

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE BOTUCATU

Programação para Dispositivos Móveis

Guia para criação de um projeto Android no Android Studio para acesso web usando JSON

Osvaldo Cesar Pinheiro de Almeida

(cesar@fatecbt.edu.br)

Botucatu - SP

Outubro - 2016

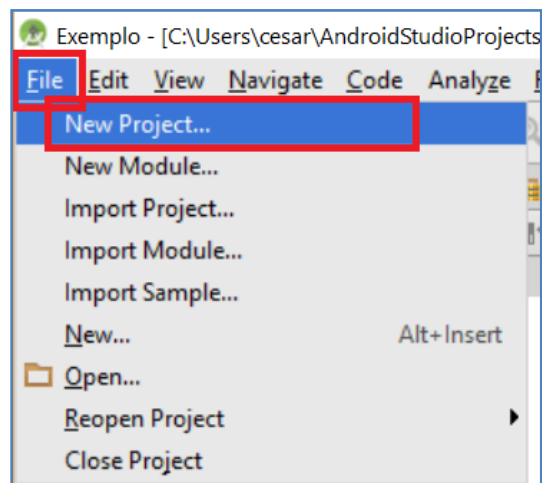
Inicialização

Requisitos iniciais:


- Android Studio completo - Versão recomendada do Android 1.1 ou maior;
- Observe as imagens com destaque em vermelho.

Criando Projeto

Usando o Android Studio iremos criar um novo projeto:



Create New Project

 **New Project**
Android Studio

Configure your new project

Application name:

JsonActivity

Company Domain:

cesar.example.com

Package name:

com.example.cesar.jsonactivity

[Edit](#)

Project location:

C:\Users\cesar\AndroidStudioProjects\JsonActivity

...


Previous

Next

Cancel

Finish

Create New Project

 **Target Android Devices**

Select the form factors your app will run on

Different platforms require separate SDKs

☒ Phone and Tablet

Minimum SDK

API 22: Android 5.1.1

Lower API levels target more devices, but have fewer features available. By targeting API 22 and later, your app will run on < 1% of the devices that are active on the Google Play Store. [Help me choose.](#)

☐ TV

Minimum SDK

API 21: Android 5.0 (Lollipop)

☐ Wear

Minimum SDK

API 21: Android 5.0 (Lollipop)

☐ Glass (Not Installed)

