

Программный код проекта, файл FetchBook.

package com.example.task71;

import android.os.AsyncTask;  
import android.widget.TextView;  
import org.json.JSONArray;  
import org.json.JSONObject;  
import java.lang.ref.WeakReference;  
  
public class FetchBook extends AsyncTask<String, Void, String> {  
 private WeakReference<TextView> mTitleText;  
 private WeakReference<TextView> mAuthorText;  
  
 public FetchBook(TextView mTitleText, TextView mAuthorText) {  
 this.mTitleText = mTitleText;  
 this.mAuthorText = mAuthorText;  
 }  
  
  
 @Override  
 protected String doInBackground(String... strings) {  
 return null;  
 }  
 @Override  
 protected void onPostExecute(String s) {  
 super.onPostExecute(s);  
 try {  
 // Преобразуйте ответ в объект JSON.  
 JSONObject jsonObject = new JSONObject(s);  
 // Получите JSONArray элементов книги.  
 JSONArray itemsArray = jsonObject.getJSONArray("items");  
  
 // Получить JSONArray элементов книги.  
 int i = 0;  
 String title = null;  
 String authors = null;  
  
 // Ищите результаты в массиве элементов, выходя  
 // когда и название, и автор  
 // найдены или когда все элементы были проверены.  
 while (i < itemsArray.length() &&  
 (authors == null && title == null)) {  
 // Получите текущую информацию о товаре.  
 JSONObject book = itemsArray.getJSONObject(i);  
 JSONObject volumeInfo = book.getJSONObject("volumeInfo");  
  
 // Попробуйте получить автора и название из текущего элемента,  
 // поймайте, если какое-либо поле пусто, и двигайтесь дальше.  
 try {  
 title = volumeInfo.getString("title");  
 authors = volumeInfo.getString("authors");  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
 // Перейти к следующему пункту.  
 i++;  
 }  
  
 // Если найдено и то, и другое, отобразите результат.  
 if (title != null && authors != null) {  
 mTitleText.get().setText(title);  
 mAuthorText.get().setText(authors);  
 } else {  
 // Если ничего не найдено, обновите пользовательский интерфейс до  
 // показать неудачные результаты.  
 mTitleText.get().setText(R.string.*no\_results*);  
 mAuthorText.get().setText("");  
 }  
  
 } catch (Exception e) {  
 // Если onPostExecute не получает правильную строку JSON,  
 // обновите пользовательский интерфейс, чтобы отобразить неудачные результаты.  
 mTitleText.get().setText(R.string.*no\_results*);  
 mAuthorText.get().setText("");  
 }  
 }  
}

Программный код проекта, файл NetworkUtils.

package com.example.task71;  
  
  
import android.net.Uri;  
import android.util.Log;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.net.HttpURLConnection;  
import java.net.URL;  
  
*/\*\*  
 Класс утилиты для использования Google Book Search API для загрузки книги Sинформация  
 \*/*public class NetworkUtils {  
  
 private static final String *LOG\_TAG* = NetworkUtils.class.getSimpleName();  
  
 //Константы для различных компонентов запроса Books API.  
 // Базовый URL конечной точки для Books API.  
  
 private static final String *BOOK\_BASE\_URL* ="https://www.googleapis.com/books/v1/volumes?";  
 // Параметр для строки поиска.  
 private static final String *QUERY\_PARAM* = "q";  
 // Параметр, ограничивающий результаты поиска.  
 private static final String *MAX\_RESULTS* = "maxResults";  
 // Параметр для фильтрации по типу печати.  
 private static final String *PRINT\_TYPE* = "printType";  
  
 */\*\*  
 Статический метод для выполнения фактического запроса к Books API.  
 \*\*/* static String getBookInfo(String queryString) {  
  
