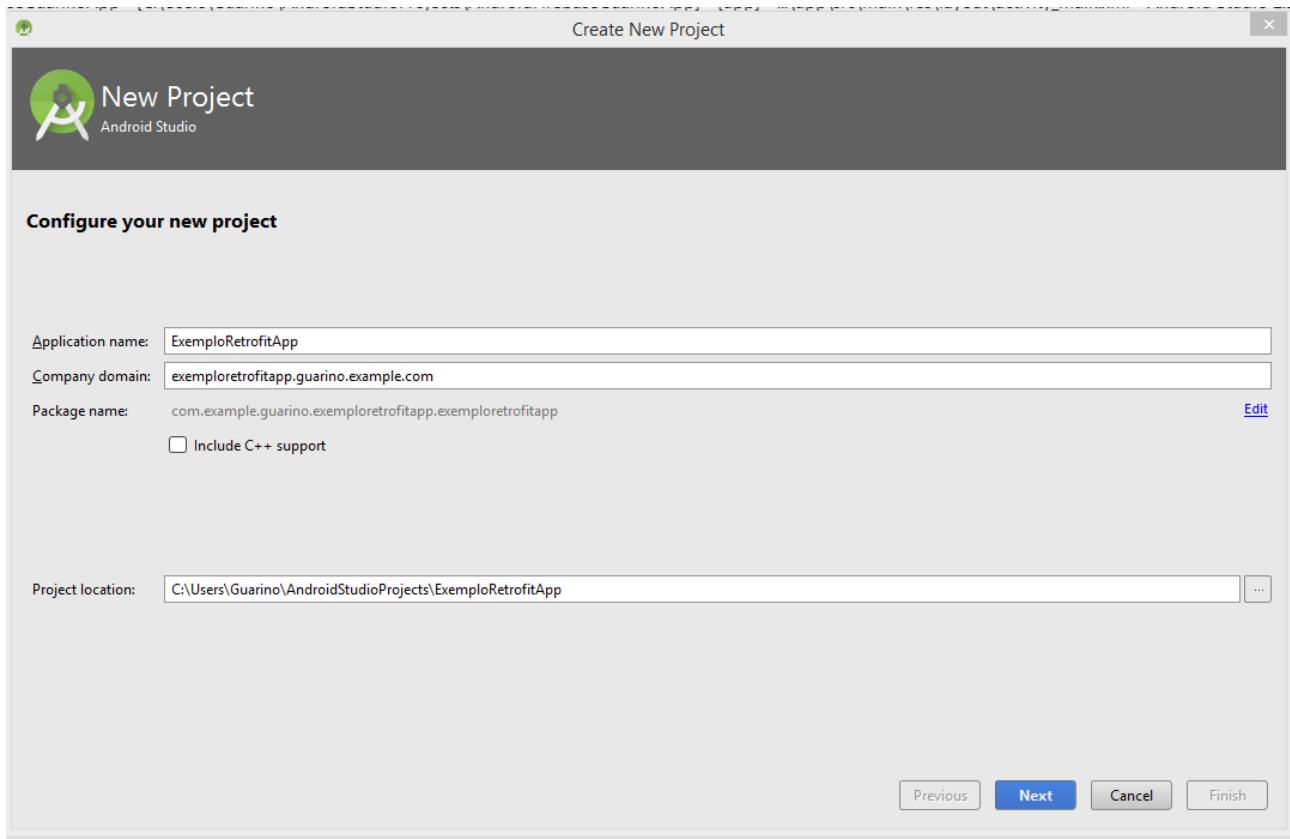


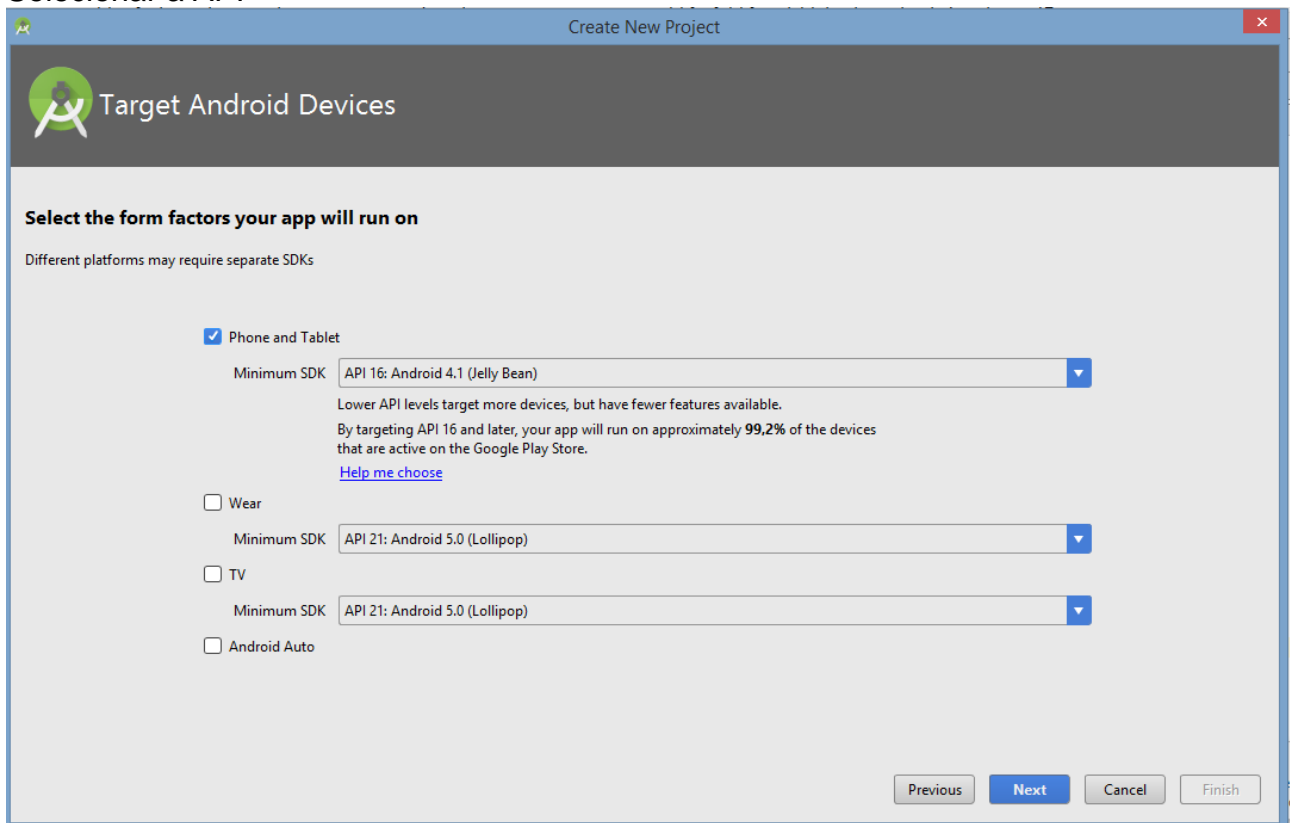
Retrofit

Criar um novo projeto



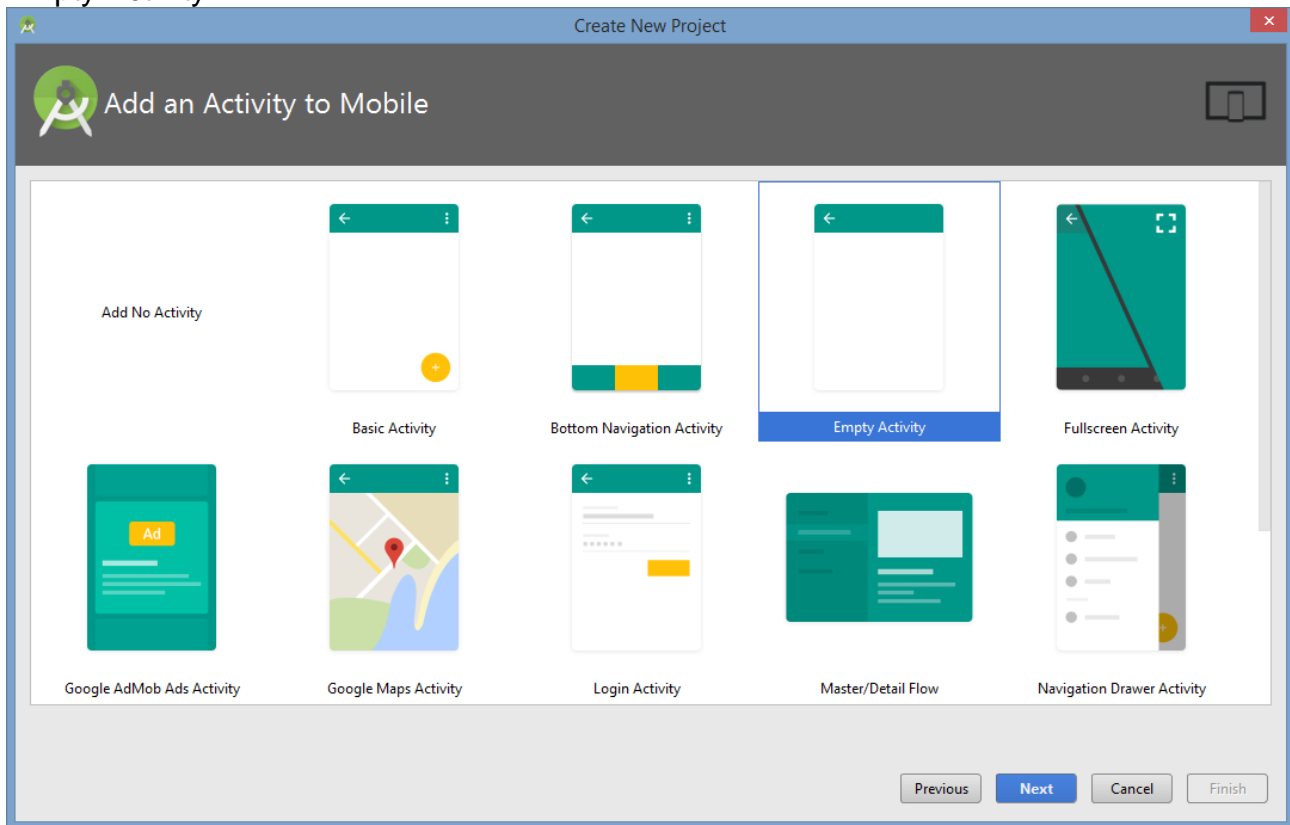
The screenshot shows the 'Create New Project' dialog in Android Studio. The title bar says 'Create New Project'. The main header area has the Android Studio logo and the text 'New Project' and 'Android Studio'. Below this, the section 'Configure your new project' is visible. It contains several input fields: 'Application name' with the value 'ExemploRetrofitApp', 'Company domain' with 'exemploretrofitapp.guarino.example.com', and 'Package name' with 'com.example.guarino.exemploretrofitapp.exemploretrofitapp'. There is an 'Include C++ support' checkbox which is unchecked. A 'Project location' field shows the path 'C:\Users\Guarino\AndroidStudioProjects\ExemploRetrofitApp'. At the bottom right, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

