

android-retrofit

Sumário

1. Informativo	3
2. Objetivo	4
3. Pré-Requisitos.....	5
4. Versão.....	6
4.1. minSdkVersion	6
4.2. targetSdkVersion	6
4.3. Android Studio	6
5. Retrofit.....	7
6. Novo Projeto.....	8
7. Ajustando nosso Layout.....	12
8. Configuração do Retrofit	14
9. Configuração do Serviço	18
10. Definindo o Modelo.....	20
11. Realizando a chamada	22
12. Resumo	23
13. Referências	24

1. Informativo

Autor(a): Helena Strada

Data de Criação: jan/2018.

2. Objetivo

O objetivo desta apostila é mostrar as funcionalidades do Retrofit e criar uma aplicação de exemplo – buscar cep.

3. Pré-Requisitos

Java (Orientação a Objetos, APIs e Bibliotecas).

4. Versão

4.1. minSdkVersion

15

4.2. targetSdkVersion

27

4.3. Android Studio

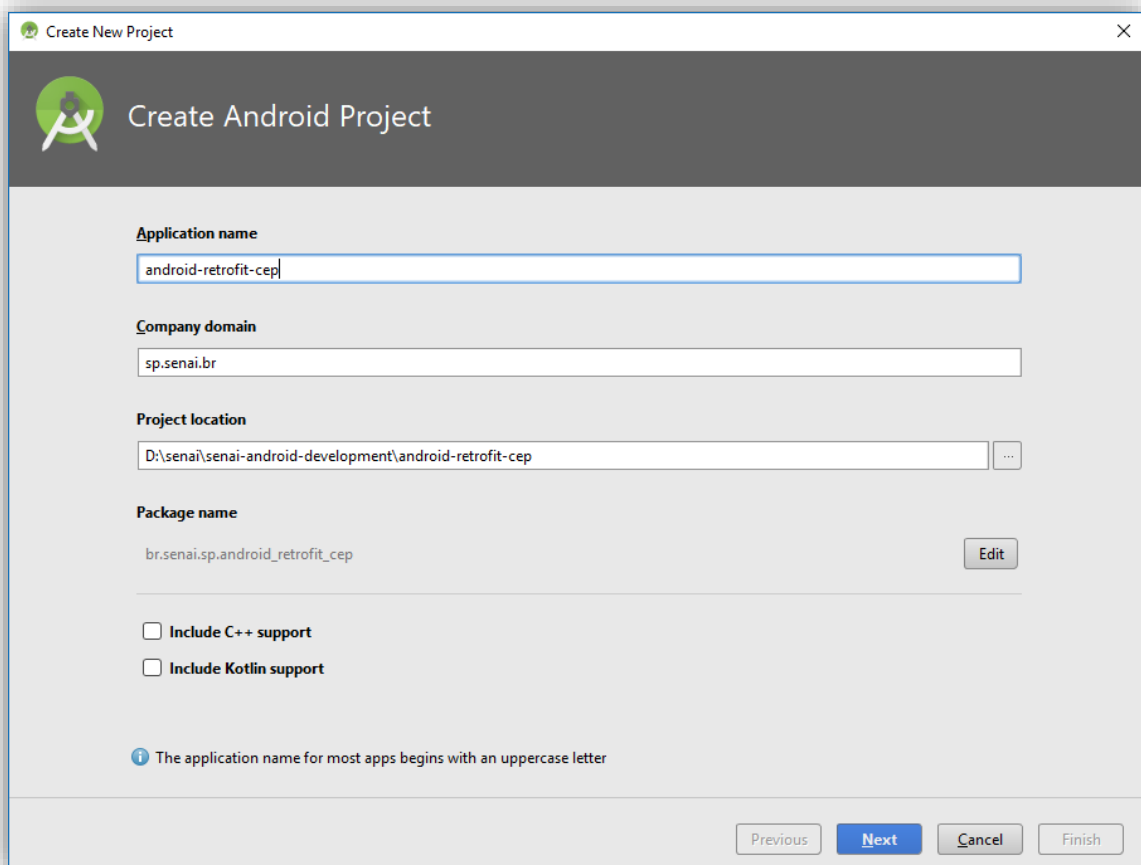
3.0.0

5. Retrofit

O Retrofit é uma biblioteca *open source* que facilita o processo de realização de requisições. Além disso, não precisamos fazer a serialização manual dos objetos. Nós precisamos apenas definir a conexão que desejamos fazer.

6. Novo Projeto

Vamos começar criando um novo projeto chamado android-retrofit-cep.




