FIAP

# DESENVOLVIMENTO MOBILE - ANDROID

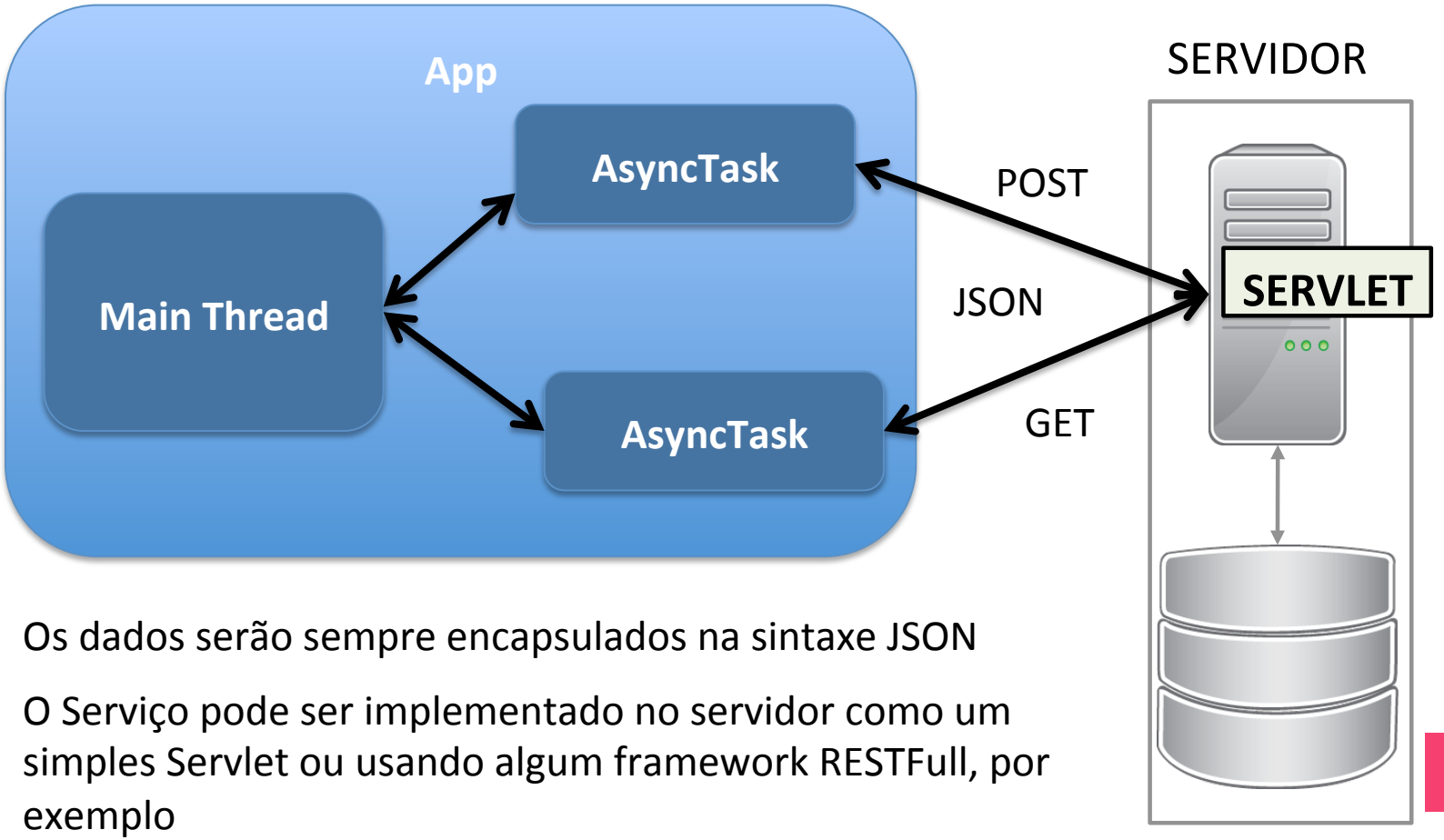
## HTTP REQUESTS

PROF. EDSON A. SENSATO
profedsonsensato@fiap.com.br

Versão 1– 2016

# ARQUITETURA

O App irá gerar dois tipos de requisição: POST para cadastrar dados e GET para obter dados do servidor