Selecionar a API

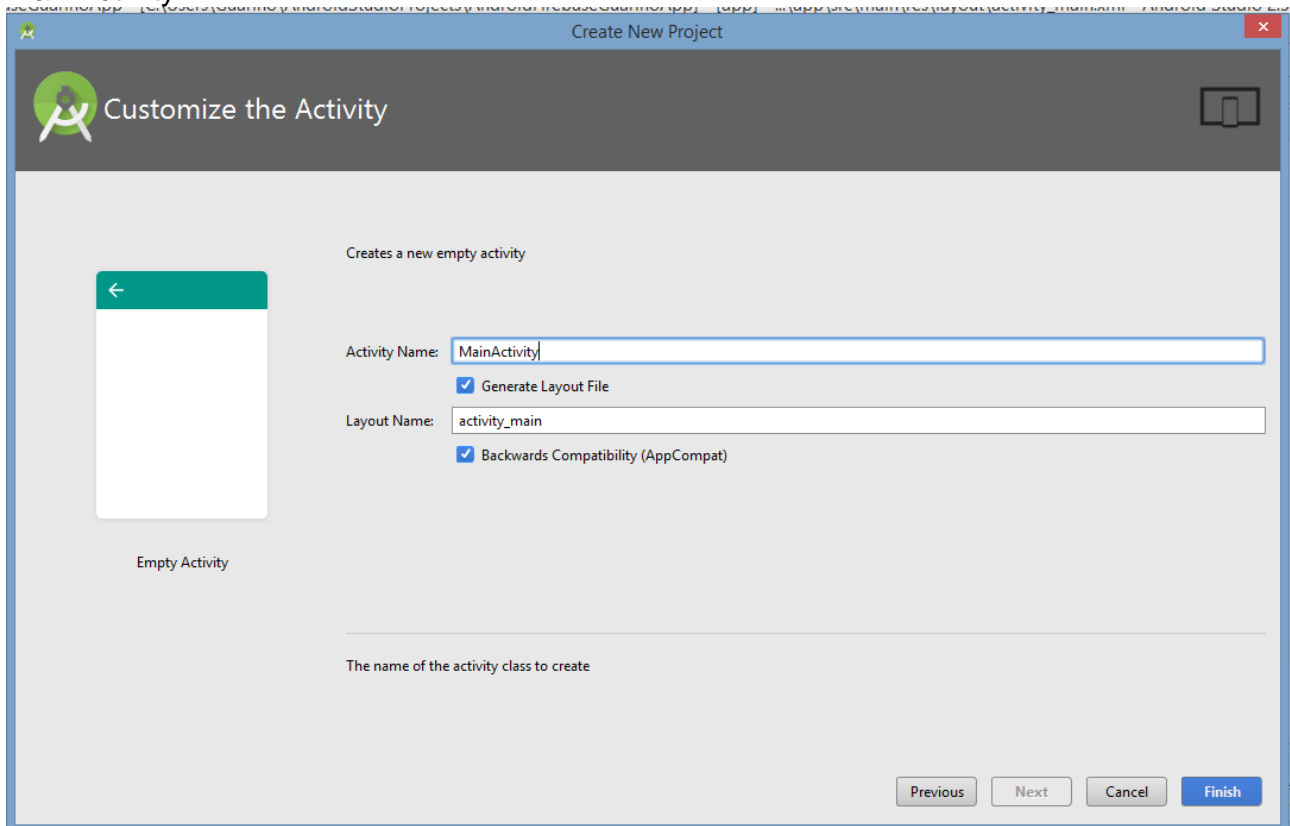


The screenshot shows the 'Target Android Devices' dialog in Android Studio. The title bar says 'Create New Project'. The main header area has the Android Studio logo and the text 'Target Android Devices'. Below this, the section 'Select the form factors your app will run on' is visible, with a subtitle 'Different platforms may require separate SDKs'. There are four options for form factors: 'Phone and Tablet' (checked), 'Wear', 'TV', and 'Android Auto'. Each option has a 'Minimum SDK' dropdown menu. For 'Phone and Tablet', the dropdown shows 'API 16: Android 4.1 (Jelly Bean)'. Below this dropdown, there is a note: 'Lower API levels target more devices, but have fewer features available. By targeting API 16 and later, your app will run on approximately 99,2% of the devices that are active on the Google Play Store.' and a link 'Help me choose'. The other three options ('Wear', 'TV', 'Android Auto') have their 'Minimum SDK' dropdowns set to 'API 21: Android 5.0 (Lollipop)'. At the bottom right, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

Empty Activity



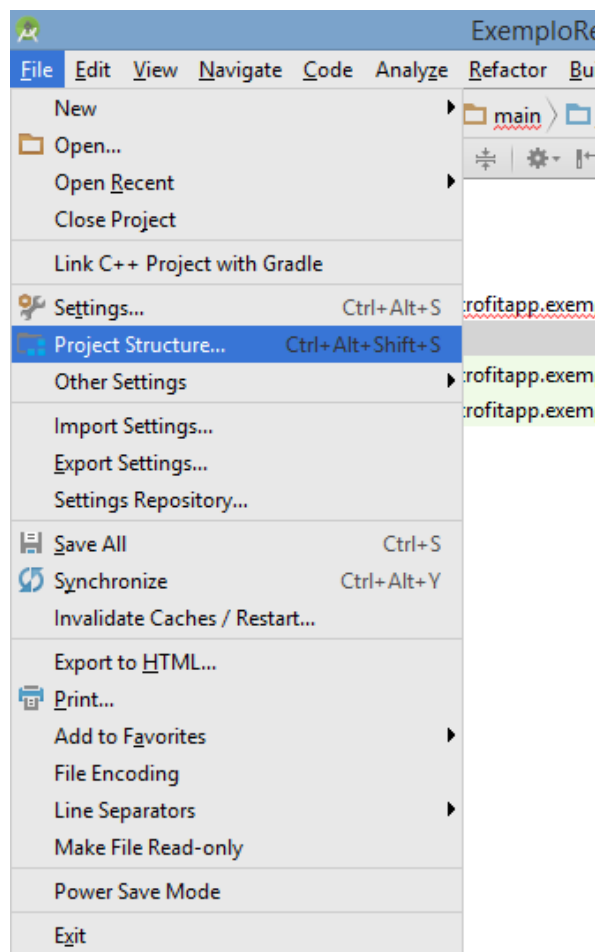
MainActivity



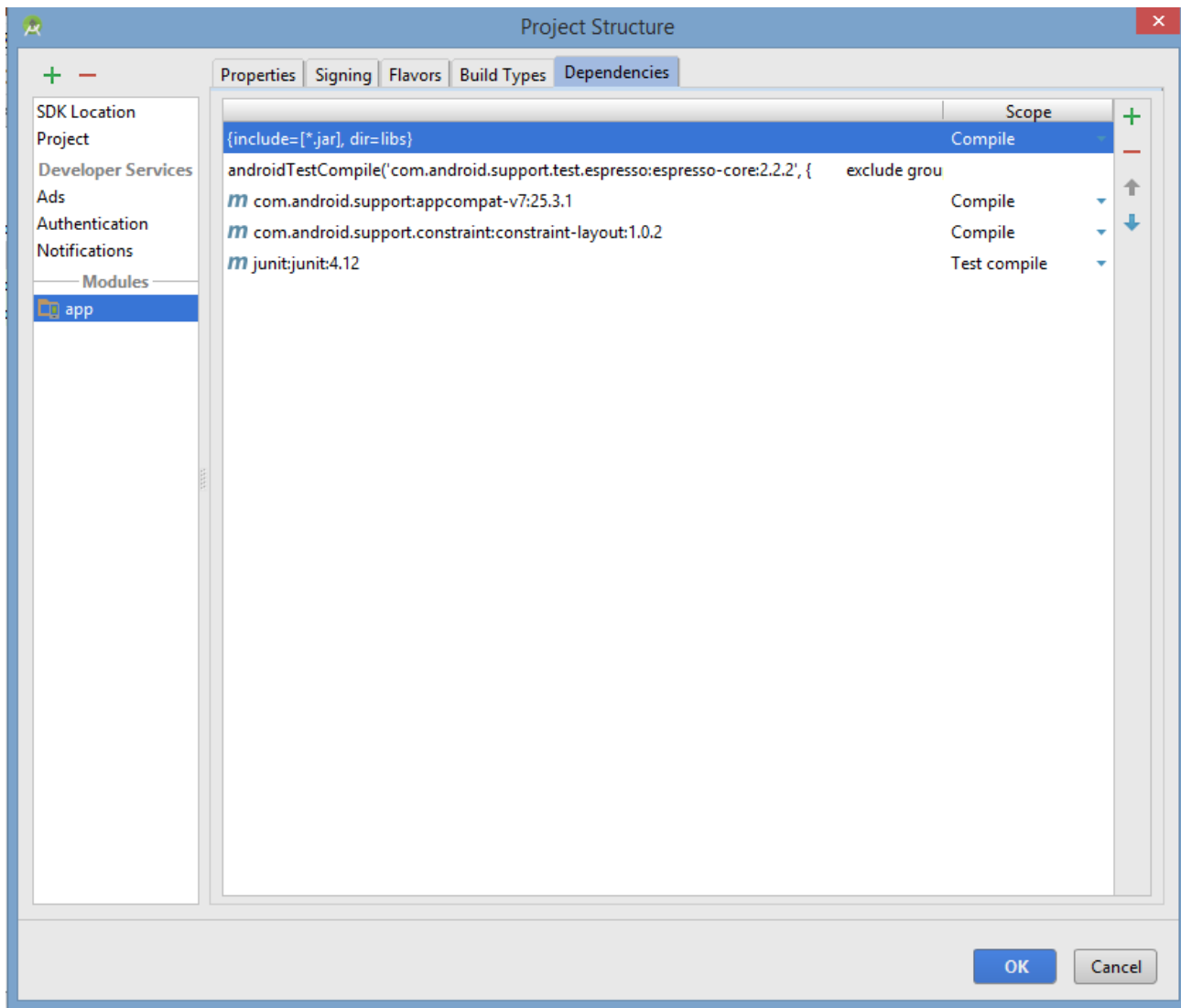
Finish.