// Настройте переменные для блока try, которые необходимо закрыть в блоке  
// finally.  
 HttpURLConnection urlConnection = null;  
 BufferedReader reader = null;  
 String bookJSONString = null;  
  
 try {  
 // Создайте полный URI запроса, ограничив результаты 10 элементами и  
 // печатными книгами.  
 Uri builtURI = Uri.*parse*(*BOOK\_BASE\_URL*).buildUpon()  
 .appendQueryParameter(*QUERY\_PARAM*, queryString)  
 .appendQueryParameter(*MAX\_RESULTS*, "10")  
 .appendQueryParameter(*PRINT\_TYPE*, "books")  
 .build();  
  
 // Преобразуйте URI в URL.  
 URL requestURL = new URL(builtURI.toString());  
  
 // Откройте сетевое подключение.  
 urlConnection = (HttpURLConnection) requestURL.openConnection();  
 urlConnection.setRequestMethod("GET");  
 urlConnection.connect();  
  
  
  
  
 // Получить входной поток.  
 InputStream inputStream = urlConnection.getInputStream();  
 // Создайте буферизованный считыватель из этого входного потока.  
 reader = new BufferedReader(new InputStreamReader(inputStream));  
  
 // Используйте StringBuilder для хранения входящего ответа.  
 StringBuilder builder = new StringBuilder();  
 String line;  
 while ((line = reader.readLine()) != null) {  
 // Добавьте текущую строку в строку.  
 builder.append(line);  
  
 // Поскольку это JSON, добавление новой строки необязательно (это не  
 // повлияет на синтаксический анализ), но это делает отладку \* намного \* проще  
 // если вы распечатаете заполненный буфер для отладки.  
 builder.append("\n");  
 }  
  
 if (builder.length() == 0) {  
 //Выйдитбез разбора.  
 return null;  
 }  
  
 bookJSONString = builder.toString();  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 } finally {  
 // Закройте соединение и буферизованный считыватель.  
 if (urlConnection != null) {  
 urlConnection.disconnect();  
 }  
 if (reader != null) {  
 try {  
 reader.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 // Запишите окончательный ответ JSON в журнал  
 Log.*d*(*LOG\_TAG*, bookJSONString);  
  
 return bookJSONString;  
 }  
}

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout 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"  
 tools:context=".MainActivity">  
  
 <TextView  
 android:id="@+id/instructions"  
 android:layout\_width="386dp"  
 android:layout\_height="57dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="64dp"  
 android:text="@string/instructions"  
 android:textAppearance="@style/TextAppearance.AppCompat.Title"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <EditText  
 android:id="@+id/bookInput"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:hint="@string/input\_hint"  
 android:inputType="text"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/instructions" />  
  
 <Button  
 android:id="@+id/searchButton"  
 android:layout\_width="402dp"  
 android:layout\_height="60dp"  
 android:layout\_marginTop="16dp"  
 android:onClick="searchBooks"  
 android:text="@string/button\_text"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/bookInput" />  
  
 <TextView  
 android:id="@+id/titleText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:textAppearance="@style/TextAppearance.AppCompat.Headline"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/searchButton" />  
  
 <TextView  
 android:id="@+id/authorText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:textAppearance="@style/TextAppearance.AppCompat.Headline"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/titleText"/>  
  
</androidx.constraintlayout.widget.ConstraintLayout>

На рисунке 1 показано начальное состояния приложения.