Os dados serão sempre encapsulados na sintaxe JSON

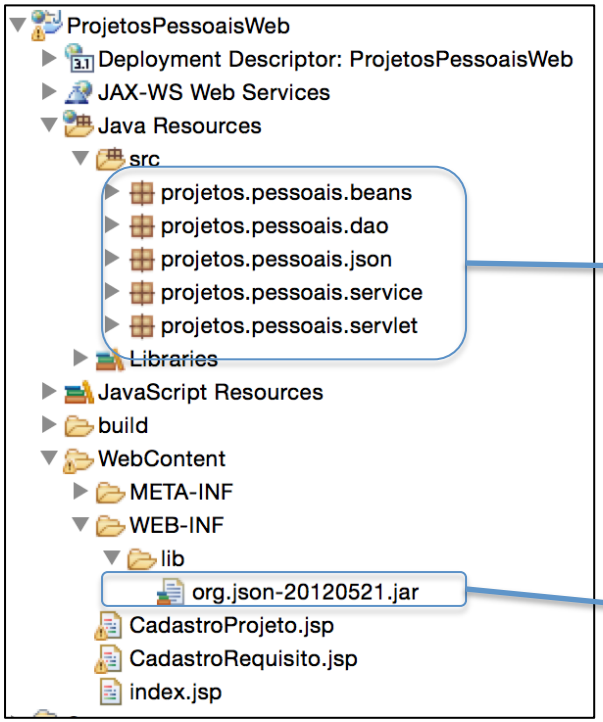
O Serviço pode ser implementado no servidor como um simples Servlet ou usando algum framework RESTFull, por exemplo

# JSON NO SERVIDOR

Existem várias bibliotecas prontas responsáveis por gerar a estrutura JSON a partir de objetos e também por interpretar uma String JSON (parsing)

Por exemplo: https://code.google.com/p/org-json-java/downloads/list

Você pode efetuar o donwload do arquivo jar, criar um simples Java Project ou Dynamic Web Project e adicionar o arquivo ao class path




Estrutura do projeto: beans, daos, servlets...

json → contém as classes para conversão bean x json e vice-versa

service → contém os Servlets que irão expor os serviços a serem consumidos pelo clientes

Biblioteca JSON

# SERVIDOR GET

Temos abaixo um simples exemplo de um Servlet atuando como serviço que devolve dados JSON em atendimento a uma requisição GET:

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {

    try {
        // ao invés de definir uma String hard coded como abaixo ela poderia
        // vir do resultado da execução de um método no DAO, por exemplo

        JSONObject aluno = new JSONObject();
        aluno.put("nome", "João da Silva");
        aluno.put("nota", 8);

        // retornar a String no formato JSON codificada em UTF-8 (acentos)

        resp.setCharacterEncoding("UTF-8");
        resp.getWriter().println(aluno.toString());
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

# SERVIDOR POST

Abaixo um atendimento à requisições POST (cadastro) em um Servlet

Neste caso, o JSON vem no corpo da requisição POST em um atributo json

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
    // lê o parâmetro json do corpo da requisição POST
    String json = request.getParameter("json"));
    try {
        JSONObject obj = new JSONObject(json);
    } catch (JSONException e) {
        e.printStackTrace();
    }
    // devolve um código de retorno ou erro, por exemplo
    String ret = "{\"resultado\":\"OK\"}";
    resp.setCharacterEncoding("UTF-8");
    resp.getWriter().println(ret);
}
```

# CLASSE HttpURLConnection

A classe HttpURLConnection permite realizar requisições (GET, POST, PUT, etc...) a um servidor, bastando informar a sua URL

```
// URL para uma requisição que retorna os dados do aluno 150

URL url = new URL("http://10.0.2.2:8080/AlunoServlet?aluno=150");

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

// realiza uma requisição GET, mas poderia ser POST também

conn.setRequestMethod("GET");
```

**ATENÇÃO!!!! O IP para localhost (127.0.0.1) no emulador deve ser 10.0.2.2!!!**

# LENDO A RESPOSTA

Após a requisição ser feita podemos ter acesso ao código de retorno (padrão HTTP) e o corpo da mensagem