Adicionar Novas Dependências

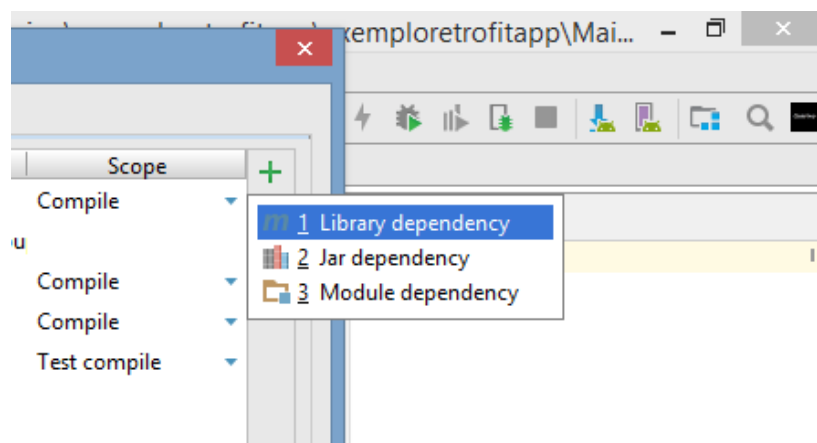
Menu File > Project Structure



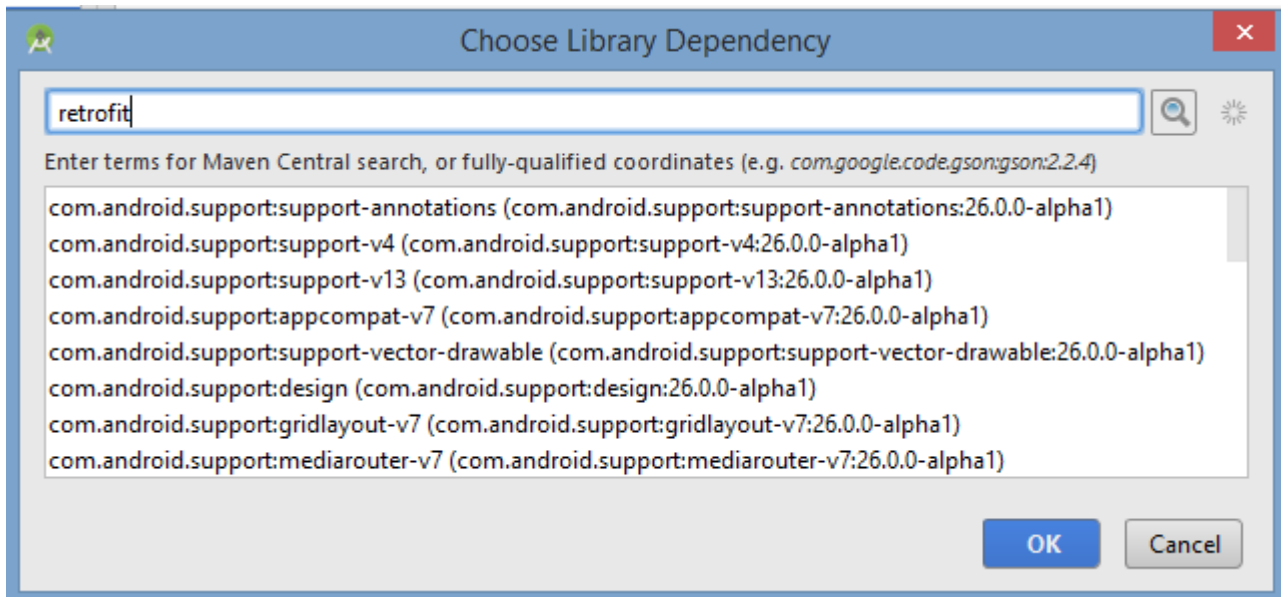
Modulo app > Aba Dependencies



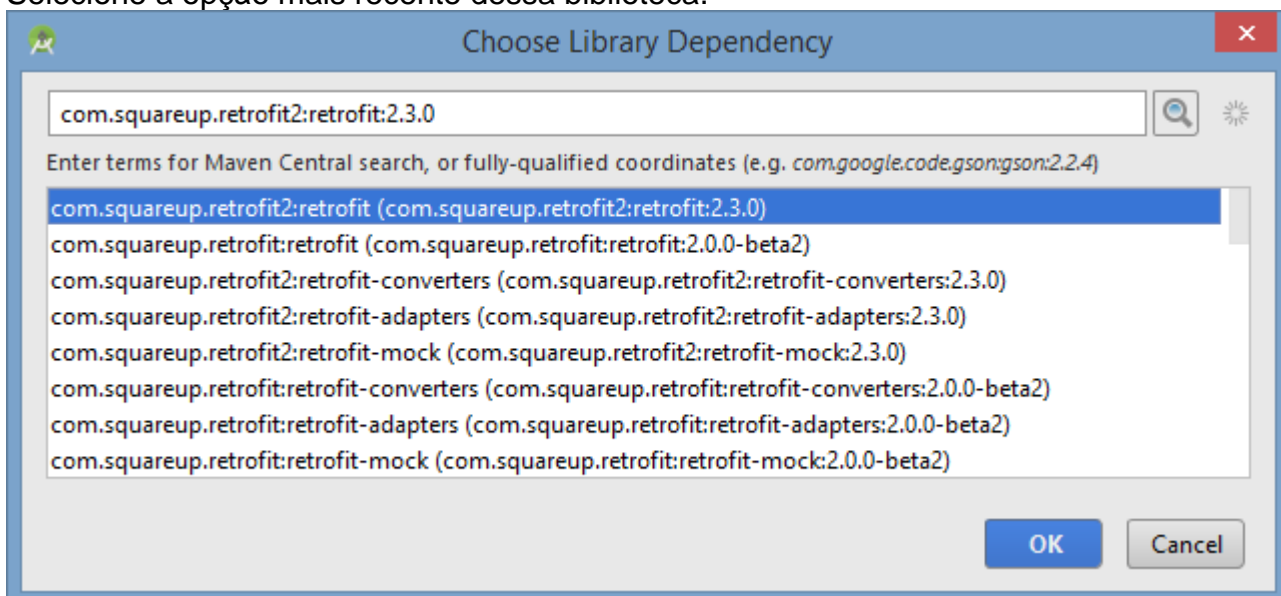
Clique em adicionar (+) no canto superior direito. Depois, clique em Library dependency.



