Missão Prática - Nível 4 Mundo 3

Campus : POLO COPACABANA

Curso : DESENVOLVIMENTO FULL STACK

Disciplina : RPG0017 - Vamos integrar sistemas

Turma : 9003

Semestre: 2023.3 FLEX

Nome : MARCO SERGIO ALBINO VITTORIO BAROZZI

1º Procedimento | Camadas de Persistência e Controle

Objetivos:

1-Implementar persistência com base em JPA.

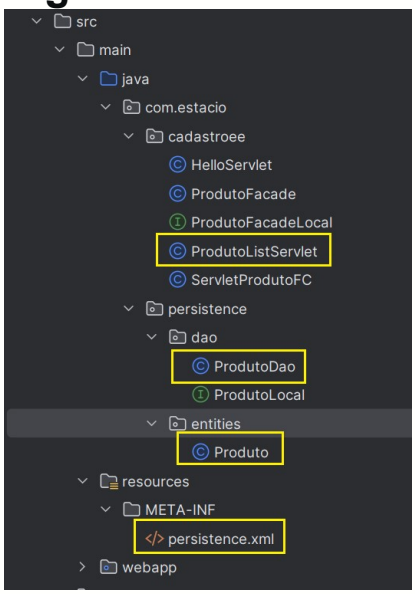
2-Implementar regras de negócio na plataforma JEE, através de EJBs.

3-Implementar sistema cadastral Web com base em Servlets e JSPs.

4-Utilizar a biblioteca Bootstrap para melhoria do design.

5-No final do exercício, o aluno terá criado todos os elementos necessários para exibição e entrada de dados na plataforma Java Web, tornando-se capacitado para lidar com contextos reais de aplicação.

Códigos desenvolvidos :



Classe Produto.java

```java
package com.estacio.persistence.entities;

import jakarta.persistence.*;

import java.math.BigDecimal;
import java.util.Objects;

@Entity
public class Produto {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    @Column(name = "idProduto")
    private int idProduto;
    @Basic
    @Column(name = "nome")
    private String nome;
    @Basic
    @Column(name = "quantidade")
    private Integer quantidade;
    @Basic
    @Column(name = "precoVenda")
    private BigDecimal precoVenda;

    public int getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(int idProduto) {
        this.idProduto = idProduto;
    }

    public String getNome() {
        return nome;
```

```
    }

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

    public Integer getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(Integer quantidade) {
        this.quantidade = quantidade;
    }

    public BigDecimal getPrecoVenda() {
        return precoVenda;
    }

    public void setPrecoVenda(BigDecimal precoVenda) {
        this.precoVenda = precoVenda;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Produto produto = (Produto) o;
        return idProduto == produto.idProduto && Objects.equals(nome,
produto.nome) && Objects.equals(quantidade, produto.quantidade) &&
Objects.equals(precoVenda, produto.precoVenda);
    }

    @Override
    public int hashCode() {
        return Objects.hash(idProduto, nome, quantidade, precoVenda);
    }
}
```

Classe ProdutoDao.java

```
package com.estacio.persistence.dao;

import com.estacio.persistence.entities.Produto;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import jakarta.persistence.TypedQuery;

import java.math.BigDecimal;
import java.util.List;
@Stateless
public class ProdutoDao {

    @PersistenceContext
    private EntityManager em;


    public Produto criarProduto(String nome, BigDecimal preco) {
```

```
        Produto produto = new Produto();
        produto.setNome(nome);
        produto.setPrecoVenda(preco);
        em.persist(produto);
        return produto;
    }

     public Produto buscarPorId(Long id) {
        return em.find(Produto.class, id);
    }

     public List<Produto> listarTodos() {
        //return em.createQuery("SELECT p FROM Produto p",
Produto.class).getResultList();

        TypedQuery<Produto> query= em.createQuery("SELECT p FROM Produto p",
Produto.class);
        return query.getResultList();
    }

}
```

Classe ProdutoListServlet.java

```
package com.estacio.cadastroee;
import com.estacio.persistence.dao.ProdutoDao;
import com.estacio.persistence.entities.Produto;
import jakarta.ejb.EJB;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.util.List;

@WebServlet("/SevletProduto")
public class ProdutoListServlet extends HttpServlet {

    @EJB
    private ProdutoDao produtoDao;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        try {
            List<Produto> produtos = produtoDao.listarTodos();

            resp.setContentType("text/html;charset=UTF-8");
            resp.getWriter().write("<h1>Servlet ServletProduto at /
CadastroEE-WAR</h1>");
            for (Produto produto : produtos) {
                resp.getWriter().write("<p>ID: " + produto.getIdProduto() +
", Nome: " + produto.getNome() + ", Preço: " + produto.getPrecoVenda() +
"</p>");
            }
        } catch (Exception e) {
```

```
            resp.getWriter().write("Erro ao listar produtos: " +
e.getMessage());
        }
    }
}
```