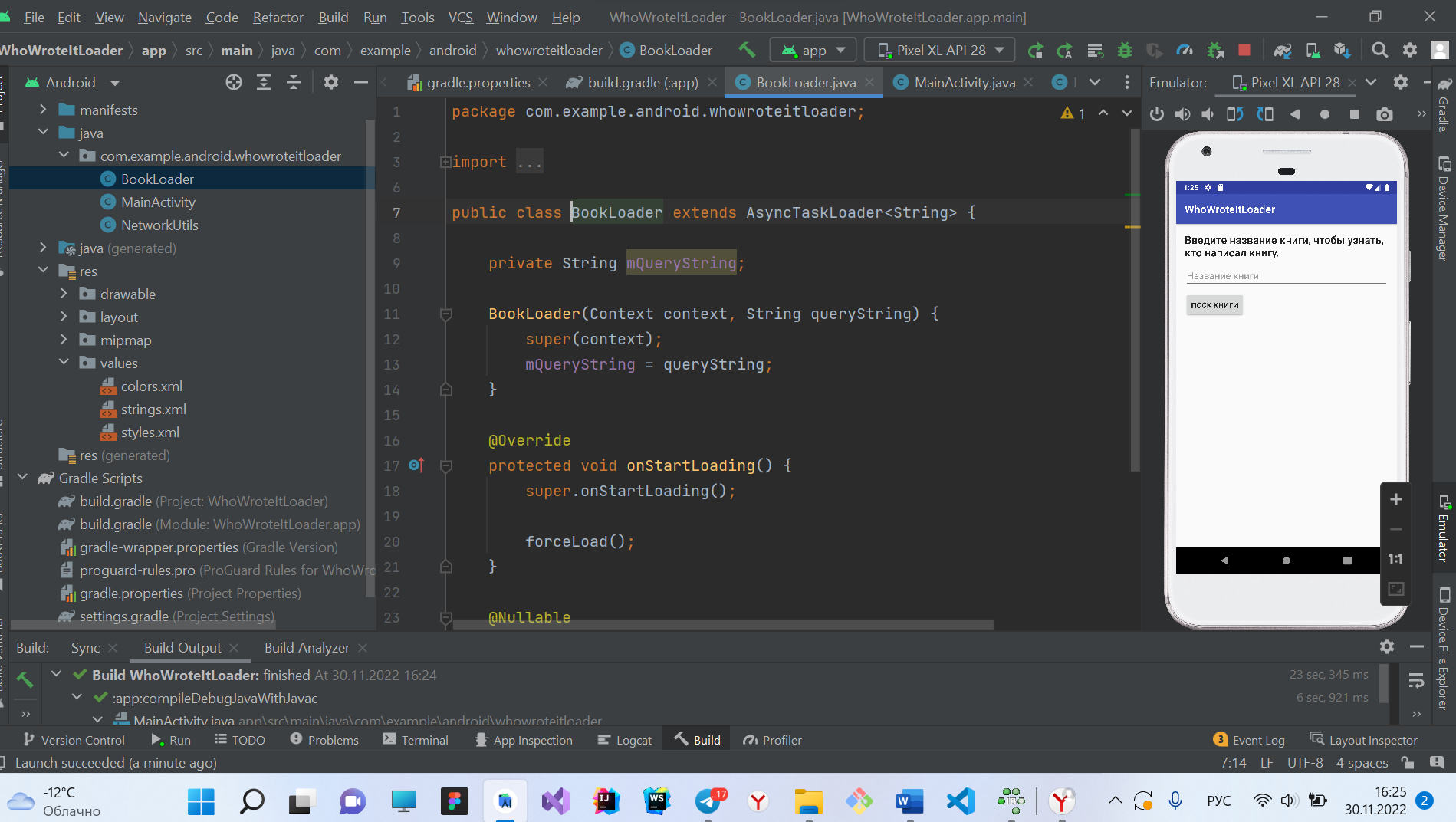


Рисунок 1 – Поиск книги

На рисунке 2 показано результат работы программы после успешного запроса.