Pesquise por retrofit. Pressione Enter ou clique no botão de Pesquisa.

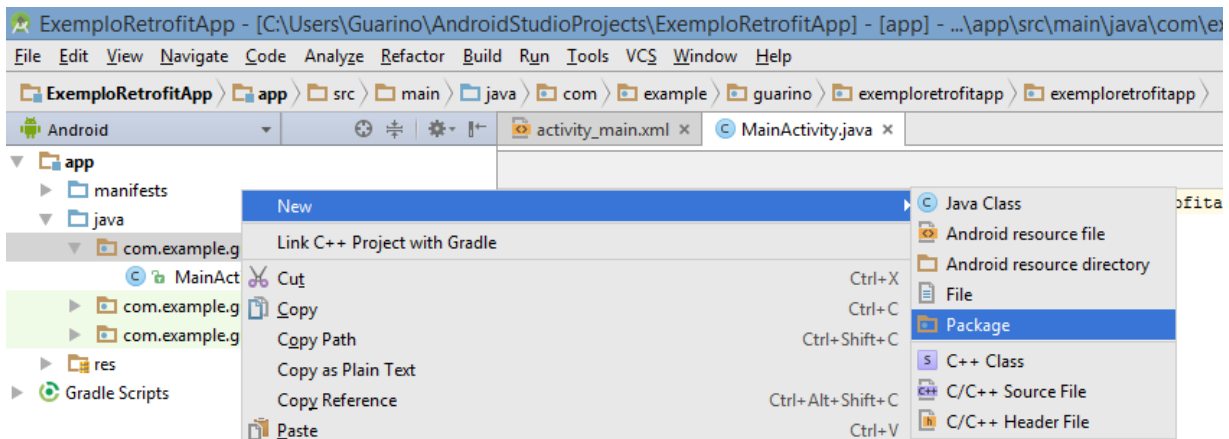


Selecione a opção mais recente dessa biblioteca.

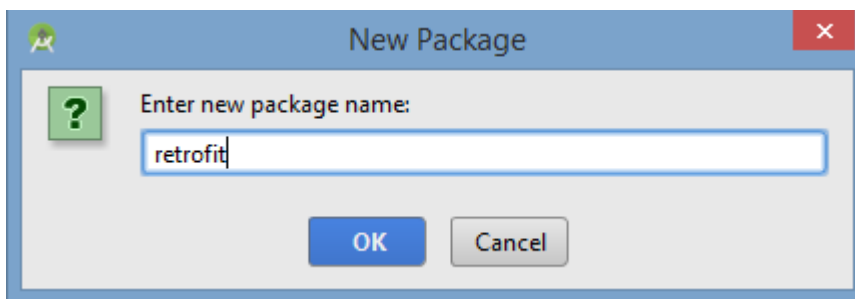


Clique em OK. Depois em OK novamente.

Crie um novo pacote no pacote principal.

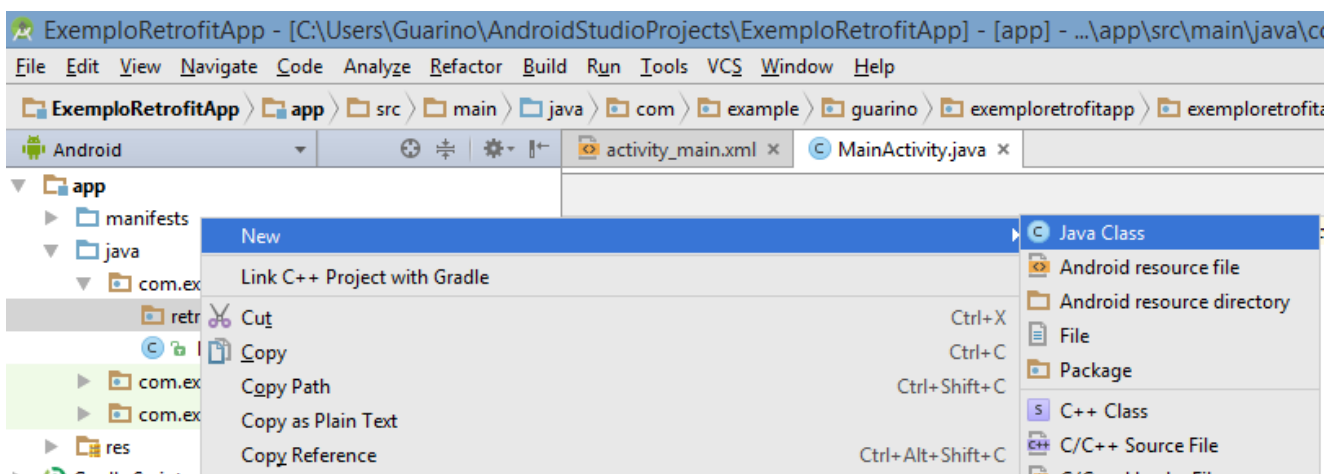


Nome do pacote: retrofit.

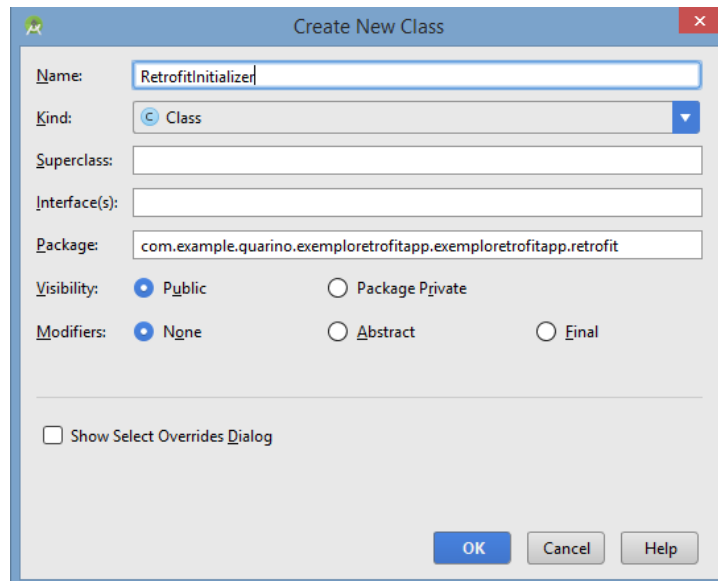


Clique em OK.

Nesse pacote, adicione uma Nova Classe.



Classe RetrofitInitializer



Clique em OK.

Deixe a classe como:

```
package com.example.guarino.exemploretrofitapp.exemploretrofitapp.retrofit;

import retrofit2.Retrofit;

/**
 * Created by Guarino on 18/09/2017.
 */

public class RetrofitInitializer {

    public RetrofitInitializer() {
        new Retrofit.Builder().baseUrl("https://viacep.com.br/ws/");
    }
}
```

No site do Retrofit existem alguns conversores.

