Roteiro de Atividades – *Webservices*

1. Instalação das ferramentas

1 Instalar o Eclipse para desenvolvimento J2EE (disponível em https://www.eclipse.org/downloads/packages/release/2019-06/r/eclipse-ide-enterprise-java-developers)

2 Instalar o Apache Tomcat (versão recomendável: 8.5.x)
- 2.1 Fazer o download do arquivo disponível em http://ftp.unicamp.br/pub/apache/tomcat/tomcat-8/v8.5.39/bin/apache-tomcat-8.5.39.zip
- 2.2 Descompactar em alguma pasta
- 2.3 Executar o arquivo *startup.sh* (Linux) ou *startup.bat* (Windows)
- 2.4 Em um navegador, acessar http://localhost:8080

3 Instalar o cliente http POSTMAN.
- 3.1 Em linux, no terminal, utilizar o comando `snap install postman`
- 3.2 Em outros sistemas operacionais, acessar https://www.getpostman.com/downloads/

2. Atividades – Parte 1

1 No eclipse, criar um novo *Dynamic Web Project*

2 No projeto recém criado, verificar se o arquivo *web.xml* existe dentro da pasta *WebContent>WEB-INF.* Se não existir, clicar com o botão direito sobre o projeto e selecionar a opção *Java EE Tools > Generate Deployment Descriptor Stub.*

3 Ajustar o conteúdo do arquivo *web.xml* para que o conteúdo fique conforme o seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        version="3.0">
        <display-name>tutorial_jaxrs</display-name>
        <servlet>
                <servlet-name>blu3024</servlet-name>
                        <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
                        <load-on-startup>1</load-on-startup>
        </servlet>
        <servlet-mapping>
                        <servlet-name>blu3024</servlet-name>
                        <url-pattern>/*</url-pattern>
        </servlet-mapping>

</web-app>
```

4 Transformar o projeto em um projeto *Maven*: clicar sobre o projeto com o botão direito do *mouse* e selecionar a opção *Configure > Convert to Maven Project*.

5 Adicionar as dependências necessárias ao arquivo *pom.xml*, que deve ter o seguinte conteúdo:

```
<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>Teste_REST_1</groupId>
        <artifactId>Teste_REST_1</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>war</packaging>
        <build>
                <sourceDirectory>src</sourceDirectory>
                <plugins>
                        <plugin>
                                <artifactId>maven-compiler-plugin</artifactId>
                                <version>3.7.0</version>
                                <configuration>
                                        <source>1.8</source>
                                        <target>1.8</target>
                                </configuration>
                        </plugin>
                        <plugin>
                                <artifactId>maven-war-plugin</artifactId>
                                <version>3.2.1</version>
                                <configuration>
                                        <warSourceDirectory>WebContent</warSourceDirectory>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
        <dependencies>
                <dependency>
                        <groupId>com.sun.jersey</groupId> <!-- javax.* classes -->
                        <artifactId>jersey-bundle</artifactId>
                        <version>1.19.4</version>
                </dependency>
                <dependency>
                        <groupId>com.owlike</groupId> <!-- for handling JSON media type -->
                        <artifactId>genson</artifactId>
                        <version>0.99</version>
                </dependency>
        </dependencies>
</project>
```

6 Atualizar as dependências: clicar com o botão direito do *mouse* sobre o projeto e selecionar a opção *Maven>Update Project.*

7 Criar uma classe Java para implementar o webservice: clicar sobre o projeto com o botão direito do *mouse* e selecionar a opção *New>Class.*

8 Implementar a classe conforme a seguir:

```
public class Aluno {

	private int id;
	private String nome;
	private String curso;
	private int idade;


	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public String getCurso() {
		return curso;
	}

	public void setCurso(String curso) {
		this.curso = curso;
	}

	public int getIdade() {
		return idade;
	}

	public void setIdade(int idade) {
		this.idade = idade;
	}


	@Override
	public String toString() {
		return "Aluno [id=" + id + ", nome=" + nome + ", curso=" + curso +
", idade=" + idade + "]";
	}
```

```java
public class AlunoDAO {

	private static final AlunoDAO instance = new AlunoDAO();
	private static List<Aluno> alunos = new ArrayList<Aluno>();


	private AlunoDAO() {

	}

	public static AlunoDAO getInstance() {
		return instance;
	}

	public void add(Aluno aluno) {
		alunos.add(aluno);
	}

	public Aluno first() {
		return alunos.get(0);
	}


	public static Aluno getById(int id) {
		for(Aluno a:alunos) {
			if(a.getId()==id) {
				return a;
			}
		}
		return null;
	}

}
```

```java
@Path("/teste")
public class Teste_WS {

		@PUT
		@Path("/addaluno")
		@Consumes("text/plain")
		public void add(String nome) {
			Aluno aluno = new Aluno();
			aluno.setId(1);
			aluno.setNome(nome);
			aluno.setCurso("eca");
			aluno.setIdade(99);
			AlunoDAO.getInstance().add(aluno);
		}

		@GET
		@Path("/getfirstjson")
		@Produces(MediaType.APPLICATION_JSON)
		public Aluno primeiroJson() {
			AlunoDAO alunos = AlunoDAO.getInstance();
			return alunos.first();
		}

		@PUT
		@Path("/addalunojson")
		@Consumes(MediaType.APPLICATION_JSON)
		public void addJson(Aluno aluno) {
			AlunoDAO.getInstance().add(aluno);
		}

}
```

9 Utilizar os *webservices* implementados para adicionar alunos e para consultar os alunos adicionados;

10 Implementar *webservices* para
- 10.1 Adicionar alunos no fim da lista;
- 10.2 Retornar os dados do último aluno;
- 10.3 Retornar a quantidade de alunos cadastrados.
- 10.4 Retornar uma lista de alunos;