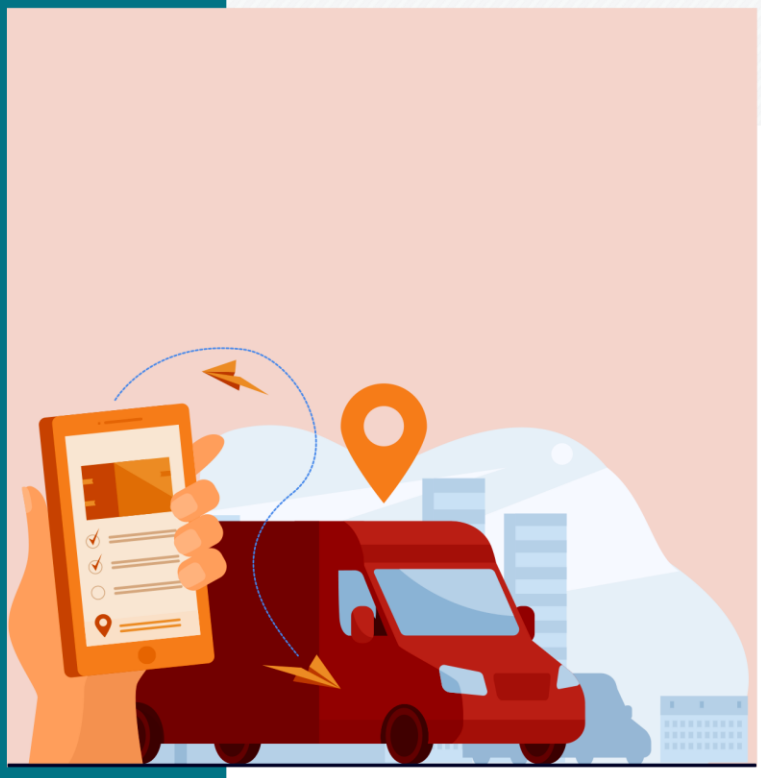
AGENDA 4

CONSUMINDO

API E SERVIÇOS WEB



MERGULHANDO NO TEMA...

Application Programming Interface (API)

A *Application Programming Interface*, ou na língua portuguesa Interface de programação de aplicações, ou ainda como é mais conhecida pela sigla API, é um conjunto de padrões previamente desenvolvidos para gerar a espécie de uma "ponte" entre dois sistemas ou mais, sejam eles produzidos na mesma ou em linguagens diferentes, permitindo assim a interação e troca de informações entre eles.

O desenvolvimento de API surgiu com a necessidade de troca de informações entre instituições e serviços, desta forma essa comunicação tinha a necessidade de uma alta segurança, não bastando apenas que uma determinada instituição fornecesse acesso ao seu banco de dados, ela precisava fornecer acesso apenas a um determinado conjunto de informações, não permitindo acesso geral.

Voltando ao exemplo dos Correios, imaginem os problemas que poderiam surgir se a empresa liberar acesso ao seu banco de dados para os usuários consultar informações sobre objetos postados. O nível de segurança cai para zero, e isso pode trazer problemas incalculáveis para a empresa.

Desta forma, os Correios desenvolveram uma API, ou seja, uma ferramenta on-line que oferece um acesso limitado as informações, o usuário não possui acesso direto ao banco de dados.

Garantindo assim que apenas as informações pertinentes a determinada consulta seja entregue ao usuário e que ele não consiga acesso a outras informações.

Encontramos várias formas de API's, e cada uma delas se enquadra na utilização de recursos entre ferramentas, veja a seguir os tipos encontrados:

Dynamic-link library (DLL): São classes criadas em linguagens de programação e distribuídas para que terceiros utilizem recursos sem a necessidade de perder tempo reescrevendo ou até mesmo recriando códigos. Podemos exemplificar falando de uma DLL que converte vários formatos de áudio para o formato MP3. Essa DLL pode ser utilizada por um sistema de gravação e edição de músicas de um desenvolvedor "A". E pode ser utilizada por um desenvolvedor "B" no seu projeto de gravação e edição de vídeos. Tanto o desenvolvedor "A" quanto o desenvolvedor "B" não criaram os códigos dessa DLL, e seus sistemas utilizam os recursos oferecidos por essas classes, facilitando o desenvolvimento dos programas.