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

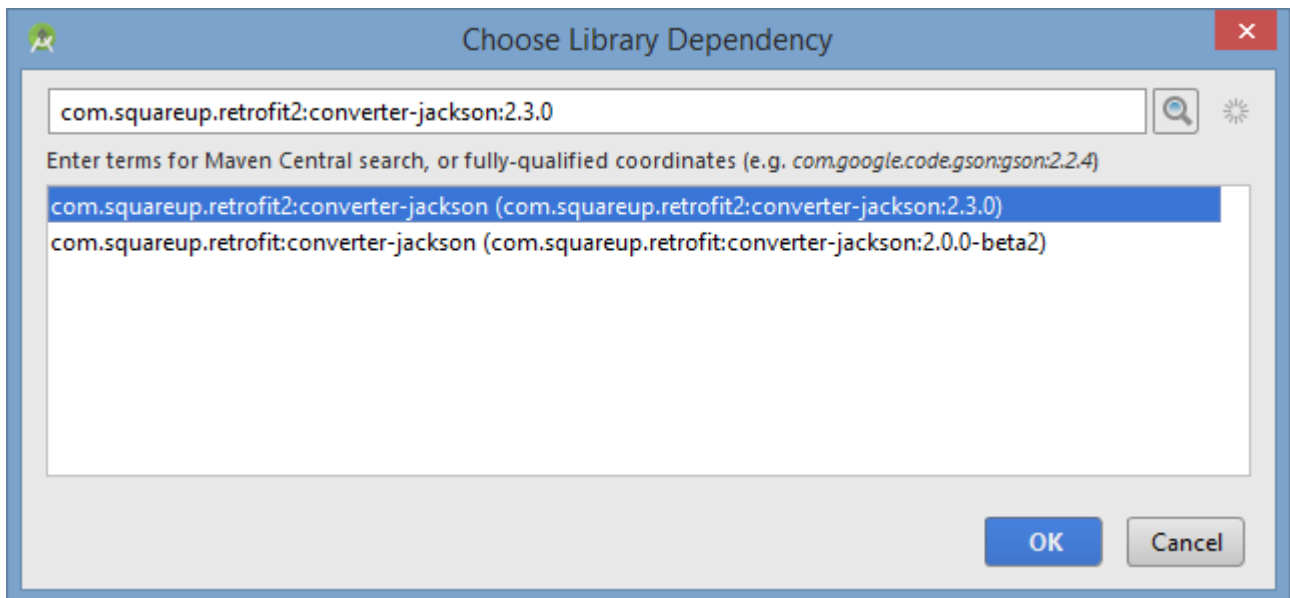
Adicione uma nova biblioteca.

File > Project Structure > Modulo app > aba Dependencies

Clique em “+” e Library dependency.

Procure por `com.squareup.retrofit2:converter-jackson` e digite Enter.

Adicione a biblioteca mais atual



Clique em OK. Depois em OK novamente.

Devemos alterar a classe RetrofitInitializer para informar qual conversor iremos usar.

```
package com.example.guarino.exemploretrofitapp.exemploretrofitapp.retrofit;

import retrofit2.Retrofit;
import retrofit2.converter.jackson.JacksonConverterFactory;

/**
 * Created by Guarino on 18/09/2017.
 */

public class RetrofitInitializer {

    private final Retrofit retrofit;

    public RetrofitInitializer() {
        retrofit = new Retrofit.Builder().baseUrl("https://viacep.com.br/ws/")
            .addConverterFactory(JacksonConverterFactory.create()).build();
    }
}
```

Alguns conversores:

- Gson - com.squareup.retrofit2:converter-gson
- Jackson - com.squareup.retrofit2:converter-jackson
- Moshi - com.squareup.retrofit2:converter-moshi
- Protobuf - com.squareup.retrofit2:converter-protobuf
- Wire - com.squareup.retrofit2:converter-wire
- Simple Framework - com.squareup.retrofit2:converter-simpleframework
- Scalars - com.squareup.retrofit2:converter-scalars

Acesse o ws para ver o exemplo de retorno.

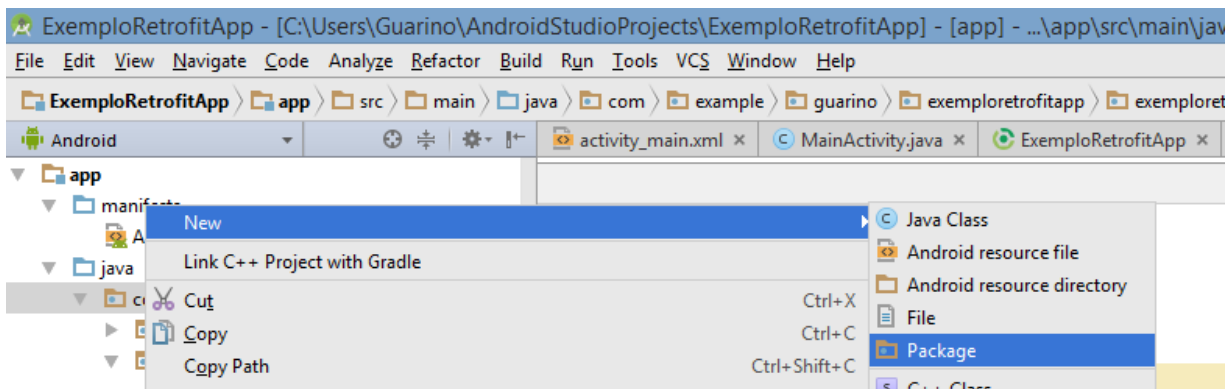
<https://viacep.com.br/ws/12570000/json/>

O retorno será

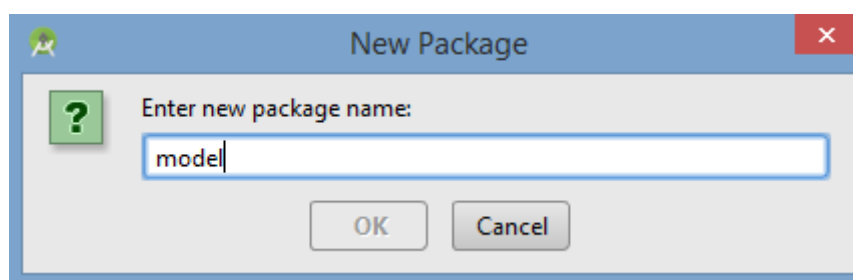
```
{
  "cep": "12570-000",
  "logradouro": "",
  "complemento": "",
  "bairro": "",
  "localidade": "Aparecida",
  "uf": "SP",
  "unidade": "",
  "ibge": "3502507",
  "gia": "1740"
}
```

É necessário criar uma classe que receberá essas informações.