The screenshot shows the 'Create New Project' dialog in Android Studio. The dialog has a title bar 'Create New Project' and a close button. The main area is titled 'Create Android Project' and contains the following fields and options:

- Application name:** A text field containing 'android-retrofit-cep'.
- Company domain:** A text field containing 'sp.senai.br'.
- Project location:** A text field containing 'D:\senai\senai-android-development\android-retrofit-cep' with a browse button (three dots) to its right.
- Package name:** A text field containing 'br.senai.sp.android_retrofit_cep' with an 'Edit' button to its right.
- Include C++ support:** A checkbox that is unchecked.
- Include Kotlin support:** A checkbox that is unchecked.
- Help text:** A small information icon followed by the text 'The application name for most apps begins with an uppercase letter'.
- Navigation buttons:** At the bottom right, there are four buttons: 'Previous' (disabled), 'Next' (active/highlighted), 'Cancel', and 'Finish' (disabled).

Clicar em Next.

Create New Project

Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☒ **Phone and Tablet**
API 15: Android 4.0.3 (IceCreamSandwich)
By targeting **API 15 and later**, your app will run on approximately **100%** of devices. [Help me choose](#)
☐ Include Android Instant App support

☐ **Wear**
API 21: Android 5.0 (Lollipop)

☐ **TV**
API 21: Android 5.0 (Lollipop)

☐ **Android Auto**

☐ **Android Things**
API 24: Android 7.0 (Nougat)