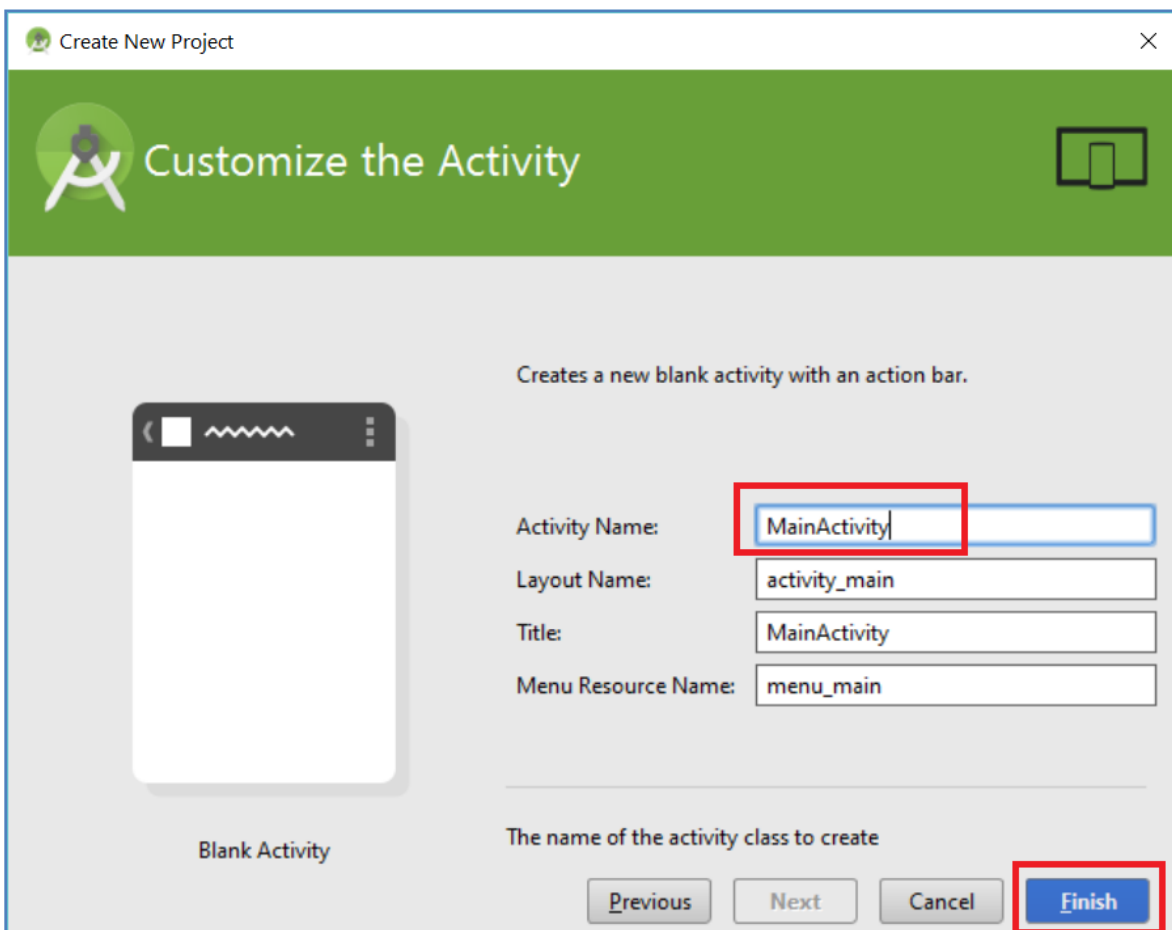
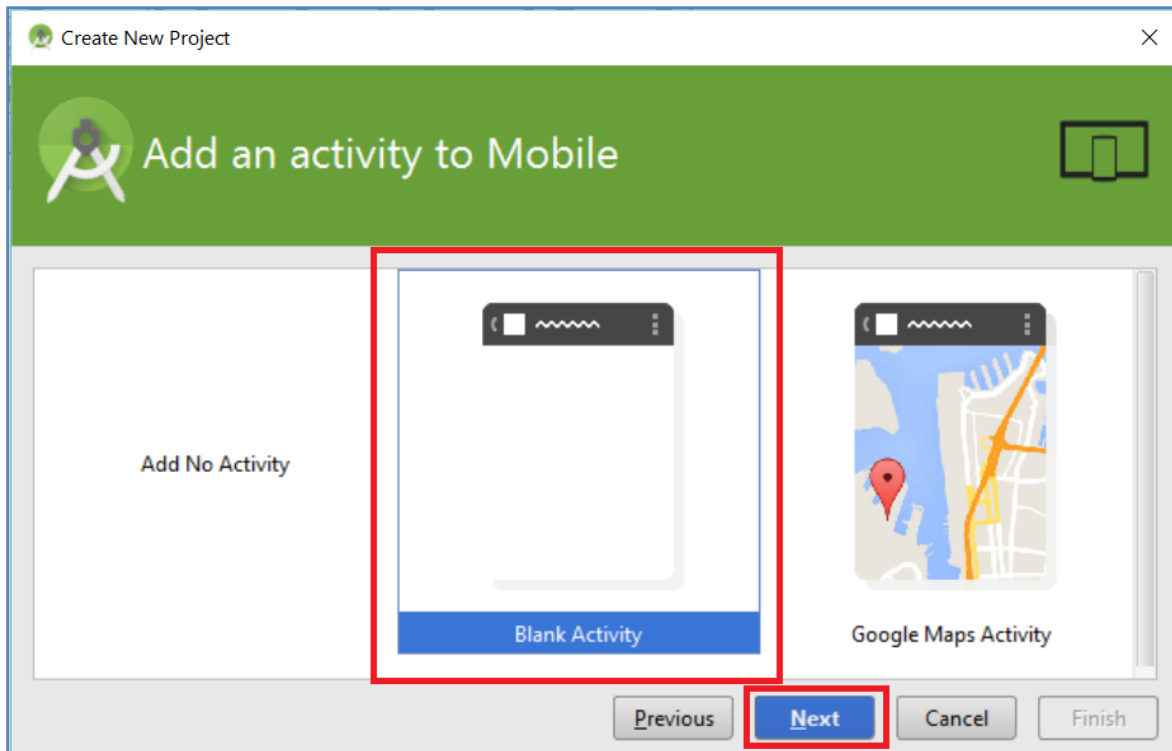
Minimum SDK