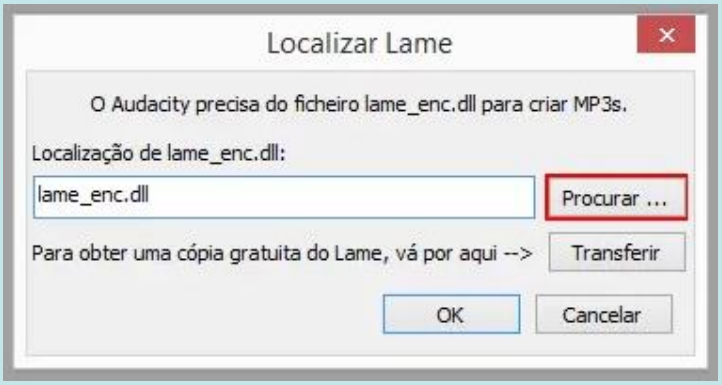


Imagem 3 – Tela do programa Audacity, solicitando a localização da DLL responsável por criar arquivos MP3.

Plugins: São encontradas na WEB e responsáveis por alguns serviços utilizados atualmente como certificação de login e senha, animação de interfaces, entre outros. Um exemplo é o "Captcha" que é responsável por atribuir uma segurança em páginas WEB, para que certos serviços não sejam executados por Robôs.

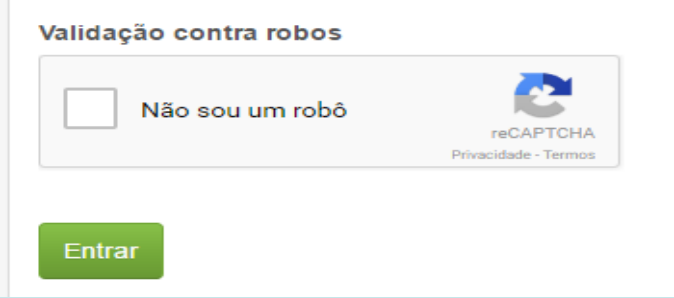


Imagem 4 – Utilização de "Captcha" em site.

Frameworks: São amplamente utilizados na WEB atual, e permitem recursos e envolvimentos visuais para sites alguns exemplos são o JQuery e Bootstrap. Como exemplo o Bootstrap permite o desenvolvimento profissional de sites com um alto nível de interface gráfica e de maneira rápida, com construção de formulários, botões, menus, entre outros componentes que valorizam o projeto.

Imagem 5 – Modelos de sites que utilizam Bootstrap na sua interface.

WebAPI: Essa última e não menos importante será o tema da nossa agenda. São API's disponíveis na WEB e que oferecem a interação entre ferramentas. São utilizadas por meio de serviços disponíveis em sites/servidores de conteúdo e para o seu funcionamento é necessário seguir alguns padrões de criação e comunicação utilizando protocolos de comunicação e códigos XML, JSON, entre outros.

Na imagem 6 encontramos um exemplo de API WEB onde é retornado à consulta de um CEP. Esta API é disponibilizada gratuitamente pelo site ViaCEP, disponível em: https://viacep.com.br/.

Essa API é responsável por receber um determinado CEP por meio da sua URL, e retornar um arquivo JSON com as informações do CEP consultado.