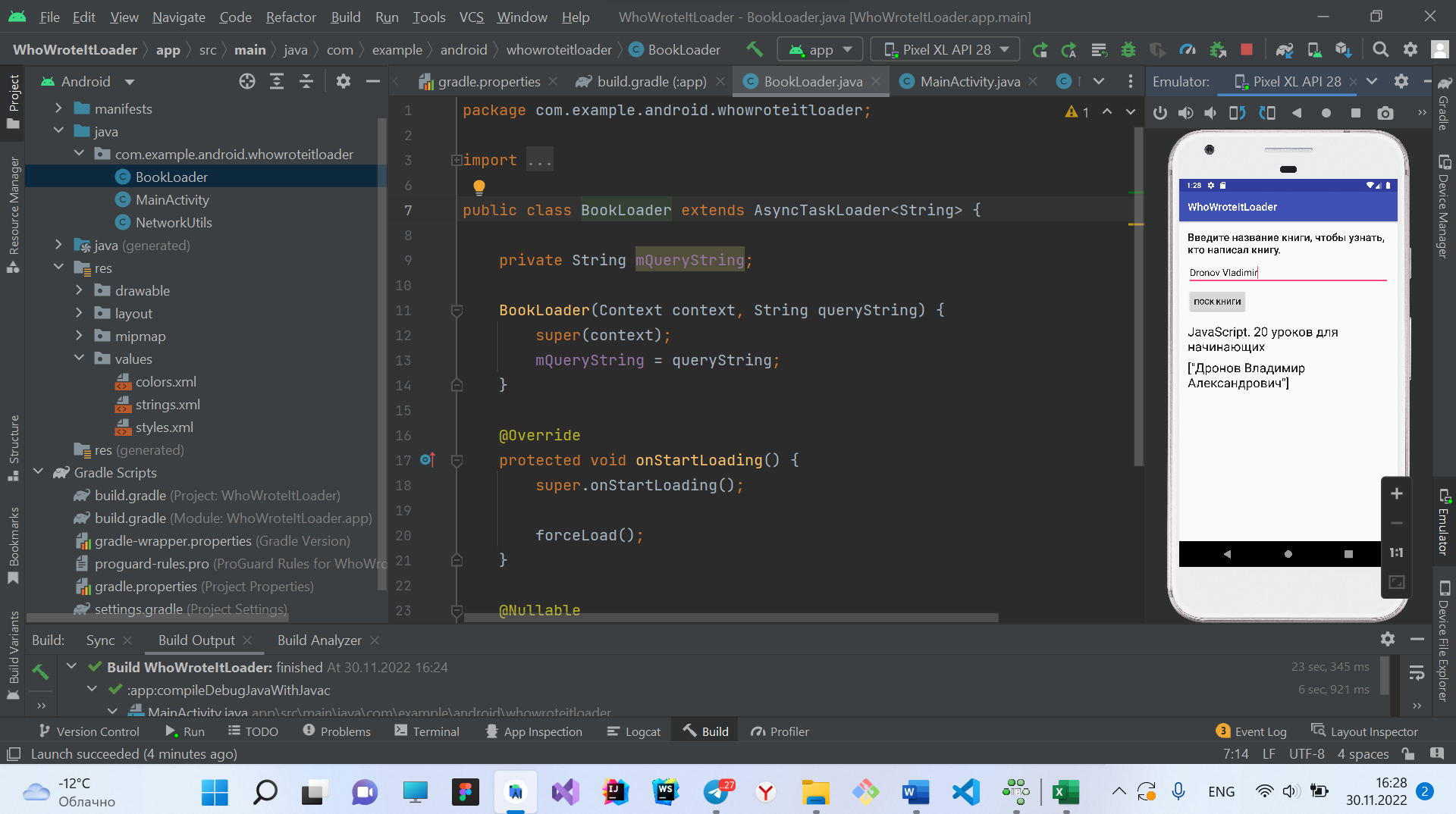


Рисунок 2 – Ответ на запрос

## **Анализ полученных результатов**

В результате выполнения данной лабораторной работы мной был разработан проект, в котором я научился использовать Google APIs Explorer для изучения API Google и просмотра ответов JSON на HTTP-запросы. Как использовать API Google books для получения данных через Интернет и поддерживать быстрый и отзывчивый пользовательский интерфейс. Вы не будете изучать API книг в деталях — ваше приложение будет использовать только простую функцию поиска книг. Как проанализировать результаты JSON из вашего запроса API. Как реализовать Async функцию, которая сохраняет данные об изменениях конфигурации. Как обновить пользовательский интерфейс с помощью обратных вызовов загрузчика.

.