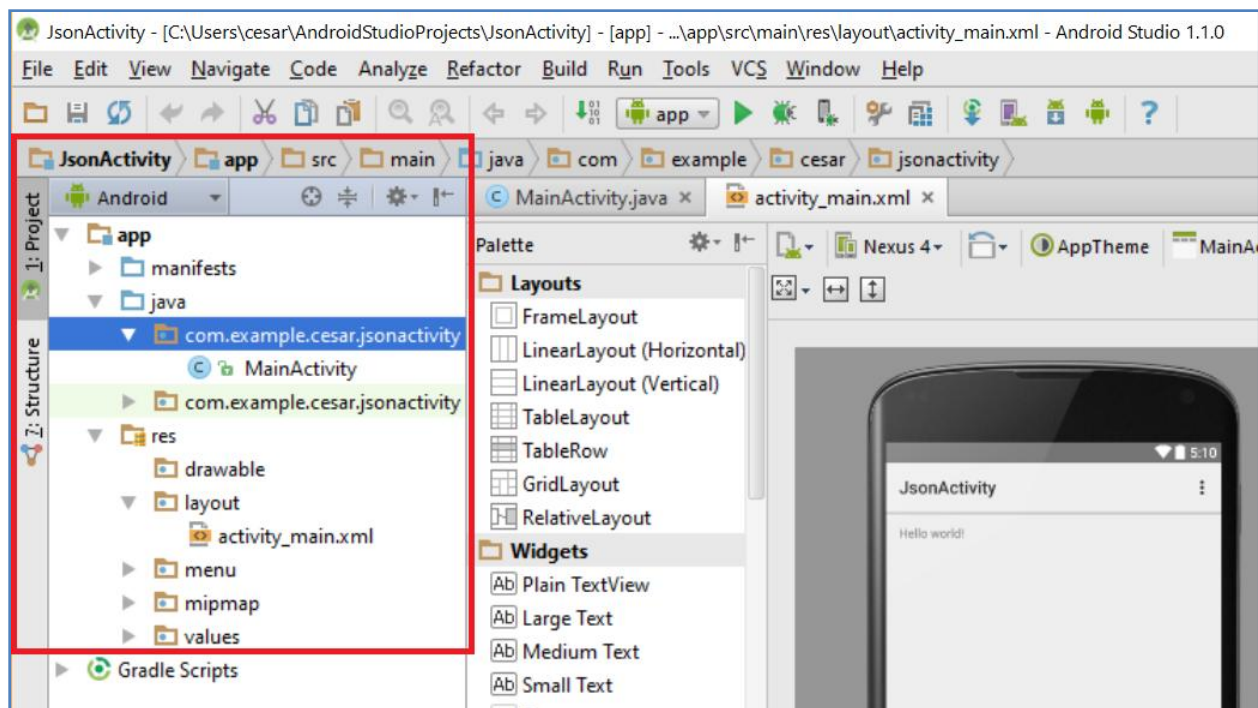
Previous

Next

Cancel

Finish





Nesse projeto utilizaremos a Activity para permitir que o usuário possa trabalhar com acesso de informações via internet (via web). A intenção é ilustrar os mecanismos utilizados por muitas aplicações Android que fazem uso do acesso web para recuperar ou transmitir informações à um servidor de dados. Isso pode ser realizado de várias maneiras, sendo que dois processos são bastante comuns: o uso de JSON e XML.