```
int codigo = conn.getResponseCode());
// verifica o código de retorno, 200 = sucesso
if (codigo == 200) {

        BufferedInputStream in = new BufferedInputStream(conn.getInputStream());
        InputStreamReader r = new InputStreamReader(conn.getInputStream(), "UTF-8");
        StringWriter w = new StringWriter();
        int v = r.read();
        while (v != -1) {
                w.write(v);
                v = r.read();
        }
        // corpoRetorno contém o texto (JSON) retornado pelo servidor
        String corpoRetorno = w.toString();
}
```

# REQUISIÇÃO POST

```
// URL para uma requisição POST para cadastro de um aluno
URL url = new URL("http://10.0.2.2:8080/AlunoServlet");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("POST");
// define que os dados serão enviados no corpo da requisição POST
conn.setDoOutput(true);
DataOutputStream out = new DataOutputStream(conn.getOutputStream());
try {

        // coloca os dados do aluno (formato JSON) no corpo da requisição
        JSONObject aluno = new JSONObject();
        aluno.put("nome", "João da Silva");
        aluno.put("nota", 8);
        // cria um atributo chamado json no corpo da requisição POST
        out.writeBytes("json=" + aluno.toString());

} catch (JSONException e) {
        e.printStackTrace();
}
out.flush();
out.close();
```

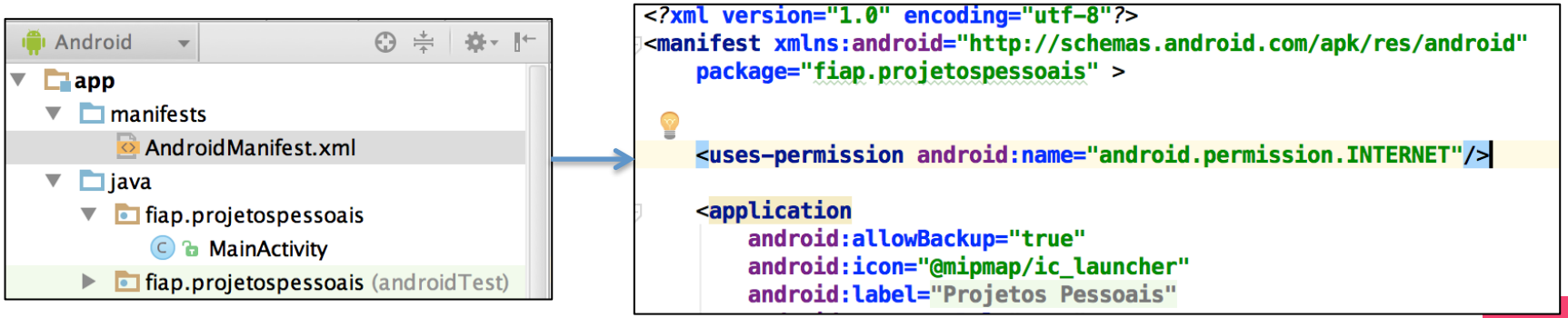
# AUTORIZAÇÃO INTERNET

Uma aplicação que realiza acessos à internet sem que o usuário tenha conhecimento pode ser perigosa

Desta forma, devemos declarar explicitamente que a aplicação fará o uso da INTERNET para se comunicar

Desta forma, ao instalar o app o usuário será advertido sobre este recurso

Temos que editar o AndroidManifest.xml e adicionar uma solicitação de permissão de acesso à internet:



# CHAMANDO O ASYNCTASK

Conforme já visto, cada requisição deve ser encapsulada em um AsyncTask

Pode-se passar como parâmetros ao AsyncTask a URL a ser requisitada, os dados JSON para serem enviados (numa requisição POST), etc…

```
String json = AlunoJSON.getJSON();
String URL = "http://10.0.2.2:8080/AlunoServlet";
PostTask post = new PostTask();

post.execute(URL, json);

public class PostTask extends AsyncTask<String, Void, String> {

@Override
protected JSONObject doInBackground(String... params) {

    String URL = params[0];
    String json = parms[1];

}
```

FIAP