**Aula 1 – Android 2 – Mobilidade e nuvem**

**Prof. Wagner Mendes Voltz - wagnerfusca@gmail.com**

**Objetivo:**

* Ter uma aplicação Java que utilize serviços
* Ter a aplicação Java disponibilizada na nuvem
* Usar serviço de versionamento na nuvem
* Usar serviço de container web na nuvem
* Realizar deploy na nuvem
* Ter o serviço usando POST/json
* Ter banco de dados na nuvem
* Android consumir serviço de banco de dados

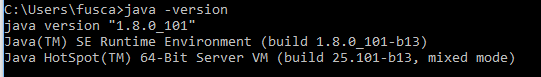
**Material Aula**

* Disponibilizar Java 8
  + Nas máquinas da FCV já tem instalado
  + Quem tem Java 7 também vai funcionar
* Disponibilizar Apache Maven 3.3.9
  + Caso você já tenha um Maven configurado, pode continuar a usar a sua versão.
* Disponibilizar Eclipse Neon (IDE Java)
  + Caso você tenha outra versão do Eclipse, pode usar a sua versão.
  + Caso você tenha Netbeans, pode usar esta IDE.
  + Caso você tenha Intellij, pode usar esta IDE.
* Ter uma conta no Heroku
* Instalar o Heroku Toolbelt
* Utilizar Postman

**Instalação JAVA**

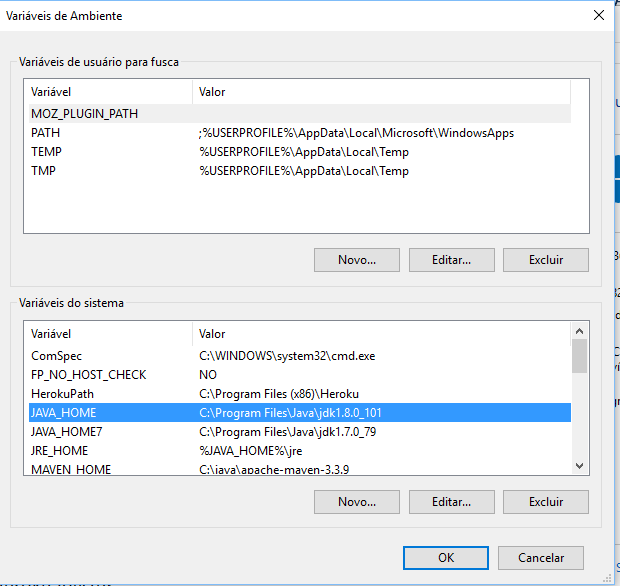
1) abra um prompt de commando (cmd.exe)

2) verifique se o java está instalado (digite **java -version**)



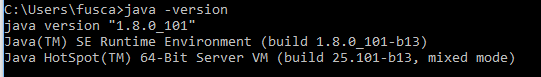
3) Caso não esteja configurado,

* Clique em instale jdk-8u101-windows-x64.exe
* Após instalação, pressione ***"bandeira windows" + PAUSE*** ***--> configurações avançadas do sistema --> aba avançado --> botão VARIAVEIS DE AMBIENTE*** e irá abrir a tela abaixo



* No quadrado VARIAVEIS DO SISTEMA (segundo quadrado), clique no botão NOVO. Adicione a variavel JAVA\_HOME (***em maiúsculo***) e adicione onde está a instalação da sua JDK (**não é a JRE!!!!!**)
* Após criar esta variável, clique em **PATH** e no início acrescente **%JAVA\_HOME%\bin\;** *(cuidado com o ponto e virgula, ele separa os comandos)*

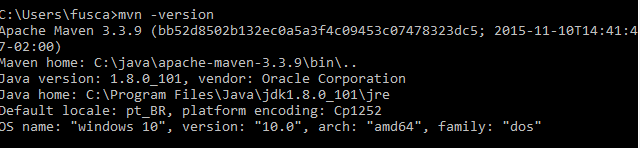
1. Pressione OK e abra um novo prompt de comando (cmd.exe) e digite **java –version**. Irá aparecer