Para exemplificar a representação de arquivos do tipo JSON e XML serão apresentados abaixo dois exemplos, contendo a definição de dados de empregados de uma empresa, com dados de nome e sobrenome de três empregados:

JSON

```
{"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
]}
```

XML

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

Para a criação do projeto será usado um endereço web com informações que podem ser recuperadas em JSON. A partir do endereço <http://jsonplaceholder.typicode.com> é possível recuperar uma série de informações como Posts e Comments. Será usado o acesso aos Posts do site, onde cada Post contém os atributos: userId, id, title e body.

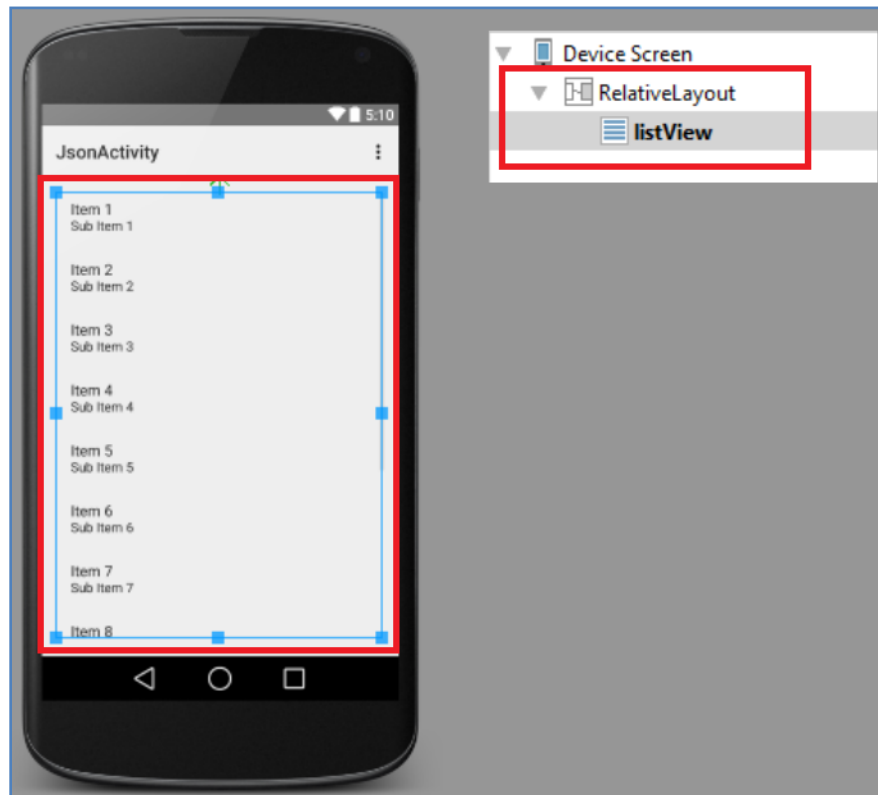
A aplicação de exemplo que iremos apresentar vai utilizar duas Activitys: uma para iniciar o acesso ao link <http://jsonplaceholder.typicode.com/posts> e listar todos os 100 itens de Posts disponibilizados; outra para visualizar os dados de cada Post com detalhe, após escolher qual Post deve ser detalhado.

Para construir de maneira organizada o projeto serão criadas as seguintes classes:

- Post - contendo a definição dos quatro atributos referentes ao Post, além da sobrescrita do método "toString()";
- JsonConsumer - classe que estende a classe AsyncTask, que representa uma tarefa assíncrona que deve ser executada. A classe JsonConsumer deve ser uma tarefa assíncrona pois ao solicitar o acesso de informações via web, o sistema Android irá recuperar essas informações sem paralisar a interação da APP com o usuário. Após recuperar as informações (caso seja possível), essas informações podem ser tratadas e exibidas.