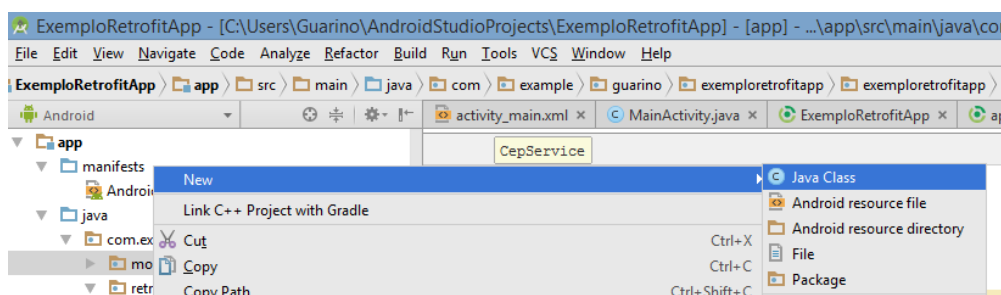
Crie um novo pacote no pacote principal.



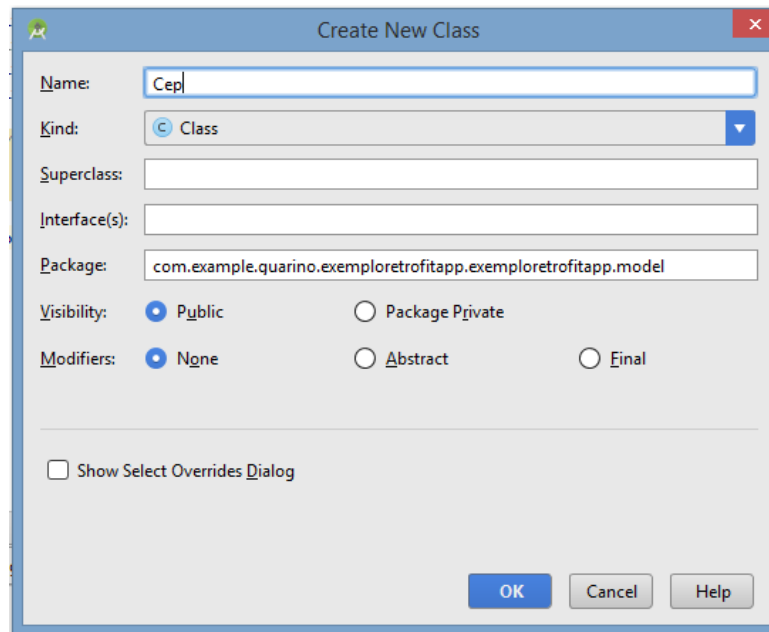
Crie o pacote model.



Nesse pacote, crie uma nova classe.



A classe deve se chamar Cep.



Crie as variáveis dessa classe.

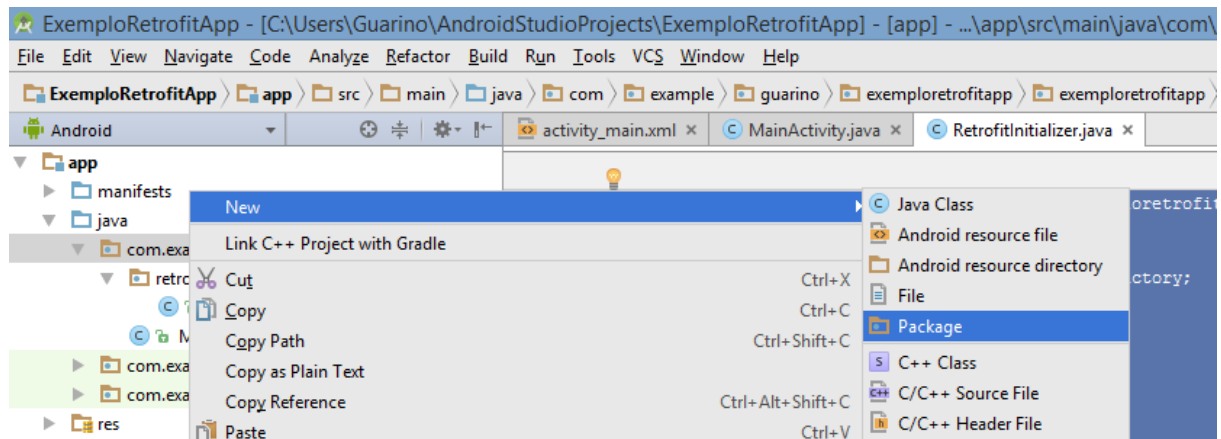
```
package com.example.guarino.exemploretrofitapp.exemploretrofitapp.model;
```

```
/**  
 * Created by Guarino on 19/09/2017.  
 */
```

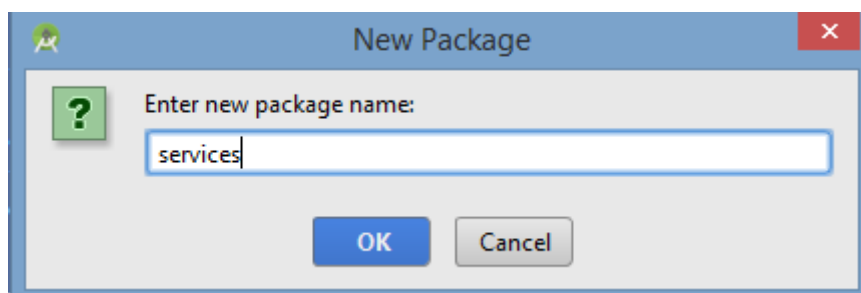
```
public class Cep {  
  
    private String cep;  
    private String logradouro;  
    private String complemento;  
    private String bairro;  
    private String localidade;  
    private String uf;  
    private String unidade;  
    private String ibge;  
    private String gia;  
  
}
```

Em seguida, use a Tecla de atalho ALT+INSERT e crie os métodos Getter e Setter de todas essas variáveis.

Crie um novo pacote no pacote principal

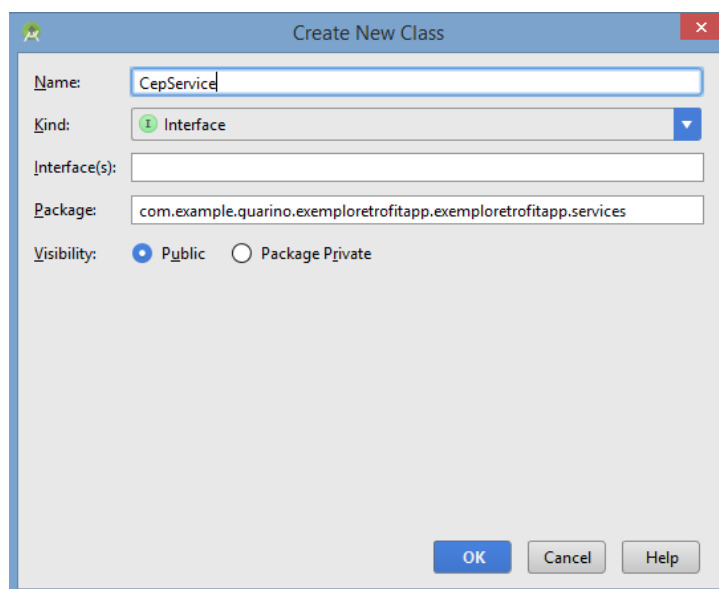


Nome do pacote: services



Clique em OK.