**Instalação Apache Maven**

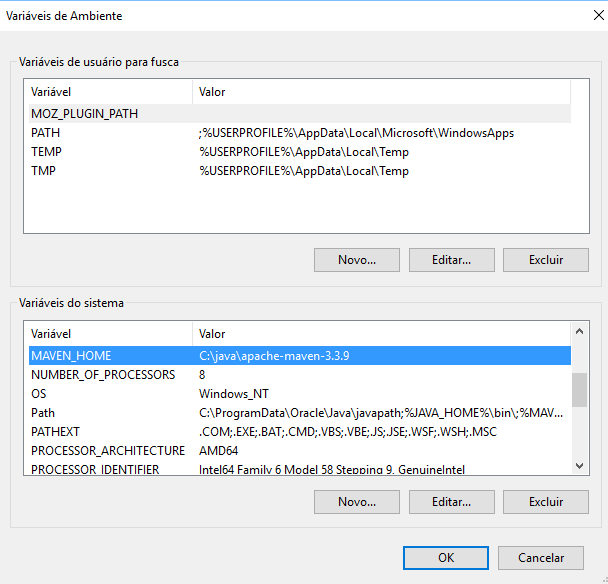
1) abra um prompt de commando (cmd.exe)

2) verifique se o Apache Maven está instalado (digite **mvn -version**)



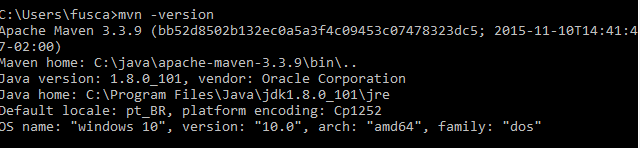
3) Caso não esteja configurado,

* Descompacte o arquivo ***apache-maven-3.3.9-bin.zip***
* Em seguida, pressione ***"bandeira windows" + PAUSE*** ***--> configurações avançadas do sistema --> aba avançado --> botão VARIAVEIS DE AMBIENTE*** e irá abrir a tela abaixo



* No quadrado VARIAVEIS DO SISTEMA (segundo quadrado), clique no botão NOVO. Adicione a variavel MAVEN\_HOME (***em maiúsculo***) e adicione onde está a Apache Maven descompactado.
* Após criar esta variável, clique em **PATH** e logo após a JAVA\_HOME, adicione a variável **%MAVEN\_HOME%\bin\;** *(cuidado com o ponto e virgula, ele separa os comandos)*

1. Pressione OK e abra um novo prompt de comando (cmd.exe) e digite **maven -version**. Irá aparecer

**

**\*\* COMANDOS IMPORTANTES HEROKU\*\***

* heroku login (faz login no heroku)
* heroku create (cria um vinculo heroku/git com aquela pasta)
* heroku open (abrir o a aplicação)
* heroku logs --tail ou heroku logs(exibe logs)
* heroku ps (exibe quantos containers estão sendo executados)--> dyno
* heroku ps:scale web=1 (sobe um container para executar as aplicações)
* heroku git:clone –a nomeDoRepositorio(obter um repositório já existente)
* heroku git:remote -a newname (renomeia o repositório)
* git init (criar um repositório)
* git add . (adiciona todas as alterações)
* git commit -m "Demo" ("envia" alteração para o repositório)
* git push heroku master (envia efetivamente a alteração para o repositório master)
* git pull heroku master (busca a alteração para o repositório master)
* git status(exibe o que tem de diferença do repositório com o local)
* git checkout \* (descarta todas as alterações locais)