pom.xml

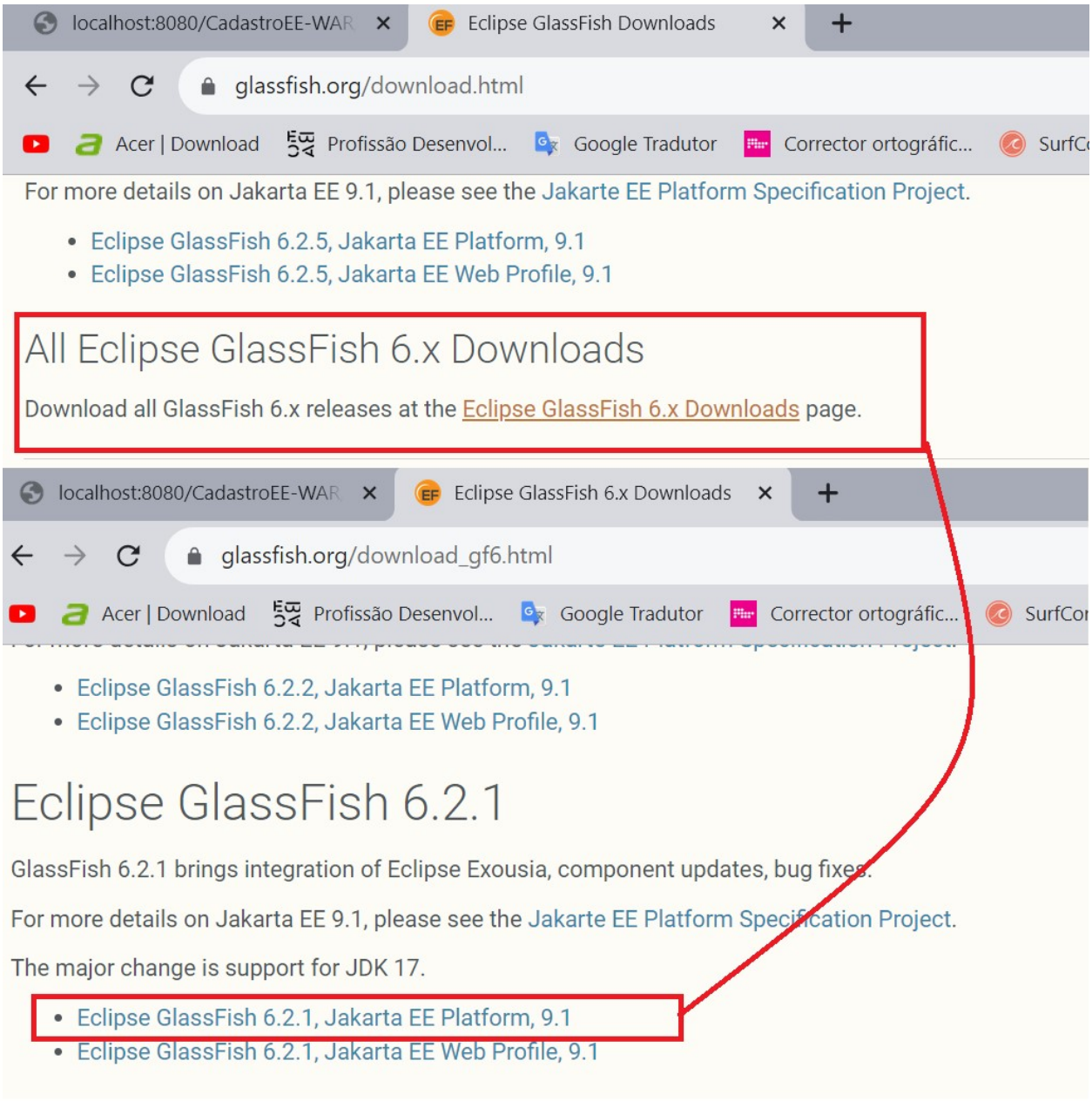
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cadastroee</groupId>
    <artifactId>CadastroEE</artifactId>
    <version>1.0-SNAPSHOT</version>
    <name>CadastroEE</name>
    <packaging>war</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.target>11</maven.compiler.target>
        <maven.compiler.source>11</maven.compiler.source>
        <junit.version>5.9.2</junit.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.eclipse.persistence</groupId>
            <artifactId>eclipselink</artifactId>
            <version>3.0.2</version>
        </dependency>
        <dependency>
            <groupId>jakarta.persistence</groupId>
            <artifactId>jakarta.persistence-api</artifactId>
            <version>3.0.0</version>
        </dependency>
        <dependency>
            <groupId>jakarta.validation</groupId>
            <artifactId>jakarta.validation-api</artifactId>
            <version>3.0.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>jakarta.ejb</groupId>
            <artifactId>jakarta.ejb-api</artifactId>
            <version>4.0.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>jakarta.servlet</groupId>
            <artifactId>jakarta.servlet-api</artifactId>
            <version>5.0.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
```

```
            <groupId>org.glassfish.web</groupId>
            <artifactId>jakarta.servlet.jsp.jstl</artifactId>
            <version>3.0.1</version>
        </dependency>


        <dependency>
            <groupId>com.microsoft.sqlserver</groupId>
            <artifactId>mssql-jdbc</artifactId>
            <version>9.4.0.jre11</version>
        </dependency>
        <dependency>
            <groupId>org.glassfish.jaxb</groupId>
            <artifactId>jaxb-runtime</artifactId>
            <version>3.0.2</version>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <finalName>CadastroEE-WAR</finalName>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <release>11</release>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