Configurando o Layout MainActivity

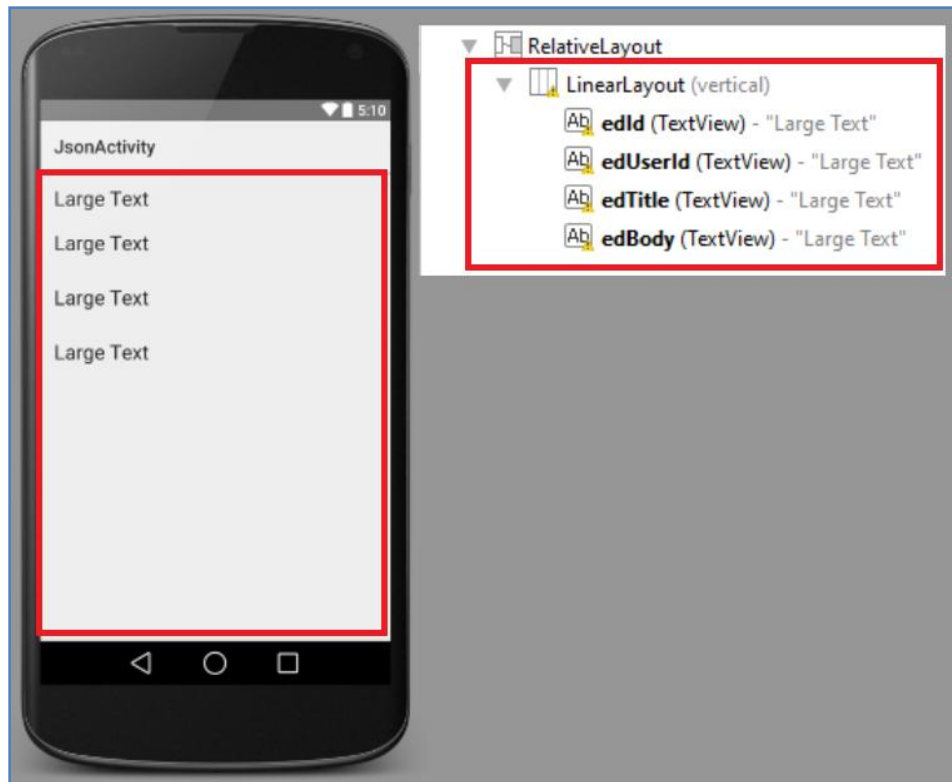
Para a criação do layout de exemplo da MainActivity será usado 1 lista de visualização. Essa lista servirá para a apresentação (listagem) de todos os 100 itens de Post do site. Assim, após remover o texto "hello world!", será adicionado um ListView. Ao clicar em cada item da lista deve ser requisitada a abertura de outra Activity que representará os dados do Post. O layout finalizado deve ficar como a figura abaixo.



Configurando o Layout PostActivity

Para a criação do layout de exemplo do PostActivity serão usados 5 componentes gráficos, sendo 4 TextView, para exibição de cada um dos atributos do Post, além de 1 LinearLayout (vertical) para organizar a exibição desses atributos.

Devemos então criar no projeto a PostActivity e editar o seu layout. Primeiro removemos o texto "hello world!". Depois será adicionado o LinearLayout (vertical) e em seguida os 4 componentes de TextView. O layout finalizado deve ficar como a figura abaixo.



Criando as classes

O projeto se baseia no uso de duas classes que devem criadas:

- Post.java

A classe Post.java deve ser criada de maneira que contenha os dados id, userId, title e body, com seus respectivos métodos "get" e "set", dois métodos construtores, sendo um sem parâmetros de entrada e outro com parâmetros referentes a todos os atributos da classe, além da sobrescrita do método "toString". A classe com a sua implementação deve ficar como a figura abaixo.