*dica: http://rogerdudler.github.io/git-guide/index.pt\_BR.html*

-------------------------------

**\*\* PARA RODAR LOCALMENTE\*\***

WINDOWS 🡪 target\bin\webapp.bat

LINUX 🡪 sh target/bin/webapp

**\*\*RESTEASY USANDO POST\*\***

1. Com form

-------------------------------

<form method="POST" action="login">

Email Address: <input type="text" name="email">

<br>

Password: <input type="text" name="password">

<input type="submit">

</form>

-------------------------------

@Path("/")

public class LoginService

{

@Path("login")

@POST

public String login(@**FormParam**("email") String e, @**FormParam**("password") String p) {

return "Logged with " + e + " " + p;

}

1. Sem form (e usando json)

\*necessário usar a biblioteca **Jackson JSON processor para converter objetos json e devolver json para o cliente**

* **Antes de começar a desenvolver, verifique se está configurado os seguintes itens:**
  + **Java (ir no prompt de comando e digite *java –version*)**
  + **Maven (ir no prompt de comando e digite *mvn –version*)**
  + **Heroku ToolBelt (ir no prompt de comando e digite *heroku version*)**
* **Realize o login no heroku (ir no prompt de comando e digite *heroku login*)**
* **Acesse a pasta do projeto (cd c:\seuprojeto, por exemplo)**
  + **Caso você não tenha o projeto na maquina, siga os passos abaixo:**
    - **Crie uma pasta (mkdir nomeDaPasta)**
    - **Acesse a pasta (cd nomeDaPasta)**
    - **Baixe o código fonte versionado na nuvem (*heroku git:clone -a seuprojetonanuvem*)**
      * **Para saber o nome do seu projeto na nuvem digite heroku apps**
* **Atualize a versão do código fonte do projeto na sua máquina (*git pull heroku master)***
* **Abra a IDE de desenvolvimento (por exemplo *Eclipse*) e importe o projeto (FILE 🡪 IMPORT 🡪 MAVEN PROJECT). A IMPORTAÇÃO É SOMENTE PARA QUEM FEZ NOVAMENTE O CLONE DO PROJETO**

**ADICIONAR UMA NOVA DEPENDÊNCIA NO POM.XML**

**<dependency>**

**<groupId>org.jboss.resteasy</groupId>**

**<artifactId>resteasy-jackson-provider</artifactId>**

**<version>3.0.19.Final</version>**

**</dependency>**

**CRIAR UMA NOVA CLASSE – Produto**

**Obs: antes, crie um pacote chamado br.edu.fcv.model**

**package br.edu.fcv.model;**

**public class Produto {**

**private String nome;**

**private int quantidade;**

**public String getNome() {**

**return nome;**

**}**

**public void setNome(String nome) {**

**this.nome = nome;**

**}**

**public int getQuantidade() {**

**return quantidade;**

**}**

**public void setQuantidade(int quantidade) {**

**this.quantidade = quantidade;**

**}**

**}**

**CRIE UM NOVO SERVIÇO - JSONService.java**

**Obs: use o pacote rest já criado**

**package br.edu.fcv.rest;**

**import javax.ws.rs.Consumes;**

**import javax.ws.rs.GET;**

**import javax.ws.rs.POST;**

**import javax.ws.rs.Path;**

**import javax.ws.rs.Produces;**

**import javax.ws.rs.core.Response;**

**@Path("/json/produto")**

**public class JSONService {**

**@GET**

**@Path("/get")**

**@Produces("application/json")**

**public Produto getProdutoJSON() {**

**Produto produto = new Produto();**

**produto.setNome("Moto X - Android");**

**produto.setQuantidade(10);**

**return produto;**

**}**

**@POST**

**@Path("/post")**

**@Consumes("application/json")**

**public Response criarProdutoJSON(Produto produto) {**

**String result = "Product created : " + produto;**

**return Response.status(201).entity(result).build();**

**}**

**}**

**ALTERE O ARQUIVO web.xml MAPEANDO O NOVO SERVIÇO**

**Obs: o arquivo web.xml está na pasta em src 🡪 main 🡪 webapp 🡪 WEB-INF, ou digite CTRL+ SHIFT + R**

*<context-param>*

*<param-name>resteasy.resources</param-name>*

*<param-value>br.edu.fcv.rest.MessageRestService,*

*br.edu.fcv.rest.JSONService</param-value>*

*</context-param>*

**COMPILE E SUBA O PROJETO LOCALMENTE**

* mvn clean install
* target\bin\webapp.bat

**PARA TESTAR 🡪 USE O POSTMAN**

* Get : <http://localhost:8080/rest/json/produto/get>
* Post: **Postman 🡪 post 🡪 body 🡪 raw 🡪 {"nome":"Notebook","quantidade":"10"}**