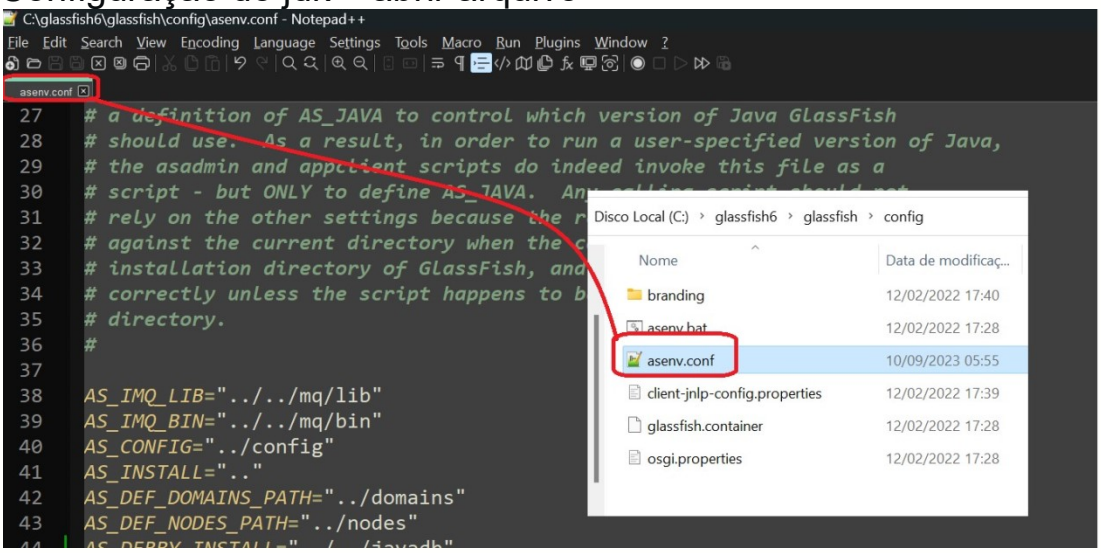
Resultados da execução dos códigos

1- Download,configuração do pool de conexão com SQL Server
Criação de recurso e start do GlassFish Server 6.2.1:

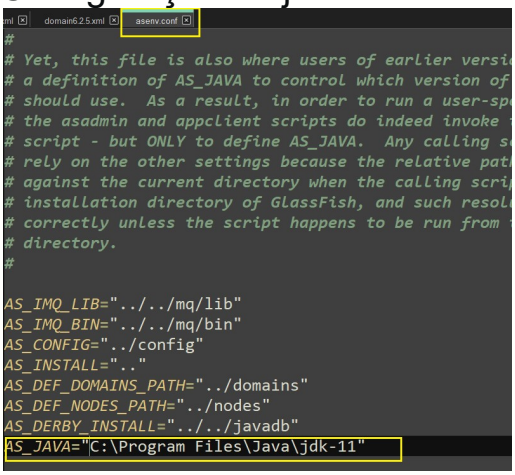
baixar servidor



Configuração do jdk – abrir arquivo

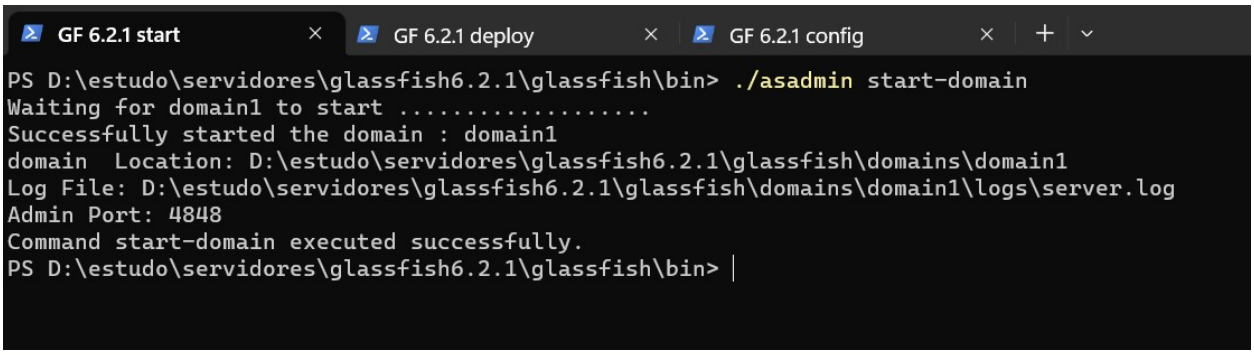


Configuração do jdk 11

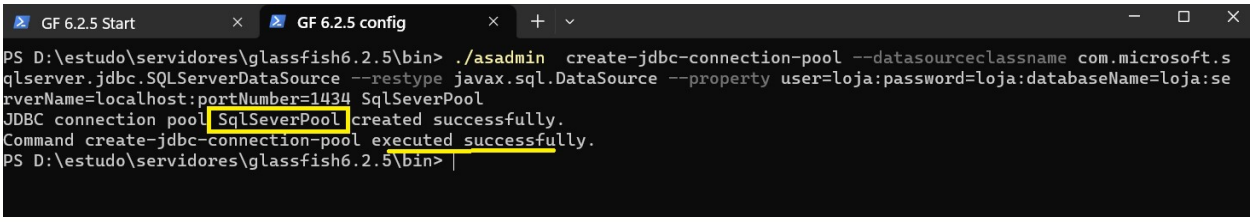


Configurando os jars de conexão no domínio a ser acionado

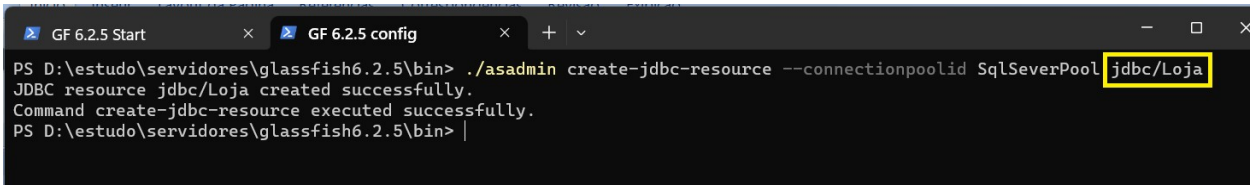
Iniciar servidor



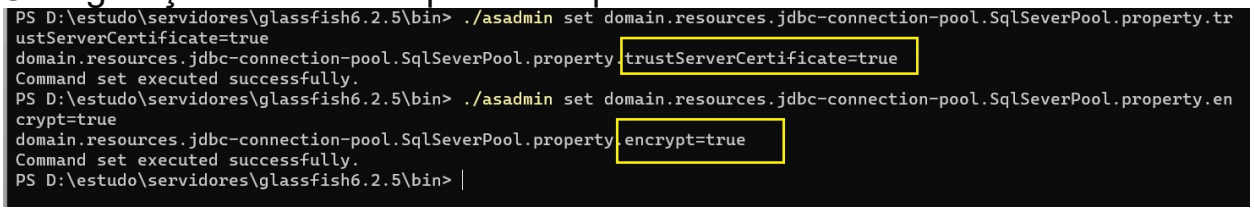
Criar conexão



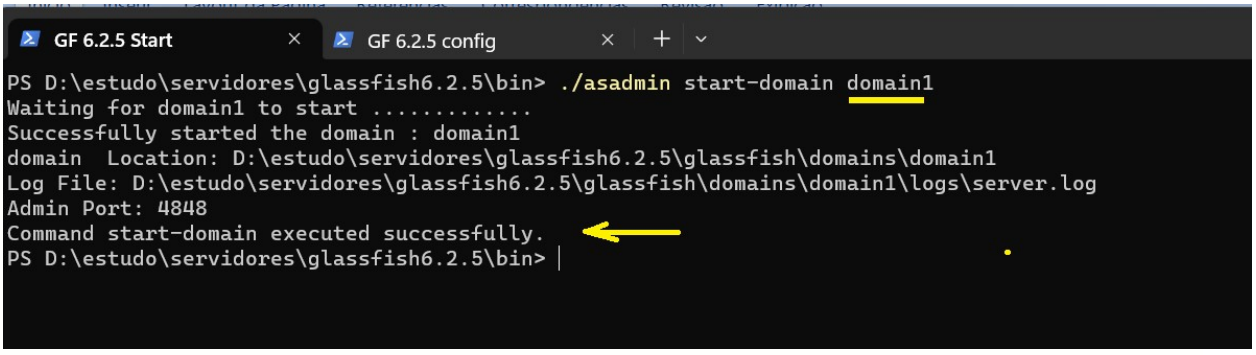
Criar recurso