Previous

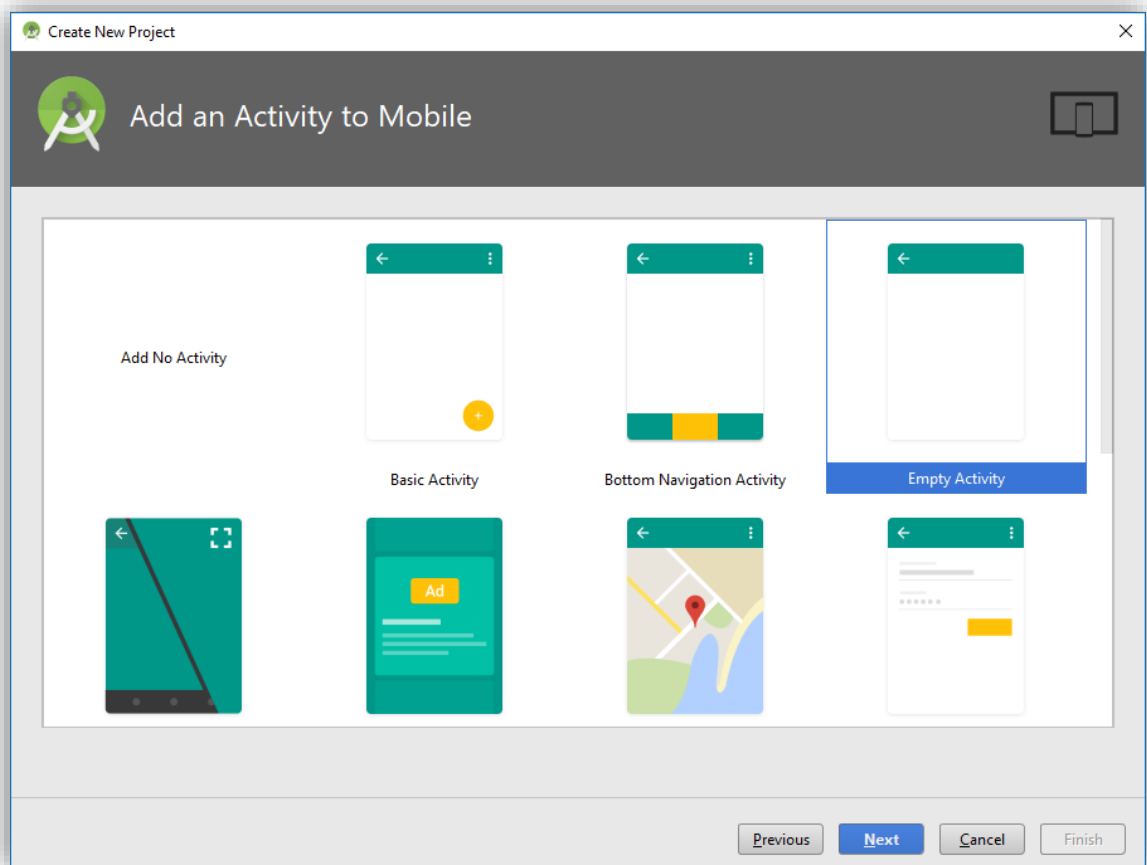
Next

Cancel

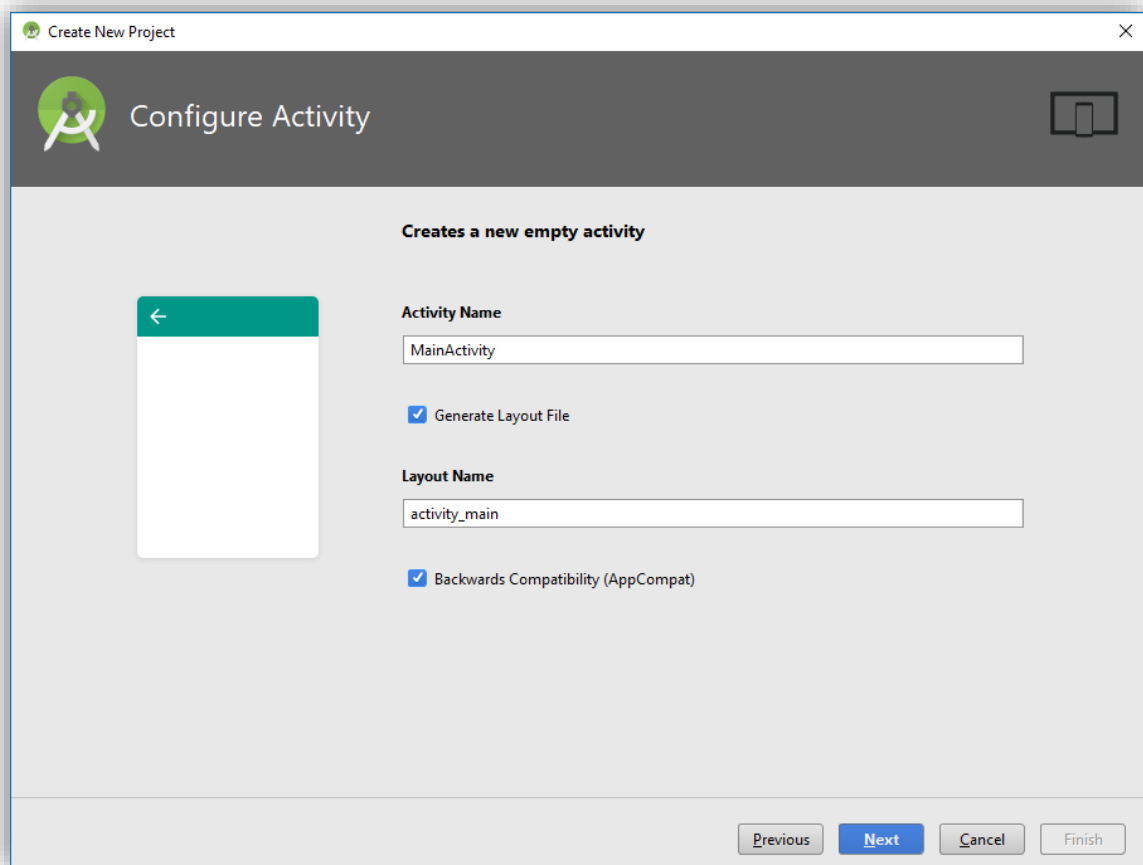
Finish

Clicar em Next.

Vamos criar novo projeto como “Empty Activity”.



Clicar em Next.



Clicar em Next.

7. Ajustando nosso Layout

Vamos ter um campo texto de entrada para o usuário. Este campo irá servir para o usuário digitar o cep que desejar. Um botão de busca. E, abaixo, um TextView simples apenas para aparecer as informações para o usuário.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"

tools:context="br.senai.sp.android_retrofit_sem_autenticacao.MainActivity"
tools:showIn="@layout/activity_main">

    <android.support.design.widget.TextInputLayout
        android:id="@+id/tilCep"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:baselineAligned="false"
            android:hint="CEP (12345-000)"
            android:inputType="number" />

    </android.support.design.widget.TextInputLayout>

    <Button
        android:id="@+id/btnBuscarCep"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Buscar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tilCep" />

    <TextView
        android:id="@+id/tvResultado"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        app:layout_constraintTop_toBottomOf="@+id/btnBuscarCep" />

</android.support.constraint.ConstraintLayout>
```

Já vamos deixar na nossa MainActivity.java, as referências criadas.