http://localhost:8080/rest/json/produto/post

**PARA RECUPERAR UMA LISTA DE DADOS**

**Obs: é necessário criar uma raiz para o JSON para que o Android identifique o JSON.**

*\*\* criar um arquivo chamado ApiResponse.java*

**package** br.edu.fcv.model;

**import** java.util.List;

**public** **class** ApiResponse {

**public** ApiResponse() {

}

**private** List<Produto> produtos;

**public** List<Produto> getProdutos() {

**return** produtos;

}

**public** **void** setProdutos(List<Produto> produtos) {

**this**.produtos = produtos;

}

}

*\*\* alterar o serviço* **JSONService***, adicionando um novo método*

@GET

@Path("/listaprodutos")

@Produces("application/json")

**public** ApiResponse getListaProdutos(){

List<Produto> novaLista = **new** ArrayList<Produto>();

**Produto produto = new Produto();**

**produto.setNome("Moto X - Android");**

**produto.setQuantidade(10);**

novaLista.add(produto);

**Produto produto = new Produto();**

**produto.setNome("Galaxy 7");**

**produto.setQuantidade(15);**

novaLista.add(produto);

**Produto produto = new Produto();**

**produto.setNome("Iphone");**

**produto.setQuantidade(5);**

novaLista.add(produto);

ApiResponse apiResponse = **new** ApiResponse();

apiResponse.setProdutos(novaLista);

**return** apiResponse;

}

**COMPILE E SUBA O PROJETO LOCALMENTE**

* mvn clean install
* target\bin\webapp.bat

**PARA TESTAR 🡪 USE O POSTMAN**

* Get : [http://localhost:8080/rest/json/produto/listaprodutos](http://localhost:8080/rest/json/produto/listaprodutos%20)

**APÓS TESTES LOCAIS, ENVIE PARA A NUVEM**

* git add
* git commit –m “mensagem”
* git push heroku master

APÓS A CONDUZIR QUEM PODERP USAR O WIKI

**\*\*USANDO BANCO DE DADOS\*\***

* Instalar o PostGreSQL no Heroku
  + Faça o login no heroku.com
  + Clique em Resources
  + Clique em Add-ons e digite POSTGRE
  + Adicione o Heroku Postgres
* Instalar PG ADMIN 3 (opcional)
* Acessar o heroku e pegar os dados de host, user e password
  + Na aba Resources clique em Heroku Postgres :: Database
  + Selecione a base de dados gerada para seu projeto
* mudar o pom do projeto java
* Criar a entidade Lembrete.java (com id e campo descrição)
* Criar o DAO
* Criar o LembreteDAO
* mapear o persistence.xml
* testcase
* criar servisse
* enviar para o heroku

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.2.3.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>4.2.3.Final</version>

</dependency>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>9.4.1211</version>

</dependency>

**Crie uma classe chamada Lembrete.java**

@Entity

**public** **class** Lembrete {

@Id

@GeneratedValue(strategy = GenerationType.*AUTO*)

**private** Long id;

@Column

**private** String lembrete;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getLembrete() {

**return** lembrete;

}

**public** **void** setLembrete(String lembrete) {

**this**.lembrete = lembrete;

}

}

**Crie uma classe chamada DAO.java**

Obs: crie um pacote chamado br.edu.fcv.dao

**public** **class** DAO {

**private** EntityManagerFactory emf;

**public** EntityManager getEntityManager() {

**return** emf.createEntityManager();

}

**public** DAO() {

emf = Persistence.*createEntityManagerFactory*("config-entidades");

