

Relatório do Laboratório API

Data: 24/02/2025

Autor: *Francisco Jarbas dos Santos Sousa*

Ambiente de Desenvolvimento: Windows 10, Eclipse IDE, WildFly 35, Maven

1. Introdução

Este relatório documenta todo o processo de configuração, desenvolvimento e implantação de uma API RESTful utilizando **Jakarta EE** e **EJB Stateless** no servidor de aplicações **WildFly 35.0.1.Final**. O objetivo foi criar uma API funcional e realizar sua execução dentro do ambiente configurado.

2. Configuração do Ambiente

Para garantir um desenvolvimento eficiente, foi necessário configurar corretamente o ambiente de trabalho:

2.1 Instalação do Apache Maven

- Verificação da instalação com:
- `mvn -version`
- Configuração das variáveis de ambiente no Windows:
 - `MAVEN_HOME` apontando para a pasta do Maven
 - Inclusão do diretório `bin` do Maven no `PATH`

2.2 Instalação e Configuração do WildFly

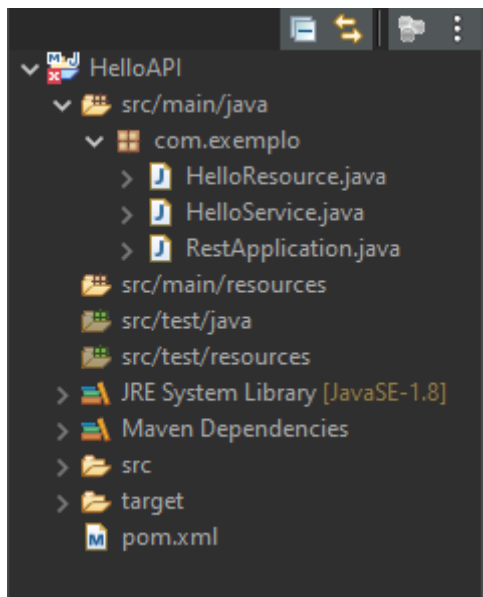
- Download do WildFly 35.0.1.Final.
- Configuração no Eclipse via **JBoss Tools**.
- Criação de um usuário administrador no WildFly:
- `C:\wildfly\bin\add-user.bat`
- Teste de acesso ao console administrativo:
- `http://localhost:9990`

3. Desenvolvimento da API RESTful

O desenvolvimento foi baseado em **Jakarta EE 10**, utilizando **JAX-RS** para expor a API e **EJB Stateless** para a lógica de negócios.

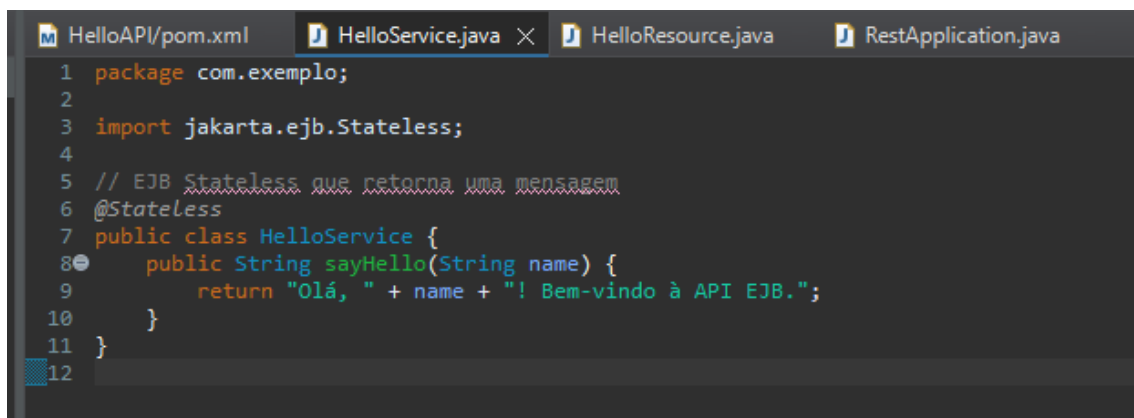
3.1 Estrutura do Projeto

O projeto foi criado como um **Maven Web Application (WAR)** com a seguinte estrutura:

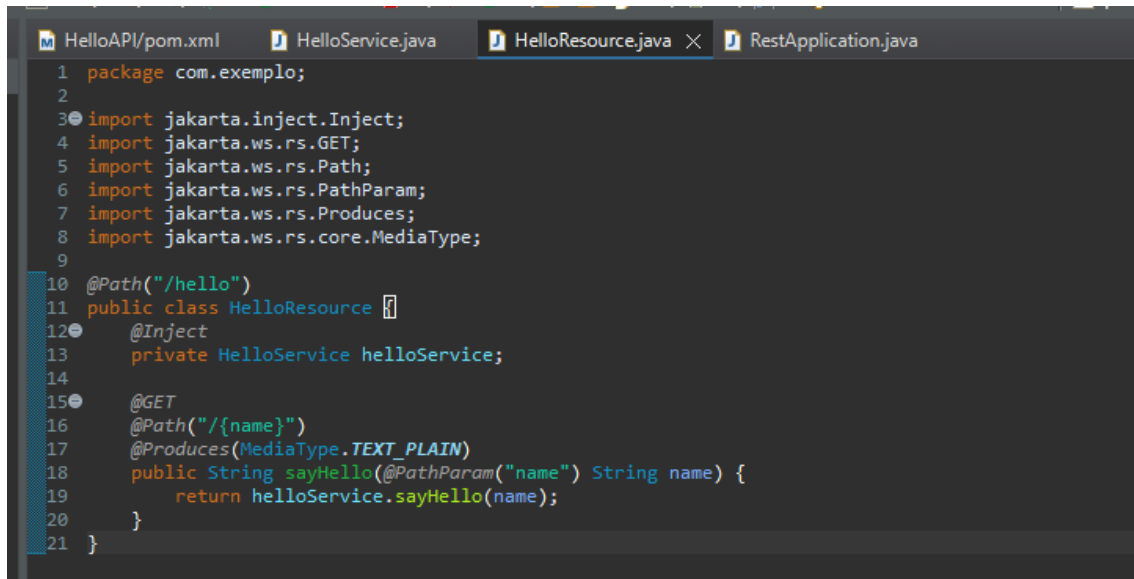


3.2 Implementação do Código

3.2.1 Classe HelloService.java (EJB Stateless)

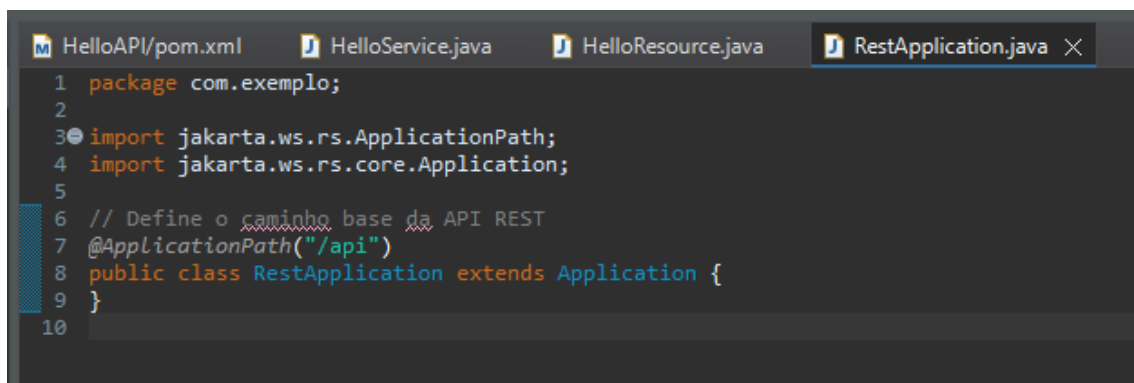


3.2.2 Classe HelloResource.java (API REST)



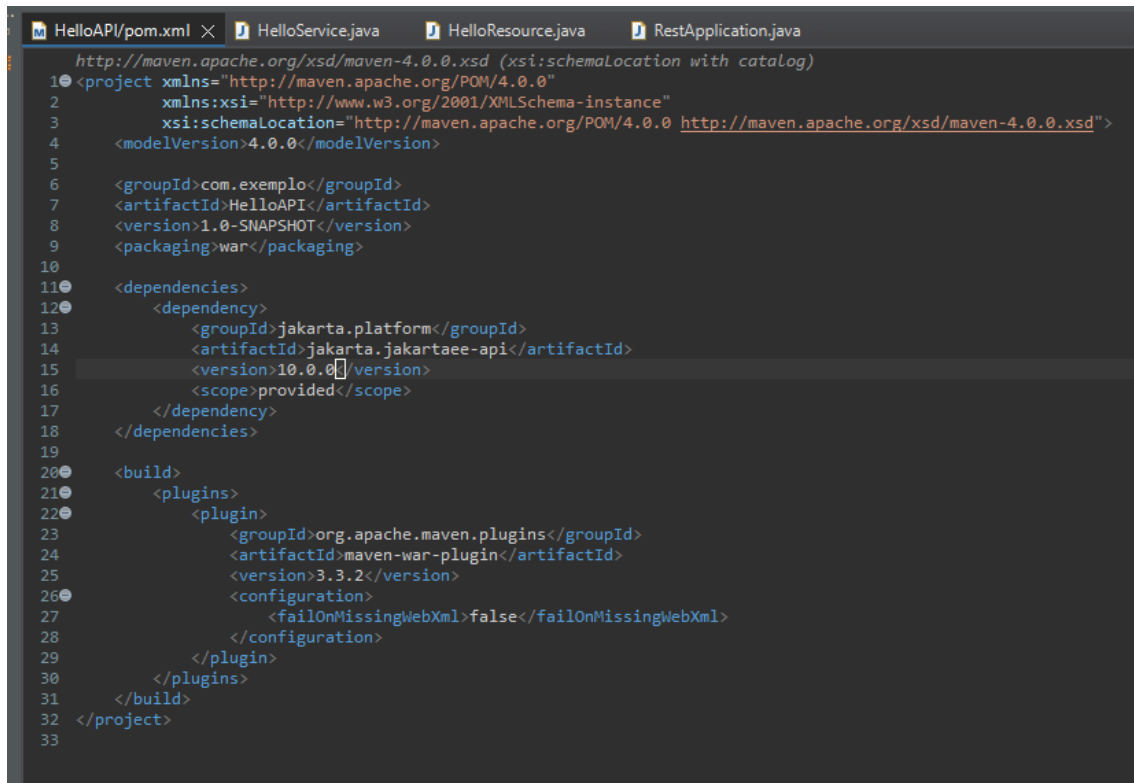
```
1 package com.exemplo;
2
3 import jakarta.inject.Inject;
4 import jakarta.ws.rs.GET;
5 import jakarta.ws.rs.Path;
6 import jakarta.ws.rs.PathParam;
7 import jakarta.ws.rs.Produces;
8 import jakarta.ws.rs.core.MediaType;
9
10 @Path("/hello")
11 public class HelloResource {
12     @Inject
13     private HelloService helloService;
14
15     @GET
16     @Path("/{name}")
17     @Produces(MediaType.TEXT_PLAIN)
18     public String sayHello(@PathParam("name") String name) {
19         return helloService.sayHello(name);
20     }
21 }
```

3.2.3 Classe RestApplication.java (Configuração JAX-RS)



```
1 package com.exemplo;
2
3 import jakarta.ws.rs.ApplicationPath;
4 import jakarta.ws.rs.core.Application;
5
6 // Define o caminho base da API REST
7 @ApplicationPath("/api")
8 public class RestApplication extends Application {
9 }
10
```

3.2.4 Configuração do pom.xml



```
1  http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5
6      <groupId>com.exemplo</groupId>
7      <artifactId>HelloAPI</artifactId>
8      <version>1.0-SNAPSHOT</version>
9      <packaging>war</packaging>
10
11     <dependencies>
12         <dependency>
13             <groupId>jakarta.platform</groupId>
14             <artifactId>jakarta.jakartaee-api</artifactId>
15             <version>10.0.0</version>
16             <scope>provided</scope>
17         </dependency>
18     </dependencies>
19
20     <build>
21         <plugins>
22             <plugin>
23                 <groupId>org.apache.maven.plugins</groupId>
24                 <artifactId>maven-war-plugin</artifactId>
25                 <version>3.3.2</version>
26                 <configuration>
27                     <failOnMissingWebXml>false</failOnMissingWebXml>
28                 </configuration>
29             </plugin>
30         </plugins>
31     </build>
32 </project>
33
```

4. Implantação no WildFly

Após a implementação, a API foi implantada no **WildFly 35.0.1.Final**.