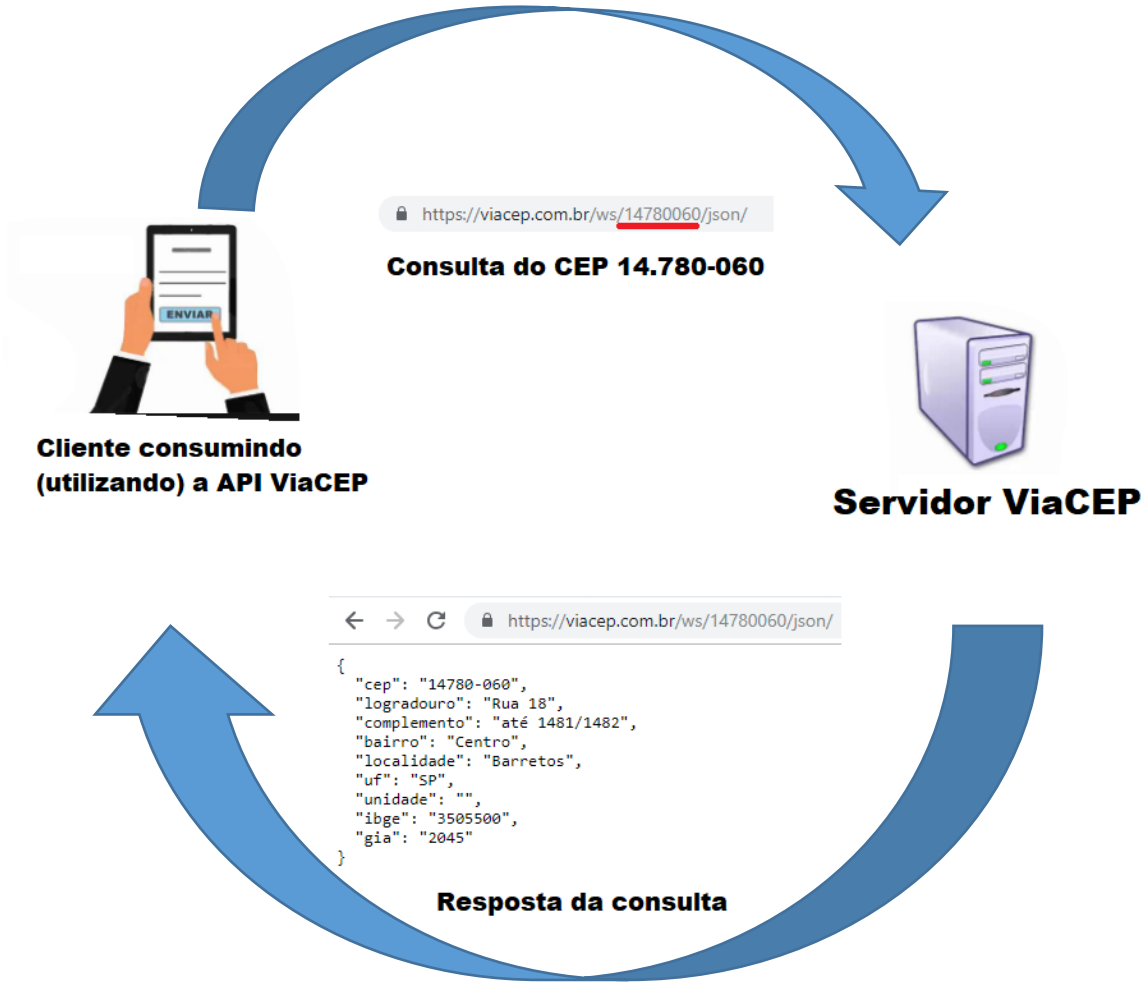


Imagem 6 – Consulta de CEP no site ViaCEP.

Na imagem anterior encontramos o retorno da consulta de maneira visual por meio do navegador, pois ele é responsável por interpretar os códigos JSON e mostrar na tela do usuário.

Vamos aprender um pouco sobre esses arquivos de retorno utilizados pela grande maioria dos serviços WEB, vamos abordar os dois principais o XML e o JSON.

Extensible Markup Language (XML)

O XML é um arquivo de marcação muito utilizado na Internet e em serviços WEB, foi desenvolvido pela W3C com o propósito de organizar textos, banco de dados, desenhos, etc. Essa organização ocorre hierarquicamente de maneira extensível, ou seja, os marcadores são customizáveis, garantindo a sua utilização nos mais diversos cenários.

O modelo XML ficou popular no Brasil, quando seus padrões foram implementados na nota fiscal eletrônica e seus arquivos eram utilizados no processo de comunicação entre as empresas e os servidores da receita para o processo de validação e conhecimento de uma nota fiscal.

Observe a imagem 7 para compreender a forma de um arquivo XML de um exemplo que guarda dados sobre clientes. A imagem mostra que os marcadores são destacados pela utilização das tag's "<>" e todo marcador aberto, tem a necessidade de ser fechado.