}

}

**Crie uma classe chamada LembreteDao.java**

**public** **class** LembreteDao **extends** DAO{

**public** **void** save(Lembrete lembrete){

EntityManager manager = getEntityManager();

manager.getTransaction().begin();

manager.persist(lembrete);

manager.getTransaction().commit();

}

**public** List<Lembrete> list(){

EntityManager manager = getEntityManager();

**return** manager.createQuery("from Lembrete").getResultList();

}

}

**Crie um arquivo chamado persistence.xml em src 🡪 main 🡪 resources 🡪 META-INF**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<persistence version="2.0"

xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

http://java.sun.com/xml/ns/persistence

http://java.sun.com/xml/ns/persistence/persistence\_2\_0.xsd" >

<persistence-unit name=*"*config-entidades*"*>

<provider>org.hibernate.ejb.HibernatePersistence</provider>

<class>br.edu.fcv.model.Lembrete</class>

<properties>

<property name=*"hibernate.bytecode.use\_reflection\_optimizer"* value=*"false"*/>

<property name=*"hibernate.connection.driver\_class"* value=*"org.postgresql.Driver"* />

<property name=*"hibernate.connection.username"* value=*"\*\*\*\*\*\*\*\*\*\*"*/>

<property name=*"hibernate.connection.password"* value=*"\*\*\*\*\*\*\*\*\*\*"*/>

<property name=*"hibernate.connection.url"* value=*"jdbc:postgresql://ec2-54-243-132-114.compute-1.amazonaws.com:5432/dfe0udqguam6oa?username=\*\*\*\*\*\*\*\*\*\*&amp;password=\*\*\*\*\*\*\*\*\*\*&amp;ssl=true&amp;sslfactory=org.postgresql.ssl.NonValidatingFactory"*/>

<property name=*"hibernate.archive.autodetection"* value=*"class"*/>

<property name=*"hibernate.hbm2ddl.auto"* value=*"update"* />

<property name=*"hibernate.show\_sql"* value=*"false"* />

<property name=*"hibernate.dialect"* value=*"org.hibernate.dialect.PostgreSQLDialect"* />

</properties>

</persistence-unit>

</persistence>

**Dentro do pacote DAO, crie a classe TestDao**

**public** **class** TestDao {

**public** **static** **void** main(String[] args) {

EntityManagerFactory factory = Persistence

.*createEntityManagerFactory*("config-entidades");

EntityManager em = factory.createEntityManager();

Lembrete lembrete = **new** Lembrete();

lembrete.setLembrete("aula sabado de manha");

em.getTransaction().begin();

em.persist(lembrete);

em.getTransaction().commit();

}

}

**Crie um serviço chamado LembreteRest.java**

Obs: não esqueça de mapea-lo no arquivo web.xml

@Path("/lembrete/")

**public** **class** LembreteRest {

LembreteDao dao = **new** LembreteDao();

@GET

@Path("salvar/{param}")

**public** Response salvarLembrete(@PathParam("param") String descricao) {

Lembrete lembrete = **new** Lembrete();

lembrete.setLembrete(descricao);

dao.save(lembrete);

**return** Response.*status*(200).entity("ok").build();

}

@GET

@Path("listar")

**public** Response listarLembretes() {

List<Lembrete> list = dao.list();

String result = "Quantidade de lembretes : " + list.size();

**return** Response.*status*(200).entity(result).build();

}

}

**COMPILE E SUBA O PROJETO LOCALMENTE**

* mvn clean install
* target\bin\webapp.bat

**APÓS TESTES LOCAIS, ENVIE PARA A NUVEM**

* git add
* git commit –m “mensagem”
* git push heroku master

Link para teste

* seusite.herokuapp.com/rest/lembrete/salvar/teste
* seusite.herokuapp.com/rest/lembrete/listar