```
private TextInputLayout tilCep;  
private Button btnBuscar;  
private TextView tvResultado;
```

E dentro do nosso método onCreate():

```
tilCep = findViewById(R.id.tilCep);  
tvResultado = findViewById(R.id.tvResultado);  
btnBuscar = findViewById(R.id.btnBuscarCep);  
  
btnBuscar.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
  
    }  
});
```

A nossa chamada do Retrofit ficará dentro do botão.

8. Configuração do Retrofit

Antes de iniciarmos o passo-a-passo, vamos importar a biblioteca do Retrofit no nosso build.gradle no nível da nossa aplicação.

```
compile 'com.squareup.retrofit2:retrofit:2.3.0'
```

Vamos criar um novo pacote chamado config e dentro dele uma classe chamada RetrofitConfig.

```
/*
 *
 * Essa classe ficará responsável por configurar e instanciar o Retrofit
 *
 * */
public class RetrofitConfig {

}
```

Nossa configuração ficará na inicialização dessa classe. Poderíamos também, sempre que precisássemos realizar uma nova chamada a uma API Rest, realizar a criação de um novo objeto do tipo Retrofit e fazer a chamada. Veremos os dois exemplos.

```
/*
 *
 * Essa classe ficará responsável por configurar e instanciar o Retrofit
 *
 * */
public class RetrofitConfig {

    // Nossa configuração será feita no construtor
    public RetrofitConfig() {

    }

}
```

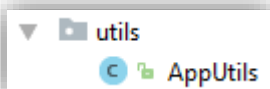
O `Builder()` permite construir um novo objeto do tipo Retrofit.

```
// Nossa configuração será feita no construtor
public RetrofitConfig() {

    // Precisamos construir um objeto do tipo retrofit
    new Retrofit.Builder();

}
```

Porém, nós precisamos definir pra ele qual a URL que desejamos chamar. Vamos criar um novo pacote chamado `utils` e dentro dele uma classe chamada `AppUtils.java`. Ela ficará responsável por conter alguns parâmetros que utilizaremos para a aplicação inteira.



Dentro de `AppUtils`, vamos definir a nossa URL Base.

```
public class AppUtils {

    public final static String BASE_URL = "http://api.postmon.com.br/v1/cep/";

}
```

Para realizar a busca de CEP, estaremos utilizando a URL:

`http://api.postmon.com.br/v1/cep/`

```
// Nossa configuração será feita no construtor
public RetrofitConfig() {

    // Precisamos construir um objeto do tipo retrofit
    new Retrofit.Builder()
        // definimos a url base da nossa aplicação
        .baseUrl(BASE_URL);

}
```

Uma vez definida a nossa URL, precisamos informar qual o conversor que desejamos utilizar para pegar a resposta que receberemos em JSON e transformá-la para String. Neste exemplo, utilizaremos o Jackson. Em outros exemplos, utilizaremos o GSON.

```
// Nossa configuração será feita no construtor
public RetrofitConfig() {

    // Precisamos construir um objeto do tipo retrofit
    new Retrofit.Builder()
        // definimos a url base da nossa aplicação
        .baseUrl(BASE_URL)
        // precisamos transformar a nossa resposta que vem em JSON para String
        .addConverterFactory(JacksonConverterFactory.create());

}
```

Lembrando: precisamos importar no nosso build.gradle da nossa aplicação a lib.

compile 'com.squareup.retrofit2:converter-jackson:2.3.0'

E para criarmos de fato o objeto, .build().

```
// Nossa configuração será feita no construtor
public RetrofitConfig() {

    // Precisamos construir um objeto do tipo retrofit
    new Retrofit.Builder()
        // definimos a url base da nossa aplicação
        .baseUrl(BASE_URL)
        // precisamos transformar a nossa resposta que vem em JSON para String
        .addConverterFactory(JacksonConverterFactory.create())
        // precisamos de fato criá-lo
        .build();

}
```

Classe final da RetrofitConfig.


```
/*
 *
 * Essa classe ficará responsável por configurar e instanciar o Retrofit
 *
 * */
public class RetrofitConfig {

    // atributo
    private final Retrofit retrofit;

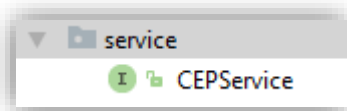
    // Nossa configuração será feita no construtor
    public RetrofitConfig() {

        // Precisamos construir um objeto do tipo retrofit
        this.retrofit = new Retrofit.Builder()
            // definimos a url base da nossa aplicação
            .baseUrl(BASE_URL)
            // precisamos transformar a nossa resposta que vem em JSON para String
            .addConverterFactory(JacksonConverterFactory.create())
            // precisamos de fato criá-lo
            .build();
    }
}
```

9. Configuração do Serviço

Vamos criar uma classe de Serviço (veremos um exemplo utilizando a denominação Service e outro exemplo com Interface).

Criar um novo pacote chamado service e criar uma nova classe chamada CEPService dentro deste pacote.



Dentro dessa classe, iremos informar as seguintes características:

- O nome da requisição que desejamos fazer;
- Qual verbo HTTP que desejamos utilizar;
- Definir a URL (<http://api.postmon.com.br/v1/cep/00100000>);
 - E, no caso, se temos parâmetro;
- E definir o retorno;
 - Call é uma classe genérica. Informamos pra ela o tipo de retorno que desejamos.