O padrão de arquivo XML ainda é amplamente utilizado em todo mundo, perdendo lugar aos poucos para o padrão JSON que é uma nova forma de arquivos com marcações para trânsito de informações.

```
<?xml version="1.0"?>
<cliente>
        <nome>João</nome>
        <telefone>1733253325</telefone>
        <endereco>Rua 10 nº 20</endereco>
</cliente>
<cliente>
        <nome>Maria</nome>
        <telefone>1633223322</telefone>
        <endereco>Rua 30 nº 40</endereco>
</cliente>
<cliente>
        <nome>José</nome>
        <telefone>1433223366</telefone>
        <endereco>Rua 50 nº 60</endereco>
</cliente>
```

Imagem 7 - Arquivo XML com dados de 3 clientes.

JavaScript Object Notation (JSON)

O cenário de desenvolvimento WEB atualmente conta com uma popularidade na utilização da linguagem JavaScript. Essa tendência fez com que naturalmente os programadores efetuem uma migração na utilização dos sistemas de comunicação, passando do XML para o JSON. Desta forma, a cada dia o JSON ganha mais território no mundo da programação.

Por ser um objeto proveniente do JavaScript, o JSON é automaticamente mais simples de ser utilizado na linguagem quando comparado ao XML. Pesquisas apontam que a utilização do JSON nos processos de troca de informações, são mais eficientes, uma vez que ele é mais leve que o XML.

```
[
        {"nome":"João",  "telefone":"1733253325","endereco":"Rua  10 nº 20"},
        {"nome":"Maria",  "telefone":"1633223322","endereco":"Rua  30 nº 40"},
        {"nome":"José",  "telefone":"1433223366","endereco":"Rua  50 nº 60"}
]
```

Imagem 8 – Arquivo JSON com dados de 3 clientes.

Na imagem 8 é apresentado um Array (composto pelas tag's "[]") de objetos (composto pelas tag's "{}"), e dentro dos objetos tem os dados com valores, como por exemplo, "nome":"João".

Agora que conhecemos as formas de transferência de informações pelos serviços WEB, vamos verificar como ocorre um consumo de serviço WEB.

Consumindo Serviços WEB no navegador

Várias empresas estão utilizando serviços WEB, e Serginho escolheu uma dessas empresas para utilizar em seu projeto. O site ViaCEP, proporciona uma consulta gratuita de CEP por meio de uma API. Ao consultar essa API, ela retorna um arquivo JSON com as informações do CEP. Na imagem 9, é mostrado a URL para utilizar a API, onde nesta URL é transmitido o CEP que será consultado.

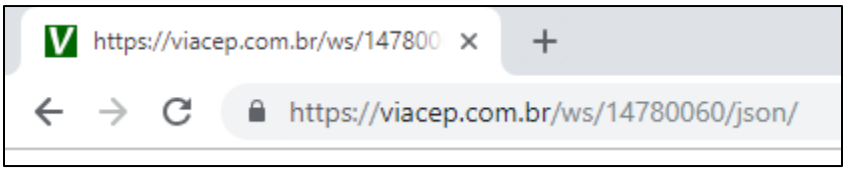
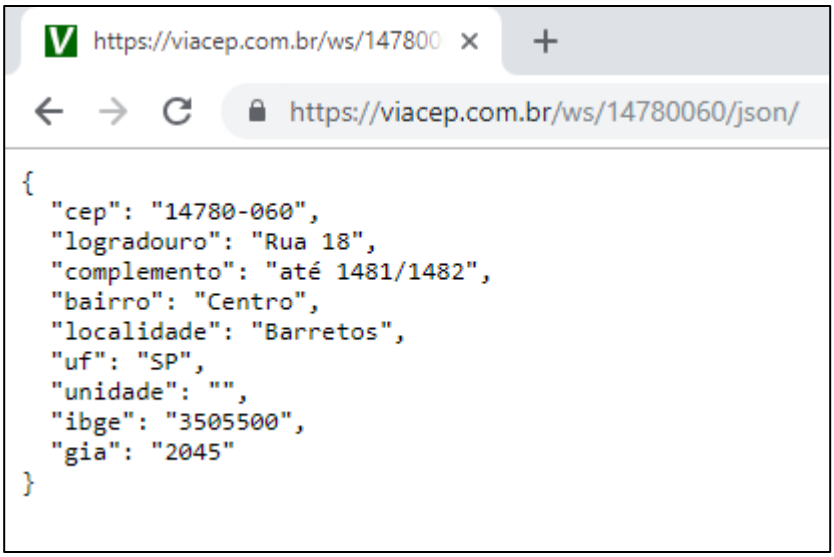


Imagem 9 – URL de consulta a API ViaCEP, buscando informações sobre o CEP: 14780-060.

