INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL	**Delegação Regional do Centro** **Centro de Emprego e Formação Profissional de Viseu**
	Programador/a de Informática (AÇÃO 24/2023)
	Formando/a: Judite da Silva Miguel **N.º: 3824316** **Data: 16 / 05 /2024**

Portefólio Reflexivo de Aprendizagens (PRA)

UFCD: 10791 – Desenvolvimento de aplicações web em JAVA

Número de Horas: 50 horas

Formador/a: Marco Alexandre Cunha Lucas **Assinatura:** _____

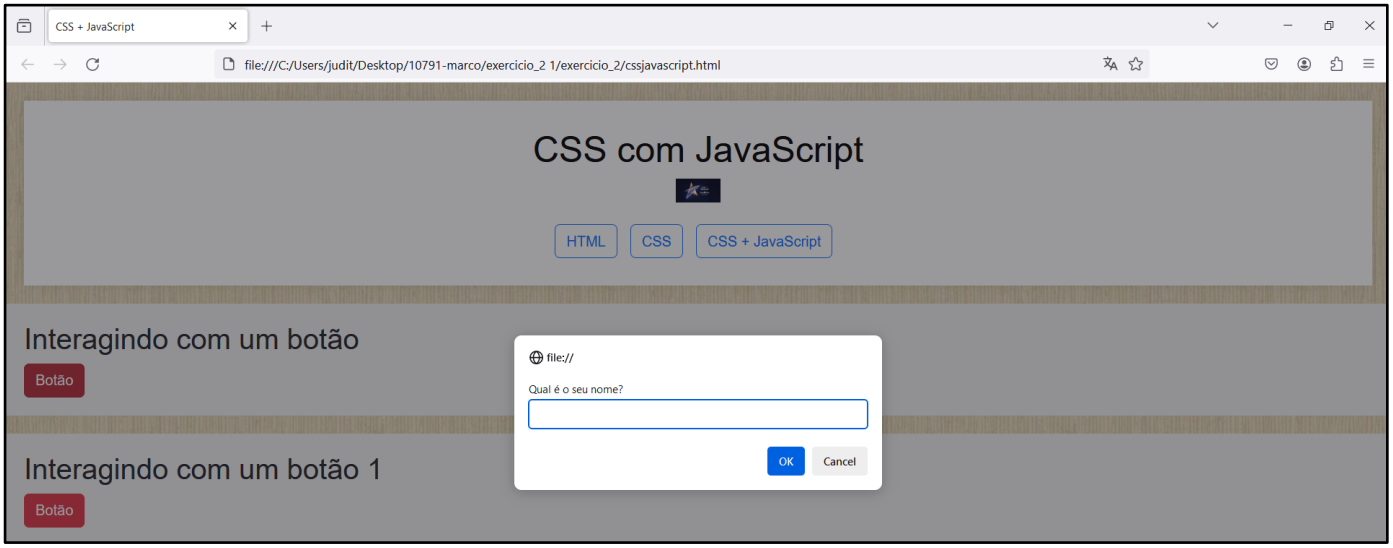
Mediadora: Carla David **Assinatura:** _____

Objetivos da UFCD:

- Implementar e criar aplicações Java EE.
- Gerir regras de negócio utilizando POJOs, EJBs, SOAP WebServices, e JMS.
- Administrar persistência utilizando JPA.
- Garantir a segurança nas aplicações web.

Nesta UFCD, começámos por abordar a introdução ao desenvolvimento de aplicações WEB, o protocolo HTTP, a plataforma Java e o Java EE. De seguida, abordámos a introdução à programação WEB, com as linguagens HTML, CSS e JavaScript. Aprofundámos a estrutura básica do HTML, os elementos que aparecem no head, body onde aprofundámos as tags de formatação de conteúdo. Fizemos alguns exercícios de consolidação de conhecimentos. De seguida abordámos o CSS (Cascading Style Sheets) e vimos como se aplica o css no html. Abordámos as frameworks que são ferramentas que auxiliam o programador a trabalhar em certa linguagem, nesta situação abordámos e instalámos o Bootstrap e aplicámos os conteúdos abordados em exercícios práticos.

Depois abordámos o JavaScript e a sua aplicação. Abordámos as variáveis, os tipos de dados, os arrays, os operadores, estruturas de seleção e as funções. Fizemos alguns exercícios abordando todo o conteúdo do JavaScript e relacionando com o HTML e o JavaScript.



Também abordámos o tema dos formulários e fizemos alguns exercícios utilizando formulários.