```
public interface CEPService {  
  
    @GET("{cep}")  
    Call<CEP> buscarCEP(@Path("cep") String cep);  
  
}
```

Uma vez que criamos o nosso serviço, vamos colocar no nosso RetrofitConfig, para podermos utilizá-lo.

```

/*
 *
 * Essa classe ficará responsável por configurar e instanciar o Retrofit
 *
 */
public class RetrofitConfig {

    // atributo
    private final Retrofit retrofit;

    // Nossa configuração será feita no construtor
    public RetrofitConfig() {

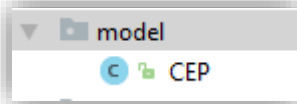
        // Precisamos construir um objeto do tipo retrofit
        this.retrofit = new Retrofit.Builder()
            // definimos a url base da nossa aplicação
            .baseUrl(BASE_URL)
            // precisamos transformar a nossa resposta que vem em JSON para String
            .addConverterFactory(JacksonConverterFactory.create())
            // precisamos de fato criá-lo
            .build();
    }

    public CEPService getCEPService() {
        return this.retrofit.create(CEPService.class);
    }
}

```

10. Definindo o Modelo

Uma vez que definimos o nosso retorno, precisamos criar a nossa classe de Modelo.



Vamos criar um novo pacote chamado model e dentro dele, uma nova classe chamada CEP.

```
@JsonIgnoreProperties({"estado_info", "cidade_info"})
public class CEP {

    private String bairro;
    private String cidade;
    private String logradouro;
    private String cep;
    private String estado;

    @Override
    public String toString() {
        return "CEP{" +
            "bairro='" + bairro + '\'' +
            ", cidade='" + cidade + '\'' +
            ", logradouro='" + logradouro + '\'' +
            ", cep='" + cep + '\'' +
            ", estado='" + estado + '\'' +
            '}';
    }

    public String getBairro() { return bairro; }

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

    public String getCidade() { return cidade; }

    public void setCidade(String cidade) { this.cidade = cidade; }

    public String getLogradouro() { return logradouro; }

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

    public String getCep() { return cep; }

    public void setCep(String cep) { this.cep = cep; }

    public String getEstado() { return estado; }

    public void setEstado(String estado) { this.estado = estado; }
}
```

Como no retorno da resposta do postmon, nós recebemos algumas propriedades a mais do que desejamos, colocamos essa propriedade no início do nosso modelo, pois estas duas propriedades não estão presentes no nosso modelo.

```
@JsonIgnoreProperties({"estado_info", "cidade_info"})
```

11. Realizando a chamada

Uma vez que definimos o nosso layout, a criação do nosso objeto Retrofit, a nossa classe de Serviço e a nossa classe de modelo, precisamos apenas colocar a chamada dentro do nosso botão de buscar.

```
private TextInputLayout tilCep;
private Button btnBuscar;
private TextView tvResultado;

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

    tilCep = findViewById(R.id.tilCep);
    tvResultado = findViewById(R.id.tvResultado);
    btnBuscar = findViewById(R.id.btnBuscarCep);

    btnBuscar.setOnClickListener((view) -> {

        // apenas configuramos, queremos agora realizar a requisição e recebermos a resposta
        Call<CEP> call = new RetrofitConfig().getCEPService().buscarCEP(tilCep.getEditText().getText().toString());
        // classe anônima do tipo callback
        call.enqueue(new Callback<CEP>() {
            @Override
            public void onResponse(Call<CEP> call, Response<CEP> response) {
                if (response.isSuccessful()) {
                    CEP cep = response.body();
                    tvResultado.setText(cep.toString());
                }
            }

            @Override
            public void onFailure(Call<CEP> call, Throwable t) {
                Toast.makeText(getApplicationContext(), text: "Não foi possível realizar a chamada.", Toast.LENGTH_LONG).show();
            }
        });
    });
}
```

12. Resumo

Aplicação criada e conceitos sobre o Retrofit.

13. Referências

<https://guides.codepath.com/android#getting-started>

<http://square.github.io/retrofit/>