Adicione uma interface no pacote services.



Clique em OK.

A interface deverá ficar como segue.

```
package com.example.guarino.exemploretrofitapp.exemploretrofitapp.services;

import com.example.guarino.exemploretrofitapp.exemploretrofitapp.model.Cep;

import retrofit2.Call;
import retrofit2.http.Body;
import retrofit2.http.GET;
import retrofit2.http.Path;

/**
 * Created by Guarino on 18/09/2017.
 */

public interface CepService {

    @GET("{id}/json")
    Call<Cep> select(@Path("id") int id);

}
```

O retrofit por padrão usa verbos HTTP para indicar como a requisição deverá ser feita para um determinado endereço. Para obter informações por exemplo, usamos o verbo GET. Para enviar informações, usamos o POST.

Como a URL base do Retrofit está apontando para `"https://viacep.com.br/ws/"` Precisamos completar o caminho da URL. Assim, precisamos de um id/json. Como o id pode ser de qualquer endereço, devemos colocá-lo como variável na annotation, ficando {id}. A informação para o {id} virá do método select e para isso usamos @Path.

Caso queira usar mais de um parâmetro, use por exemplo:

```
@GET("{id}/{formato}")
Call<Cep> selectexemplo(@Path("id") int id, @Path("formato") String formato);
```

Por fim, a chamada Call do Retrofit deverá retornar um Cep, por isso Call<Cep>. Caso seja necessário retornar uma List, use Call<List<Cep>>.

Devemos alterar o RetrofitInitializer para fazer a chamada do método.

```
package com.example.guarino.exemploretrofitapp.exemploretrofitapp.retrofit;

import com.example.guarino.exemploretrofitapp.exemploretrofitapp.services.CepService;

import retrofit2.Retrofit;
import retrofit2.converter.jackson.JacksonConverterFactory;

/**
 * Created by Guarino on 18/09/2017.
 */

public class RetrofitInitializer {

    private final Retrofit retrofit;

    public RetrofitInitializer() {
        retrofit = new Retrofit.Builder().baseUrl("https://viacep.com.br/ws/")
            .addConverterFactory(JacksonConverterFactory.create()).build();
    }

    public CepService getCep()
    {
        return retrofit.create(CepService.class);
    }
}
```

Adicione a permissão para acesso a internet pelo app.
Isso deve ser feito no AndroidManifest.

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

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        ...
    >

    </application>

</manifest>
```

Na MainActivity, vamos criar a chamada do ws.

No onCreate, coloque a linha em destaque.

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    Call<Cep> call = new RetrofitInitializer().getCep().select(12570000);
```

Após essa linha, adicione a chamada

```
call.enqueue();
```

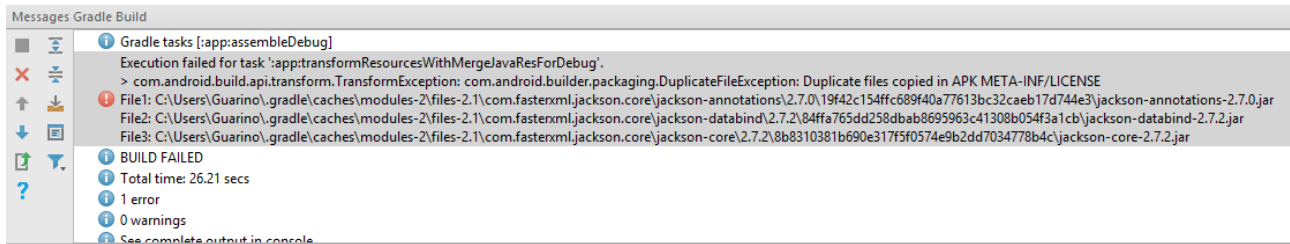
Dentro dos () do enqueue digite **new Callback** e complete o código.

O onCreate deverá ficar como segue.

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    Call<Cep> call = new RetrofitInitializer().getCep().select(12515690);  
    call.enqueue(new Callback<Cep>() {  
        @Override  
        public void onResponse(Call<Cep> call, Response<Cep> response) {  
            Log.i("Retrofit", response.body().getLogradouro());  
        }  
  
        @Override  
        public void onFailure(Call<Cep> call, Throwable t) {  
            Log.i("Retrofit", "falha");  
        }  
    });  
}
```

Execute a aplicação.

Deverá dar o seguinte erro. O erro é que algum jar está duplicado.



No Gradle do Module:app adicione o bloco

```
packagingOptions {  
    exclude "META-INF/LICENSE"  
}
```

O gradle deverá ficar assim:

```
apply plugin: 'com.android.application'  
  
android {  
  
    packagingOptions {  
        exclude "META-INF/LICENSE"  
    }  
  
    compileSdkVersion 25  
    buildToolsVersion "25.0.3"  
  
    ...  
}
```

Execute a aplicação.

Para verificar se deu certo, abra a janela de monitoramento.

No Android Studio, Menu View > Tool Windows > Android Monitor.

Alterando a tela activity_main.xml

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

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Informe o CEP" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/edtCEPInformado" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btnPesquisar"
        android:text="Pesquisar" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/txtLogradouro" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/txtBairro" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/txtLocalidade" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text=""
        android:id="@+id/txtUF" />

</LinearLayout>
```


Alterando a MainActivity

```
package com.example.guarino.exemploretrofitapp.exemploretrofitapp;

import ...

public class MainActivity extends AppCompatActivity {

    EditText edtCEPInformado;
    TextView txtLogradouro, txtBairro, txtLocalidade, txtUF;
    Button btnPesquisar;

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

        btnPesquisar = (Button) findViewById(R.id.btnPesquisar);

        btnPesquisar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                edtCEPInformado = (EditText) findViewById(R.id.edtCEPInformado);
                String cepinformado = edtCEPInformado.getText().toString();

                txtBairro = (TextView) findViewById(R.id.txtBairro);
                txtLocalidade = (TextView) findViewById(R.id.txtLocalidade);
                txtLogradouro = (TextView) findViewById(R.id.txtLogradouro);
                txtUF = (TextView) findViewById(R.id.txtUF);

                Call<Cep> call = new
                RetrofitInitializer().getCep().select(Integer.parseInt(cepinformado));
                call.enqueue(new Callback<Cep>() {
                    @Override
                    public void onResponse(Call<Cep> call, Response<Cep>
response) {

                        Cep cep = response.body();
                        txtUF.setText(cep.getUf());
                        txtBairro.setText(cep.getBairro());
                        txtLogradouro.setText(cep.getLogradouro());
                        txtLocalidade.setText(cep.getLocalidade());

                        Log.i("Retrofit", response.body().getLogradouro());
                    }

                    @Override
                    public void onFailure(Call<Cep> call, Throwable t) {
                        Log.i("Retrofit", "falha");
                    }
                });
            }
        });
    }
}
```

Execute a aplicação.