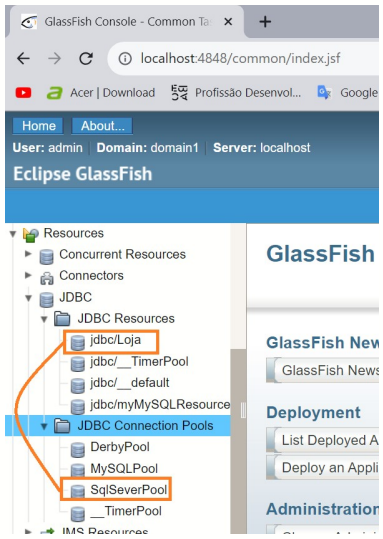
Configurações adicionais para mssql



Acionando o domínio

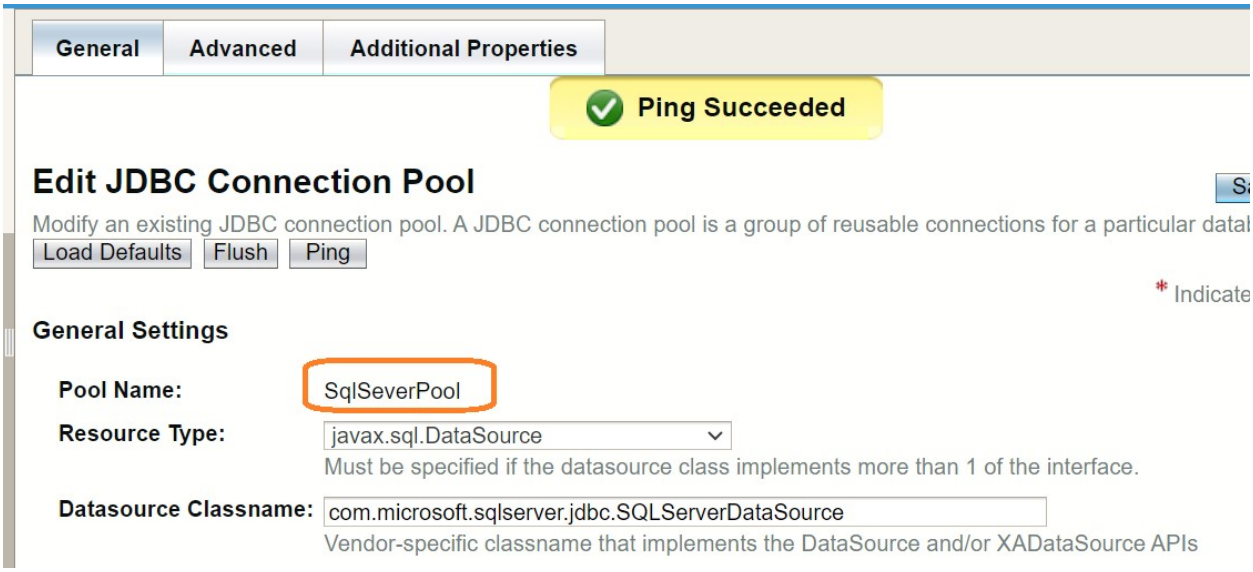


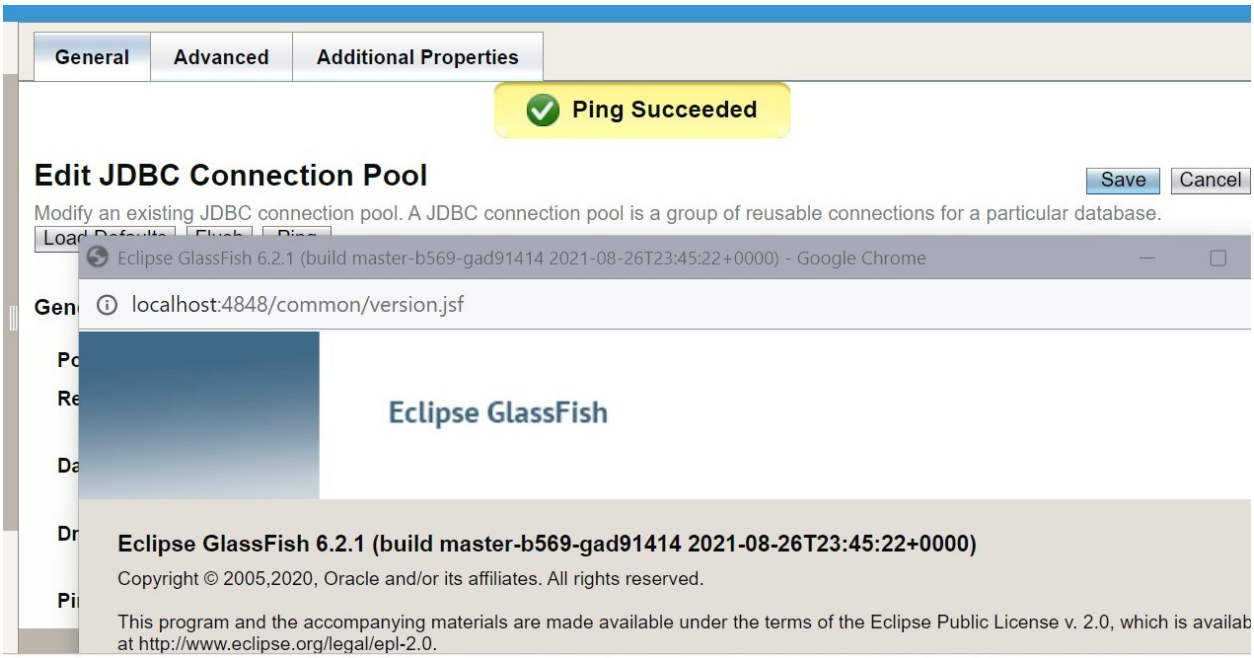
Abrindo adm



Conferindo config

Conferindo conexão SQL Server





Edit JDBC Connection Pool Properties

Modify properties of an existing JDBC connection pool.

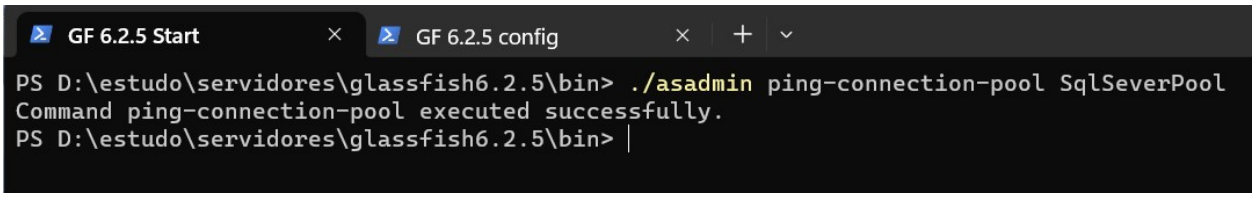
Pool Name: SqlSeverPool

Additional Properties (7)