Post.java ×

```
package com.example.cesar.jsonactivity;

import java.io.Serializable;

public class Post implements Serializable{
    private int id;
    private int userId;
    private String title;
    private String body;

    public Post() {
    }

    public Post(int id, int userId, String title, String body) {
        this.id = id;
        this.userId = userId;
        this.title = title;
        this.body = body;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public String getBody() {
        return body;
    }

    public void setBody(String body) {
        this.body = body;
    }

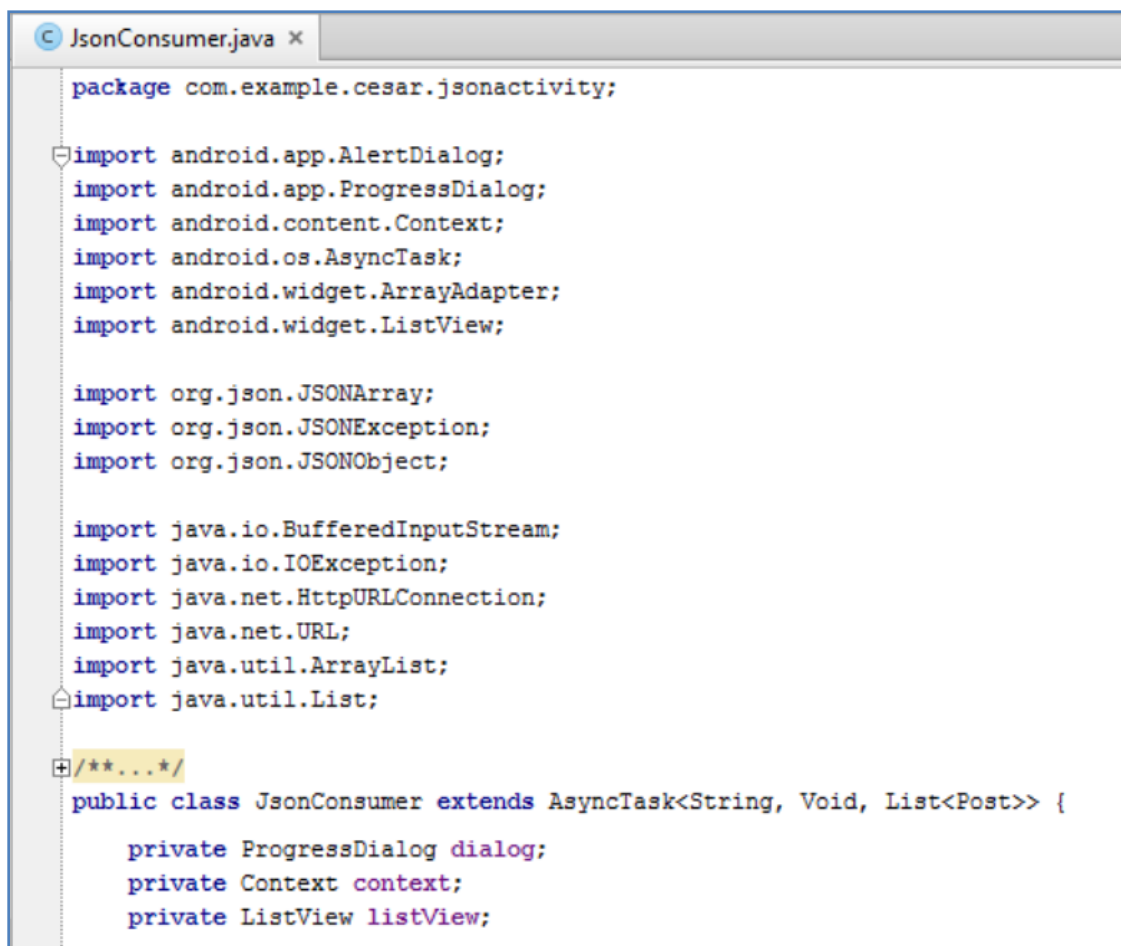
    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    @Override
    public String toString() {
        return id + " - " + title;
    }
}
```