Coloquei manualmente o arquivo HelloAPI-1.0-SNAPSHOT.war na pasta

C:\wildfly\standalone\deployments

4.1 Geração do Arquivo .war

Para empacotar o projeto como um .war, foi utilizado o Maven:

mvn clean package

O arquivo gerado foi salvo em:

target/HelloAPI-1.0-SNAPSHOT.war

4.2 Deploy Manual no WildFly

O .war foi copiado para a pasta de deploy do WildFly:

```
cp target/HelloAPI-1.0-SNAPSHOT.war C:\wildfly\standalone\deployments\
```

Com o WildFly em execução, o sistema reconheceu automaticamente o deploy.

5. Testes da API

Para testar o funcionamento da API REST, foram realizadas requisições via navegador e **cURL**.

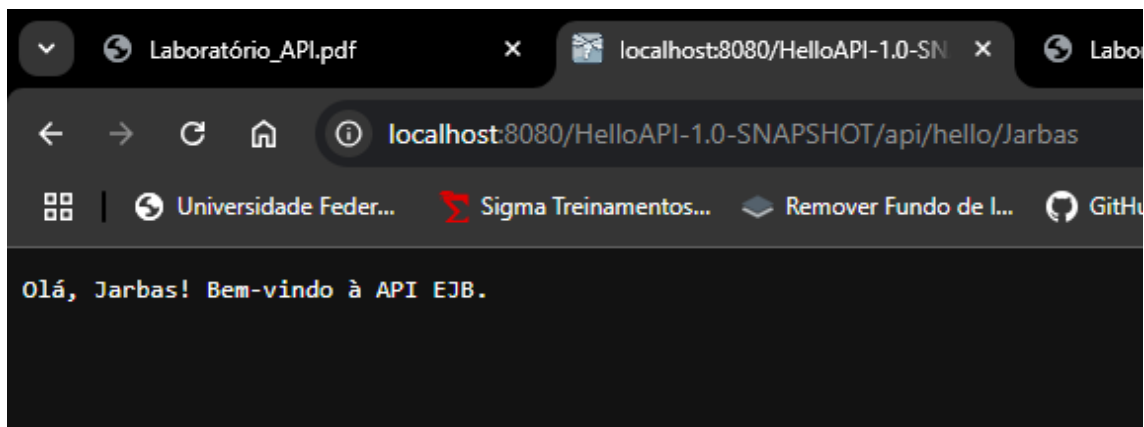
5.1 Teste pelo Navegador

Acessando a API via **GET**:

`http://localhost:8080/HelloAPI-1.0-SNAPSHOT/api/hello/Jarbas`

Retorno esperado:

Olá, Jarbas! Bem-vindo à API EJB.



5.2 Teste via cURL

`curl http://localhost:8080/HelloAPI-1.0-SNAPSHOT/api/hello/Jarbas`

Retorno esperado:

Olá, Jarbas! Bem-vindo à API EJB.

```
Microsoft Windows [versão 10.0.19045.5487]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\jarbe>curl http://localhost:8080/HelloAPI-1.0-SNAPSHOT/api/hello/Jarbas
Olá, Jarbas! Bem-vindo à API EJB.
C:\Users\jarbe>
```

6. Conclusão

O processo de desenvolvimento, configuração e implantação da API RESTful foi realizado com sucesso. Utilizando **WildFly 35, Jakarta EE 10, JAX-RS e EJB Stateless**, conseguimos criar uma aplicação funcional e testá-la em um ambiente de produção.

Resultados Alcançados

- Ambiente de desenvolvimento **configurado corretamente**.
- API REST desenvolvida **seguindo padrões modernos**.
- **Deploy no WildFly 35 funcionando sem erros**.
- Testes bem-sucedidos via navegador e cURL.

O projeto tá **pronto pra melhorar ou aumentar no futuro**, por exemplo com integração de **banco de dados** ou implementação de **autenticação com JWT e até JSON**.