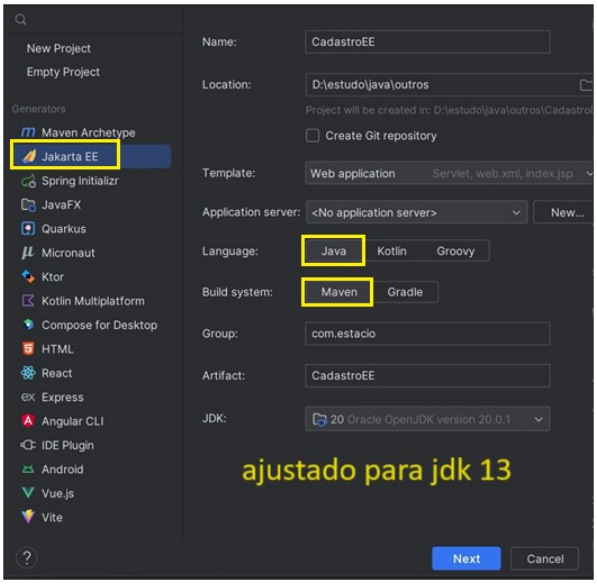
Add Property | Delete Properties

Select	Name	Value
☐	password	loja
☐	databaseName	loja
☐	serverName	localhost
☐	user	loja
☐	portNumber	1434
☐	trustServerCertificate	true
☐	encrypt	true