- JsonConsumer.java

A classe JsonConsumer.java deve ser criada para tratar de maneira assíncrona o acesso ao servidor de informações. Nesta classe estará implementado o processo de acesso web e de tratamento das informações recebidos do servidor. Essa classe deve estender da classe AsyncTask, que é uma classe abstrata para criação de tarefas assíncronas (que deve ser usado no Android). Como uma classe abstrata exige que faça a implementação dos métodos abstratos, que no caso será apenas o método "doInBackground()", método responsável por determinar qual tarefa deve ser executada em "background". Além desse método que será responsável por acessar a web, serão implementados também um método construtor, contendo como parâmetros o contexto da aplicação e a lista (ListView) onde serão apresentados os dados e o método "getPosts()", responsável por tratar as informações no formato JSON. Por fim, serão sobrescritos dois métodos: "onPreExecute()", responsável pelo pré-processamento; e "onPostExecute()", responsável pelo pós-processo, onde será implementado o processo de exibição dos dados na lista de Posts (ListView).



```
JsonConsumer.java x
package com.example.cesar.jsonactivity;

import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.os.AsyncTask;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

/**...*/
public class JsonConsumer extends AsyncTask<String, Void, List<Post>> {
    private ProgressDialog dialog;
    private Context context;
    private ListView listView;
```

```

public JsonConsumer(Context context, ListView listView) {
    this.context = context;
    this.listView = listView;
}

@Override
protected void onPreExecute() {
    super.onPreExecute();

    dialog = ProgressDialog.show(context, "Aguarde",
        "Baixando JSON, Por Favor Aguarde...");
}

@Override
protected void onPostExecute(List<Post> posts) {
    super.onPostExecute(posts);
    dialog.dismiss();
    if (posts.size() > 0) {
        ArrayAdapter<Post> adapter = new ArrayAdapter<Post>(
            context, android.R.layout.simple_list_item_1, posts);
        listView.setAdapter(adapter);
    } else {
        AlertDialog.Builder builder = new AlertDialog.Builder(
            context).setTitle("Atenção")
            .setMessage("Não foi possível acessar essas informações...")
            .setPositiveButton("OK", null);
        builder.create().show();
    }
}

private List<Post> getPosts(String jsonString) {

    List<Post> posts = new ArrayList<>();

    try {
        JSONArray postsList = new JSONArray(jsonString);

        JSONObject postJson;

        for (int i = 0; i < postsList.length(); i++) {
            postJson = new JSONObject(postsList.getString(i));

            Post post = new Post();
            post.setId(postJson.getInt("id"));
            post.setUserId(postJson.getInt("userId"));
            post.setTitle(postJson.getString("title"));
            post.setBody(postJson.getString("body"));

            posts.add(post);
        }
    } catch (JSONException e) {
    }

    return posts;
}

```

```

@Override
protected List<Post> doInBackground(String... params) {
    String urlString = params[0];

    try {
        URL url = new URL(urlString);
        HttpURLConnection http = (HttpURLConnection) url.openConnection();

        BufferedInputStream input = new BufferedInputStream(http.getInputStream());
        StringBuilder builder = new StringBuilder();
        int size;
        byte[] bytes = new byte[1024];
        while ((size = input.read(bytes)) > 0) {
            builder.append(new String(bytes, 0, size));
        }
        String dados = builder.toString();

        http.disconnect();

        return getPosts(dados);
    } catch (IOException ex) {
    }

    return null;
}

```

Programando os processos na classe MainActivity

Na classe MainActivity será implementado um processo que será vinculado ao "clique" nos itens da lista de Posts. A programação da classe constará das seguintes implementações:

1) Criação de um método chamado "openView", que irá implementar o processo para chamada da PostActivity, passando o Post que foi clicado. Este método deve ter um parâmetro de entrada do tipo Post. Este método será vinculado ao clique dos itens da lista de Posts;

```

private void openView(Post post){
    Intent intent = new Intent(getApplicationContext(), PostActivity.class);
    intent.putExtra("post", post);
    startActivity(intent);
}

```

2) O método "onCreate" deve ser alterado para tratar a instanciação de um novo JsonConsumer, que irá executar a chamada do servidor de informações da web, e a vinculação entre a ação de clicar no item da lista e o método "openView".

```
public class MainActivity extends Activity {

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

        new JsonConsumer(MainActivity.this, (ListView) findViewById(R.id.listView))
            .execute("http://jsonplaceholder.typicode.com/posts");

        ((ListView) findViewById(R.id.listView)).setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                openView((Post) parent.getItemAtPosition(position));
            }
        });
    }
}
```

