**Laboratório API RESTful**

Usaremos o Jakarta EE (JAX-RS) e EJB Stateless, e rodar em um servidor Java EE, o WildFly.

**1. Configuração do pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.exemplo</groupId>
   <artifactId>HelloAPI</artifactId>
   <version>1.0-SNAPSHOT</version>

   <dependencies>
      <!-- Dependência para Jakarta EE (JAX-RS e EJB) -->
      <dependency>
         <groupId>jakarta.platform</groupId>
         <artifactId>jakarta.jakartaee-api</artifactId>
         <version>10.0.0</version>
         <scope>provided</scope>
      </dependency>
   </dependencies>

   <build>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-war-plugin</artifactId>
            <version>3.3.2</version>
            <configuration>
               <failOnMissingWebXml>false</failOnMissingWebXml>
            </configuration>
         </plugin>
      </plugins>
   </build>
</project>
```

**2. Implementação do Código**

**2.1 Criando um EJB Stateless**

Crie o arquivo HelloService.java em src/main/java/com/exemplo:

```
package com.exemplo;

import javax.ejb.Stateless;

// EJB Stateless que retorna uma mensagem
@Stateless
public class HelloService {
```

```
    public String sayHello(String name) {
        return "Olá, " + name + "! Bem-vindo à API EJB.";
    }
}
```

### 2.2 Criando a API RESTful (JAX-RS)

Crie o arquivo HelloResource.java em src/main/java/com/exemplo:

```
package com.exemplo;

import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/hello")
public class HelloResource {

    @Inject
    private HelloService helloService;

    @GET
    @Path("/{name}")
    @Produces(MediaType.TEXT_PLAIN)
    public String sayHello(@PathParam("name") String name) {
        return helloService.sayHello(name);
    }
}
```

### 2.3 Configuração da Aplicação JAX-RS

```
package com.exemplo;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

// Define o caminho base da API REST
@ApplicationPath("/api")
public class RestApplication extends Application {
}
```

## 3. Implantação no WildFly

Para rodar essa aplicação, você precisa de um servidor de aplicações Jakarta EE, como WildFly ou Payara.

### 3.1 Compilar o Projeto

```
mvn clean package
```

### 3.2 Deploy do WAR no Servidor

Copie o arquivo gerado target/HelloAPI.war para a pasta standalone/deployments/ do WildFly.

cp target/HelloAPI.war /caminho/para/wildfly/standalone/deployments/

### 4. Testando a API

Se o servidor estiver rodando, a API estará disponível em:

http://localhost:8080/HelloAPI/api/hello/{nome}

Testando no Navegador ou cURL

curl http://localhost:8080/HelloAPI/api/hello/Usuário

Resposta esperada:

Olá, Usuário! Bem-vindo à API EJB.

### 5. Como Funciona

O EJB Stateless fornece a lógica de negócio.
A API JAX-RS chama o EJB para retornar uma resposta RESTful.
O cliente acessa a API via HTTP (/api/hello/{name}).
O servidor responde "Olá, {name}! Bem-vindo à API EJB.".