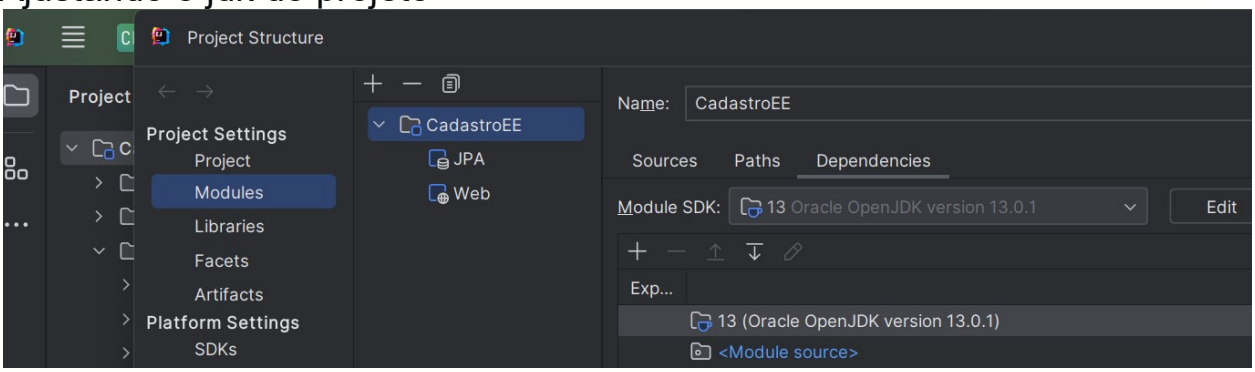
Ping conexão SQL Server via cmd



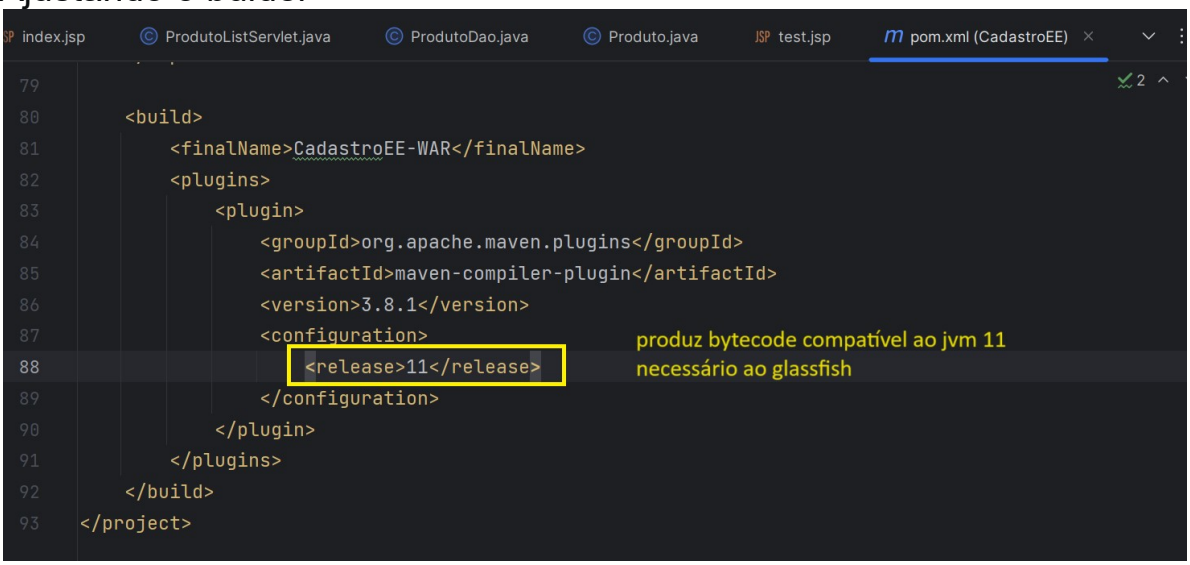
2- Criar o aplicativo corporativo



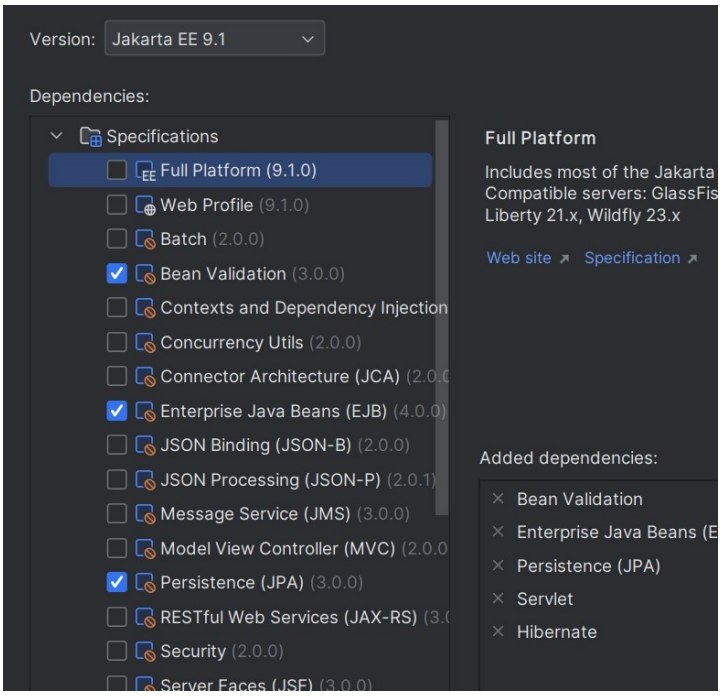
Ajustando o jdk do projeto



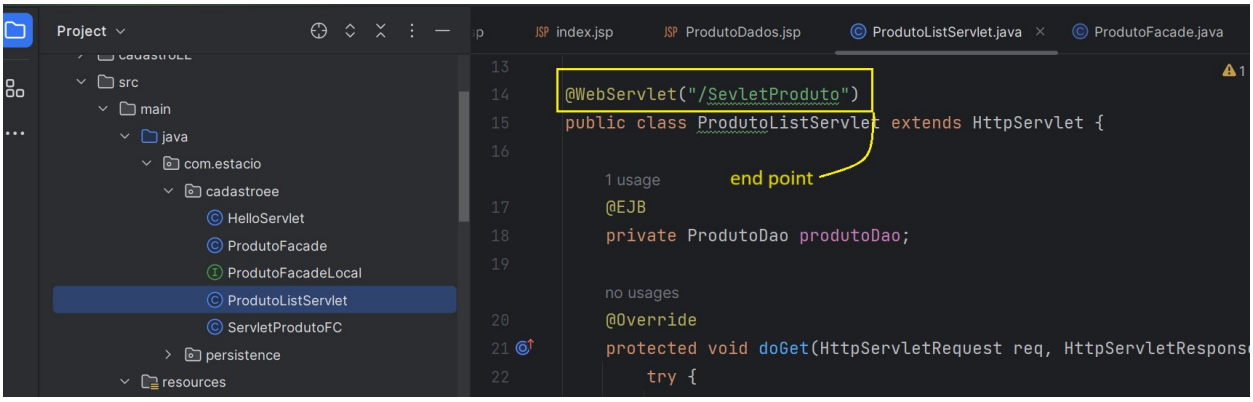
Ajustando o buider



Selecionado dependencias

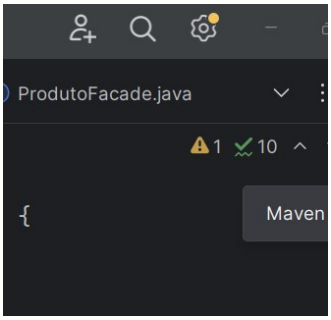


Servlet solicitado

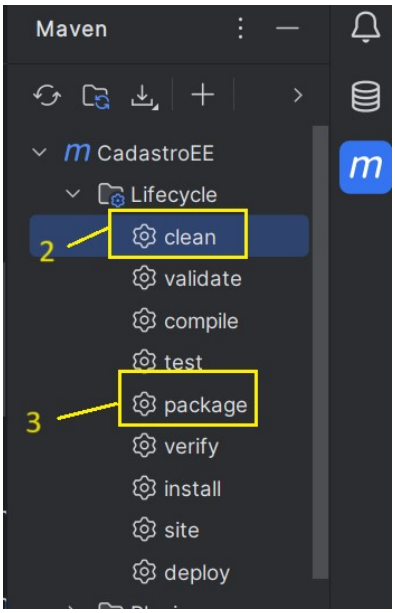


3- A execução deve ser efetuar com o uso de Run ou Deploy no projeto principal (CadastroEE)