Na imagem 10, mostramos o conteúdo do arquivo JSON com a resposta do CEP, exibida pelo navegador WEB que utilizamos na consulta.



Este processo, apresentado anteriormente, é o que leva o nome de consumir um serviço WEB, agora temos que estipular alguns padrões, para utilizar em nosso aplicativo.

Imagem 10 – Resposta da consulta a API ViaCEP, buscando informações sobre o CEP: 14780-060.

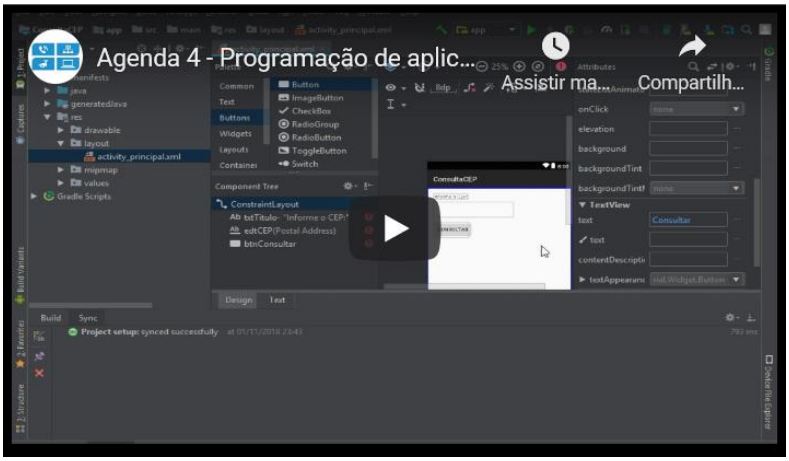
VOCÊ NO COMANDO

Consumindo Serviços WEB no aplicativo

Você já imaginou construir um aplicativo para consumir um serviço WEB? Não? Então vamos seguir juntos nessa jornada com nosso desenvolvedor Serginho.

Para consumir um serviço WEB em um aplicativo, é necessário criar uma estrutura de classes para conectar ao serviço e receber o objeto JSON. E por fim, temos que criar a estrutura de classes para efetuar a interface gráfica com o usuário, ou seja, receber o CEP e mostrar as informações para o usuário.

Vamos criar um projeto chamado “ConsultaCEP”, verifique no vídeo a seguir o processo de criação do projeto.

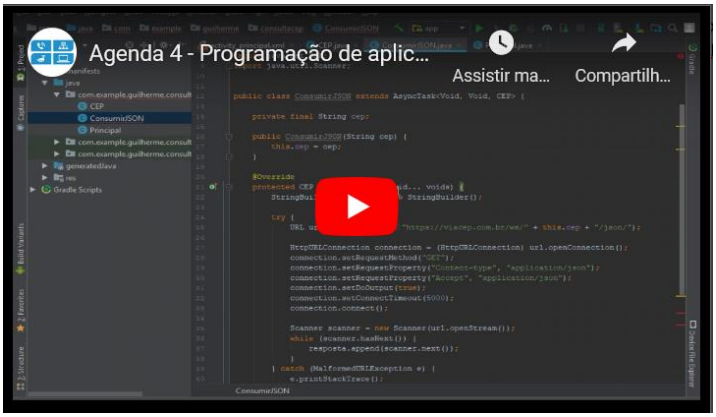


Após criar o projeto vamos criar a classe que receberá as informações do objeto JSON, obtido na API de consulta do site ViaCEP. Vamos observar os campos utilizados pela API do site, e preparar a nossa classe com os mesmos nomes. Essa nova classe chamará “CEP”.

```
package com.example.guilherme.consultacep;

public class CEP {
    private String logradouro;
    private String bairro;
    private String localidade;
    private String uf;


    public String getLogradouro() {
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getBairro() {
        return bairro;
    }

    public void setBairro(String bairro) {
        this.bairro = bairro;
    }

    public String getLocalidade() {
        return localidade;
    }

    public void setLocalidade(String localidade) {
        this.localidade = localidade;
    }

    public String getUf() {
        return uf;
    }

    public void setUf(String uf) {
        this.uf = uf;
    }

    @Override
    public String toString() {
        return
                "Logradouro: " + getLogradouro()
                + "\nBairro: " + getBairro()
                + "\nCidade:" + getLocalidade()
                + "\nEstado: " + getUf();
    }
}
```

Código Classe “CEP”.