**\*\*ANDROID\*\***

**CRIANDO O PROJETO**

1. Abra o Android Studio
   1. Para os que estão usando notebook, use o que está todo configurado
   2. Para os que estão usando computador do laboratório, use o que está em d:\android\android-studio\bin\studio64.exe
2. Crie um novo projeto Android (FILE 🡪 NEW 🡪 NEW PROJECT)
3. No campo Application Name digite **LembreteAndroid**
4. No COMPANY NAME, coloque **fcv.edu.br**
5. Na configuração da SDK
   1. Para os que estão usando notebook, use o que está configurado na sua maquina
   2. Para os que estão usando computador do laboratório, use a SDK 23
6. Selecione EMPTY ACTIVITY
7. Dê o nome de **LembreteActivity**
8. Finalize a criação do projeto

**ADICIONANDO BIBLIOTECA – ARQUIVO BUILD.GRADLE (module: app)**

useLibrary 'org.apache.http.legacy'

**CONFIGURANDO AS PERMISSÕES DO PROJETO**

**Obs: este arquivo esta em manifests🡪 AndroidManifest.xml**

Abra o AndroidManifest.xml e adicione a permissão de internet

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

**ALTERANDO A TELA activity\_lembrete.xml**

**Obs: este arquivo esta em res🡪 layout**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<EditText

android:id="@+id/lembrete"

android:layout\_width="fill\_parent"

android:layout\_height="0dp"

android:layout\_weight="0.35"

android:hint="Digite um lembrete" />

<TextView

android:id="@+id/resultado"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content" />

<Button

android:id="@+id/btnAdicionar"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Adicionar" />

<Button

android:id="@+id/btnListar"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Listar" />

<Button

android:id="@+id/btnListarProdutos"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Listar Produtos" />

<Button

android:id="@+id/btnCriarJson"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Criar Produtos" />

</LinearLayout>

**ALTERANDO A ACTIVITY LembreteActivity.java**

**Obs: este arquivo esta em java🡪 br.edu.fcv**

package br.edu.fcv.lembrete;

import android.app.Activity;

import android.app.ProgressDialog;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import org.apache.http.HttpEntity;

import org.apache.http.HttpHost;

import org.apache.http.HttpResponse;

import org.apache.http.client.HttpClient;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.entity.StringEntity;

import org.apache.http.impl.client.DefaultHttpClient;

import org.apache.http.util.EntityUtils;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.IOException;

import java.net.URLEncoder;

public class LembreteActivity extends Activity implements View.OnClickListener, Runnable {

private static final String CATEGORIA = "CONEXAO";

private ProgressDialog dialog;

private int idBotaoSelecionado;

private static final String HOST = "projetofusca.herokuapp.com";

private static final String PATH = "/rest/lembrete/";

private static final String LISTAR = "listar/";

private static final String INSERIR = "salvar/";

private static final String PATH\_PRODUTO = "/rest/json/produto/";

private static final String INSERIR\_PRODUTO = "post/";

private static final String LISTAR\_PRODUTO = "listaprodutos/";

@Override

public void onCreate(Bundle bundle) {

super.onCreate(bundle);

setContentView(R.layout.activity\_lembrete);

mapearBotoesDaTela();

}

private void mapearBotoesDaTela() {

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

btnListar.setOnClickListener(this);

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

btnAdicionar.setOnClickListener(this);

Button btnListarProduto = (Button) findViewById(R.id.btnListarProdutos);

btnListarProduto.setOnClickListener(this);

Button btnAdicionarProduto = (Button) findViewById(R.id.btnCriarJson);

btnAdicionarProduto.setOnClickListener(this);

}

public void onClick(View view) {

dialog = ProgressDialog

.show(this,

"Loading...",

"Acessando base de dados com web service, por favor aguarde...",

false, true);

idBotaoSelecionado = view.getId();

new Thread(this).start();