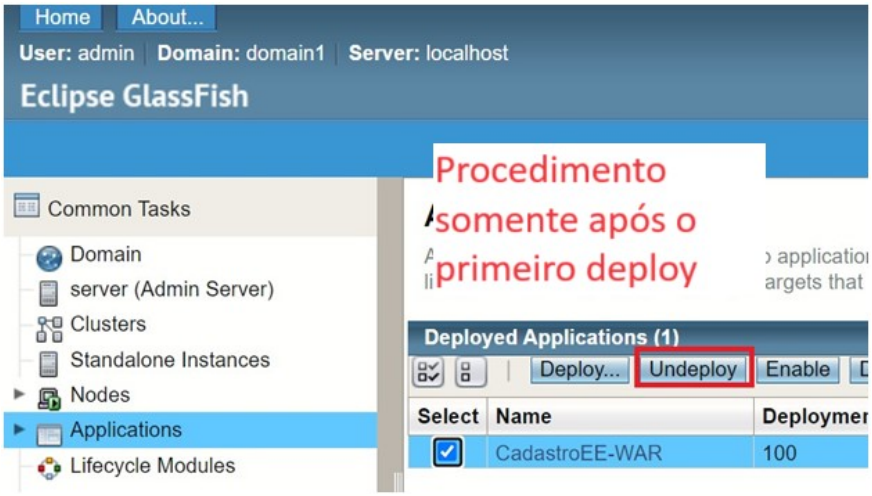
Passo 1



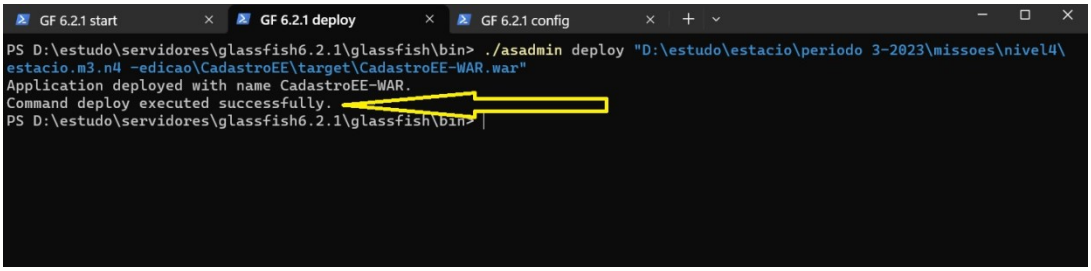
Passo 2



Passo 3

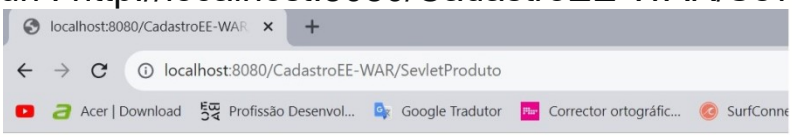


Passo 4



4- Acessar o endereço a seguir, para testar o Servlethttp://
localhost:8080/CadastroEE-war/ServletProduto
já tendo alimentado a base

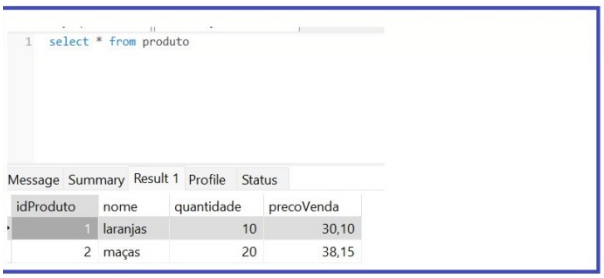
url : http://localhost:8080/CadastroEE-WAR/SevletProduto



Servlet ServletProduto at / CadastroEE-WAR

ID: 1, Nome: laranjas, Preço: 30.10

ID: 2, Nome: maças, Preço: 38.15



Análise e Conclusão:

1-Como é organizado um projeto corporativo no Intellij ?

Estrutura do Projeto como sendo pastas:

src/main/java: código fonte em Java.
src/main/resources: recursos, como arquivos de configuração.
src/test/java: código de teste.
src/test/resources: recursos usados durante os testes.

Pacotes: Dentro da estrutura src/main/java, você pode organizar seu código em pacotes para separar funcionalidades, por exemplo:

com.empresa.modelo: Classes que representam os modelos de dados.
com.empresa.servico: Classes de serviços com lógicas de negócios.
com.empresa.controlador: Controladores, por exemplo, para uma aplicação web.
Configuração de Build e Dependências:

Utilização de ferramentas como Maven ou Gradle para gerenciar as dependências e o processo de build.
.

Controle de Versão:

Utilização do .gitignore para excluir arquivos temporários e diretórios de saída.

2-Qual o papel das tecnologias JPA e EJB na construção de um aplicativo para a plataforma Web no ambiente Java?

Os EJBs lidam com a lógica de negócios e invocam a camada de persistência (JPA) para interagir com o banco de dados. A combinação dessas duas tecnologias permite que os desenvolvedores construam aplicações robustas, escaláveis e transacionais para a plataforma web no ambiente Java

3-Como o Intellij viabiliza a melhoria de produtividade ao lidar com as tecnologias JPA e EJB?

Na combinação de recursos de edição, validação, integração com bancos de dados e outras ferramentas de suporte faz da IDE uma escolha poderosa para desenvolvedores que trabalham com JPA e EJB. Estas características visam acelerar o processo de desenvolvimento, reduzir erros e melhorar a qualidade do código.

Validação:

A IDE verifica e valida automaticamente seu código JPA e EJB.

Suporte a JPQL:

Ao escrever consultas JPQL (Java Persistence Query Language).

Integração com bancos de dados:

Capacidade de conectar-se diretamente a uma fonte de dados. Com isso, se pode gerar entidades JPA a partir de tabelas existentes.

Geradores de Código:

Getters/setters, construtores e outros membros baseados em campos de classe etc.

Refatoração Avançada:

As ferramentas de refatoração da IntelliJ IDEA são inteligentes o suficiente para reconhecer padrões associados ao JPA e EJB.

Visualização Gráfica de Entidades:

A IntelliJ IDEA oferece uma visualização gráfica das entidades JPA e seus relacionamentos.

etc ...

4-O que são Servlets, e como o Intellij oferece suporte à construção desse tipo de componentes em um projeto Web?

Servlets são componentes Java que são executados no servidor e são usados para desenvolver aplicações web

Criando-se diretamente através da interface da IDE uma classe do tipo Servlet

5-Como é feita a comunicação entre os Serlvets e os Session Beans do pool de EJBs?

Uma das maneiras mais comuns de acessar EJBs em Servlets no ambiente Java EE é através da injeção de dependência usando a anotação @EJB