JAVA

**Backend REST 1 - Preparação do ambiente/Criação do Projeto (Jersey + MySQL + Eclipse)**

**Criação do projeto**

A partir de uma workspace:

**File/New/Project/Maven Project**

Buscar o catálogo:

**jersey-quickstart-webapp**

Obs: caso não seja encontrado, adicionar o catálogo:

Preferences/Maven/Archetypes/Add remote catalog

**Catalog file:** [**https://repo.maven.apache.org/maven2/archetype-catalog.xml**](https://repo.maven.apache.org/maven2/archetype-catalog.xml)

**Description: <some description>**

A versão 3.0.3 do Jersey é compatível com versões a partir do Tomcat 10

**Adicionar o servidor do Tomcat**

Caso seja utilizado o tomcat 9, é necessário alterar a versão do Jersey no arquivo pom.xml

**2.35**

Os pacotes na classe responsável pelos end-points dever ser trocados para:

[**javax.ws.rs**](http://javax.ws.rs/)

**Backend REST 2 - Banco de dados**

**Criação do banco e tabelas**

CREATE DATABASE db\_expo;

USE db\_expo;

CREATE TABLE usuario(

id int PRIMARY KEY auto\_increment,

email VARCHAR(50) NOT NULL UNIQUE,

nome VARCHAR(50) NOT NULL,

nascimento DATE NOT NULL

);

**Carga de teste nas tabelas**

USE db\_expo;

INSERT INTO usuario(email, nome, nascimento) [VALUES('usuario\_p@hotmail.com](mailto:VALUES('usuario_p@hotmail.com)', 'Primeiro Usuario', '2021-11-06');

INSERT INTO usuario(email, nome, nascimento) [VALUES('usuario\_d@hotmail.com](mailto:VALUES('usuario_d@hotmail.com)', 'Segundo Usuario', '1980-03-10');

INSERT INTO usuario(email, nome, nascimento) [VALUES('usuario\_t@hotmail.com](mailto:VALUES('usuario_t@hotmail.com)', 'Outro Usuario', '1971-10-14');

**Backend REST 3 - Classes de acesso ao banco**

Adicionar a dependência do driver mysql no arquivo pom do projeto

**<dependency>**

**<groupId>mysql</groupId>**

**<artifactId>mysql-connector-java</artifactId>**

**<version>8.0.27</version>**

**</dependency>**

Criar um novo pacote para as classes de banco de dados:

**dao**

Adicionar uma nova classe:

**Dao**

**Implementar a classe de conexão:**

public class Dao {

public static String status = "Desconectado...";

public static Connection getConexao() {

Connection cnx = null;

String driverName = "com.mysql.cj.jdbc.Driver";

try {

Class.forName(driverName);

String serverName = "localhost:3306";

String dataBase = "db\_expo";

String url = "jdbc:mysql://" + serverName + "/" + dataBase;

String user = "root";

String password = "";

cnx = DriverManager.getConnection(url, user, password);

if (cnx != null) {

status = "STATUS ---> Conectado com sucesso!";

} else {

status = "STATUS ---> Não foi possível realizar a conexão.";

}

return cnx;

} catch (ClassNotFoundException e) {

System.out.println("O driver especificado não foi encontrado.");

e.printStackTrace();

return null;

} catch (SQLException e) {

System.out.println("Não foi possível conectar ao banco de dados.");

e.printStackTrace();

return null;

}

}

public static String statusConexao() {

return status;

}

public static boolean fecharConexao() {

try {

Dao.getConexao().close();

return true;

} catch (SQLException e) {

return false;

}

}

public static Connection reiniciarConexao() {

fecharConexao();

return getConexao();

}

}

**Implementar a classe xxxdao:**

public class UsuarioDao {

@SuppressWarnings("finally")

public List<Usuario> listar(){

Dao dao = new Dao();

Connection cnx = dao.getConexao();

StringBuilder sql = new StringBuilder();

sql.append("SELECT \* FROM usuario");

PreparedStatement ps;

List<Usuario> lista = new ArrayList<Usuario>();

Usuario usuario;

try {

ps = cnx.prepareStatement(sql.toString());

ResultSet rs = ps.executeQuery();

while (rs.next()) {

usuario = new Usuario();

usuario.setId(rs.getInt("id"));

usuario.setEmail(rs.getString("email"));

usuario.setNome(rs.getString("nome"));

usuario.setNascimento(rs.getDate("nascimento"));

lista.add(usuario);

}

} catch (SQLException e) {

e.printStackTrace();

} finally {

dao.fecharConexao();

return lista;

}

}

}

**Implementar a classe de modelo:**

public class Usuario {

private int id;

private String email;

private String nome;

private Date nascimento;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public Date getNascimento() {

return nascimento;

}

public void setNascimento(Date nascimento) {

this.nascimento = nascimento;

}

}

**Implementar a classe de resource:**

@Path("usuario")

public class UsuarioResource {

@GET

@Produces(MediaType.APPLICATION\_JSON)

@Path("listar")

public Response getUsuarios() {

UsuarioService service = new UsuarioService();

List<Usuario> lista = service.listar();

Response response = Response.ok().entity(lista).build();

return response;

}

}