Programando os processos na classe PostActivity

A programação na classe PostActivity deve implementar o processo de exibição dos dados do Post nos componentes TextView. Para isso deve ser recuperado do Intent o Post passado na chamada da Activity. Esse processo deve ser implementado no método "onCreate".

```
public class PostActivity extends Activity {

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

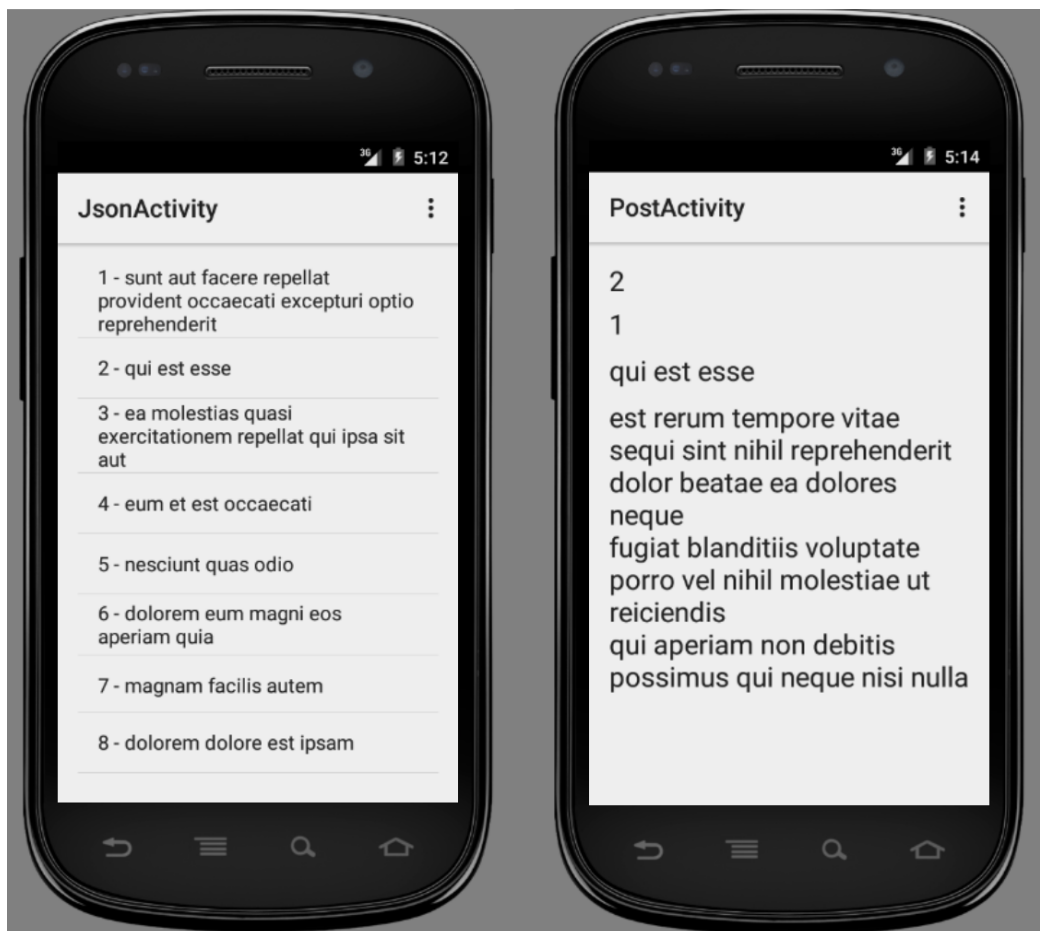
        Post post = (Post) getIntent().getSerializableExtra("post");
        ((TextView) findViewById(R.id.edId)).setText(String.valueOf(post.getId()));
        ((TextView) findViewById(R.id.edUserId)).setText(String.valueOf(post.getUserId()));
        ((TextView) findViewById(R.id.edTitle)).setText(post.getTitle());
        ((TextView) findViewById(R.id.edBody)).setText(post.getBody());
    }
}
```

Ajuste de permissão no Manifest

Para que a aplicação possa acessar os recursos de internet é necessário inserir a permissão de usuário "INTERNET" no arquivo AndroidManifest.xml, como ilustrado abaixo.



Se tudo estiver correto, basta executar o programa.



Bom trabalho!!!