}

@Override

public void run() {

EditText textLembrete = (EditText) findViewById(R.id.lembrete);

final TextView textResultado = (TextView) findViewById(R.id.resultado);

String novoPath = "";

String resultado = "";

try {

switch (idBotaoSelecionado) {

case R.id.btnListar:

novoPath = PATH + LISTAR;

resultado = converseGet(HOST, 80, novoPath);

apresentaResultadoNoTextView(textResultado, resultado);

break;

case R.id.btnAdicionar:

novoPath = PATH + INSERIR + URLEncoder.encode(textLembrete.getText().toString());

resultado = converseGet(HOST, 80, novoPath);

apresentaResultadoNoTextView(textResultado, resultado);

break;

case R.id.btnListarProdutos:

novoPath = PATH\_PRODUTO + LISTAR\_PRODUTO;

resultado = converseGet(HOST, 80, novoPath);

listarJson(resultado);

break;

case R.id.btnCriarJson:

novoPath = PATH\_PRODUTO + INSERIR\_PRODUTO;

resultado = conversePostJson(HOST, 80, novoPath, criarJson());

apresentaResultadoNoTextView(textResultado, resultado);

break;

default:

break;

}

} catch (Exception e){

Log.e(CATEGORIA, e.getMessage(), e);

} finally{

dialog.dismiss();

}

}

private void apresentaResultadoNoTextView(TextView textResultado, String resultado) {

textResultado.setVisibility(View.VISIBLE);

textResultado.setText("Resultado: " + resultado);

Log.i(CATEGORIA, String.valueOf(resultado));

}

public static String converseGet(String host, int port, String path)

throws IOException {

HttpHost target = new HttpHost(host, port);

HttpClient client = new DefaultHttpClient();

HttpGet get = new HttpGet(path);

HttpEntity results = null;

try {

HttpResponse response = client.execute(target, get);

results = response.getEntity();

return EntityUtils.toString(results);

} catch (Exception e) {

Log.e(CATEGORIA, e.getMessage(), e);

throw new RuntimeException("Nao encontrou o webservice " + e.getMessage());

} finally {

if (results != null) {

try {

results.consumeContent();

} catch (IOException e) {

}

}

}

}

public static String conversePostJson(String host, int port, String path, JSONObject json)

throws IOException {

HttpHost target = new HttpHost(host, port);

HttpClient client = new DefaultHttpClient();

HttpPost post = new HttpPost(path);

HttpEntity results = null;

StringEntity se = new StringEntity(json.toString());

post.setEntity(se);

post.setHeader("Accept", "application/json");

post.setHeader("Content-type", "application/json");

try {

HttpResponse response = client.execute(target, post);

results = response.getEntity();

return EntityUtils.toString(results);

} catch (Exception e) {

Log.e(CATEGORIA, e.getMessage(), e);

throw new RuntimeException("Nao encontrou o webservice " + e.getMessage());

} finally {

if (results != null) {

try {

results.consumeContent();

} catch (IOException e) {

}

}

}

}

private void listarJson(String resultado) throws JSONException {

JSONObject root = new JSONObject(resultado);

JSONArray produtos = root.getJSONArray("produtos");

for (int i = 0; i < produtos.length(); i++) {

JSONObject c = produtos.getJSONObject(i);

String nome = c.getString("nome");

Integer quantidade = c.getInt("quantidade");

Log.i(CATEGORIA, nome + "-" + quantidade);

}

}

private JSONObject criarJson() {

try {

JSONObject produto2 = new JSONObject();

produto2.put("nome", "Zenphone");

produto2.put("quantidade", 11);

Log.i(CATEGORIA, produto2.toString());

return produto2;

} catch (JSONException e) {

e.printStackTrace();

}

return null;

}

}

**RODE A APLICAÇÃO E VERIFIQUE SE ESTÁ INSERINDO E LISTANDO**

Nova biblioteca para Android 6.0🡪 volley

http://www.androidhive.info/2014/05/android-working-with-volley-library-1/