Para comunicar com o serviço WEB, vamos criar uma classe chamada “ConsumirJSON”, essa classe será responsável em se conectar com a API por meio do endereço URL do site ViaCEP, e depois converter o arquivo JSON, obtido na conexão em um objeto construído por meio da classe “CEP” criada anteriormente.



A classe “ConsumirJSON” é uma extensão da classe “AsyncTask”, desenvolvida para trabalhar em segundo plano, ou seja, todo serviço que necessita rodar em um segundo plano ou realizar uma determinada ação durante um intervalo de tempo nos bastidores da aplicação utilizam essa classe, uma vez que ela já vem com todos os métodos necessários para essa função.

Código Classe “ConsumirJSON”.

A estrutura da “AsyncTask” é dividida em:

```
public class ConsumirJSON extends AsyncTask <Parâmetros,Progresso,Resultado> {


}
```

Os parâmetros são valores utilizados pelos métodos da classe. O progresso é um valor do tipo inteiro informado para passar o status do trabalho. E o resultado é o retorno do processamento.

No caso, passamos como parâmetros e o progresso o argumento “Void”. E para o retorno do processamento utilizamos uma classe do tipo “CEP”, ou seja, vamos utilizar “CEP” para armazenar o resultado da consulta.

```
public class ConsumirJSON extends AsyncTask <Void, Void, CEP> {


}
```

A classe “AsyncTask” possui um método que vamos sobrescrever, responsável pela execução da consulta em segundo plano. Acompanhe no vídeo a seguir o desenvolvimento da classe “ConsumirCEP”.

```
package com.example.guilherme.consultacep;

import android.os.AsyncTask;
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Scanner;
import com.google.gson.Gson;

public class ConsumirJSON extends AsyncTask<Void, Void, CEP> {

    private final String cep;
```

```
    public ConsumirJSON(String cep) {
        this.cep = cep;
    }

    @Override
    protected CEP doInBackground(Void... voids) {
        StringBuilder resposta = new StringBuilder();

        try {
            URL url = new URL("https://viacep.com.br/ws/" + this.cep + "/json/");

            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setRequestMethod("GET");
            connection.setRequestProperty("Content-type", "application/json");
            connection.setRequestProperty("Accept", "application/json");
            connection.setDoOutput(true);
            connection.setConnectTimeout(5000);
            connection.connect();

            Scanner scanner = new Scanner(url.openStream());
            while (scanner.hasNext()) {
                resposta.append(scanner.next());
            }
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        Gson gs = new Gson();

        return gs.fromJson(resposta.toString(), CEP.class);

    }
}
```

Importante:

É importante lembrar de incluir a dependência em **Gradle Scripts > build.gradle (Module: App).**


```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "com.example.guilherme.consultacep"
        minSdkVersion 24
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-
core:3.0.2'
    implementation 'com.google.code.gson:gson:2.8.2'
}
```

Código Gradle Scripts > build.gradle (Module: App).

Para finalizar o aplicativo, vamos desenvolver a classe “Principal” do projeto.

```
package com.example.guilherme.consultacep;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import java.util.concurrent.ExecutionException;

public class Principal extends Activity {

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

        Button btnConsultarProg = (Button) findViewById(R.id.btnConsultar);
        final EditText cep = findViewById(R.id.edtCEP);
        final TextView resultado = findViewById(R.id.txtResultado);

        btnConsultarProg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    CEP retorno = new CEP();
                    ConsumirJSON cj = new ConsumirJSON(cep.getText().toString());
                    retorno = cj.execute().get();
                    resultado.setText(retorno.toString());
                } catch (InterruptedException e) {
                    e.printStackTrace();
                } catch (ExecutionException e) {
                    e.printStackTrace();
                }
            }
        });
    }
}
```

Código Classe “Principal”.

Importante:

É importante lembrar de incluir a permissão de acesso à Internet no dispositivo Mobile, por meio do arquivo **Manifests > AndroidManifest.xml.**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.guilherme.consultacep">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".Principal">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.INTERNET"/>
</manifest>
```

Código Manifests > AndroidManifest.xml.