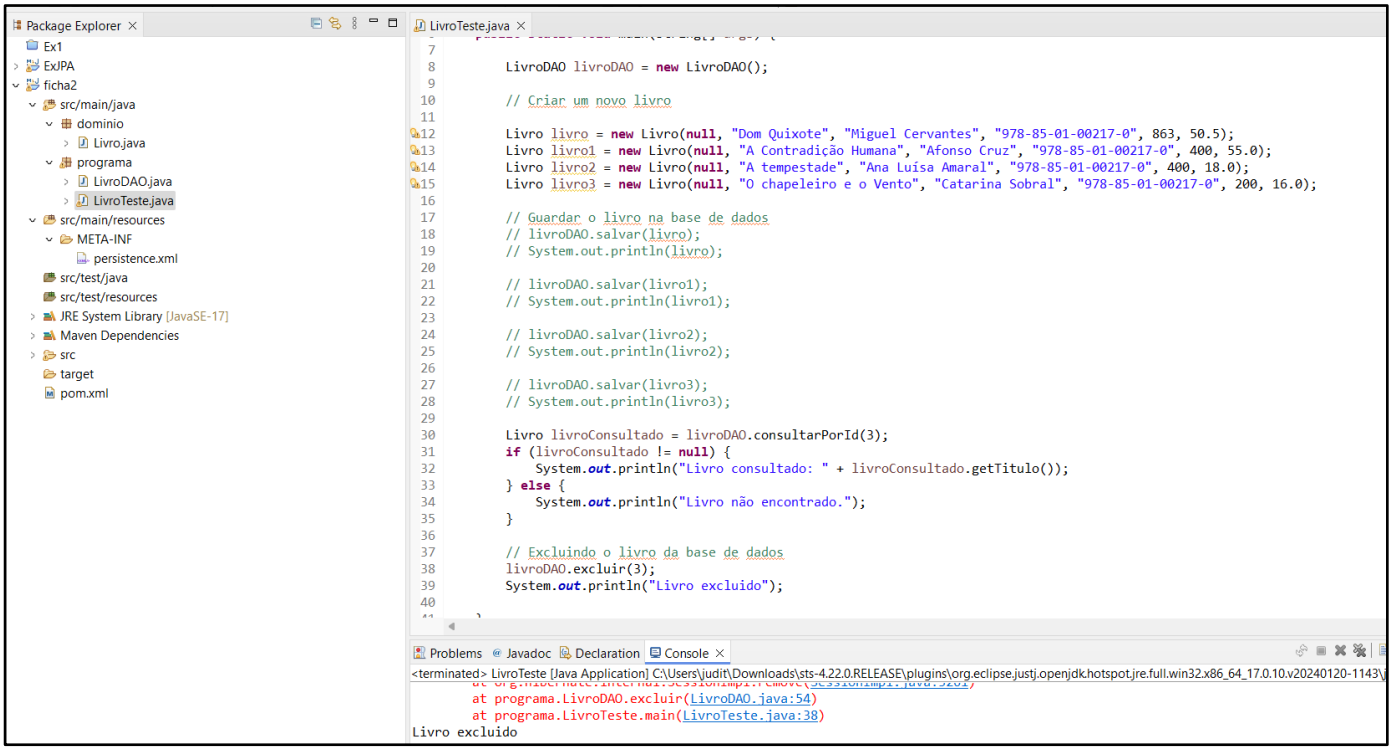
De seguida, abordámos o tema da persistência utilizando entidades JPA. Abordámos a persistência de dados, a sua importância, técnicas de persistência de dados, bases de dados persistentes, frameworks de acesso a bases de dados persistentes e persistência de dados com JPA. No Java Persistence API abordámos o diagrama sobre ORM - Object Relational Mapping (Mapeamento Objeto Relacional) e o seu funcionamento.

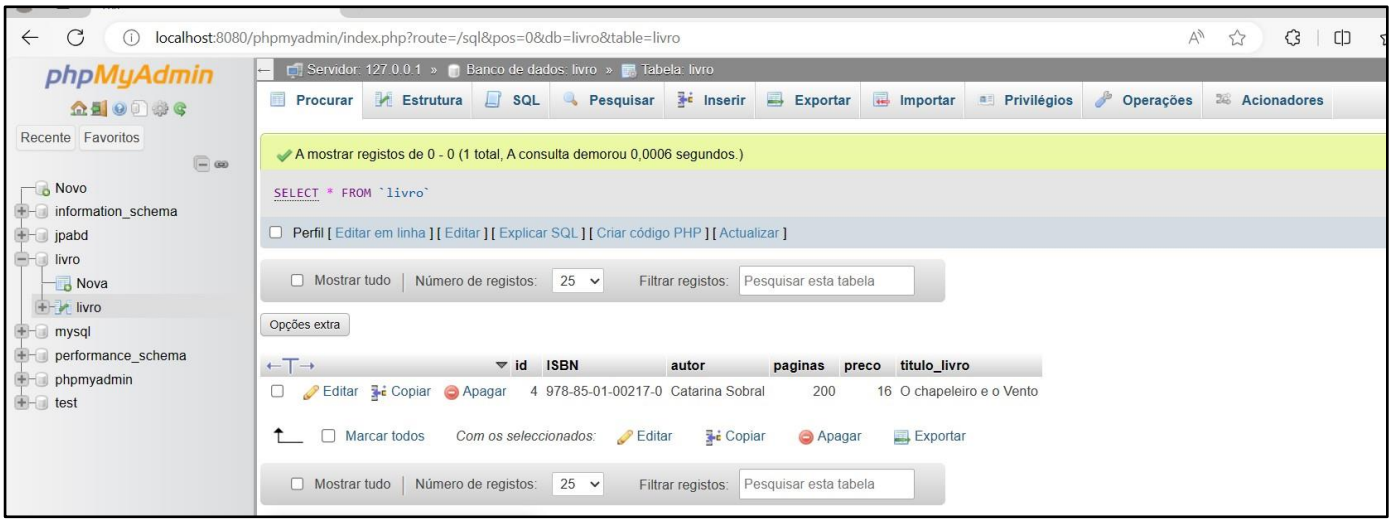
Após a abordagem dos conteúdos teóricos instalámos os programas necessários, o XAMPP, que é um pacote com os principais servidores de código aberto do mercado, incluindo FTP, base de dados MySQL e Apache com suporte as linguagens PHP e Perl e instalamos o Spring Tool Suite(STS) – que é a IDE mais popular e amplamente recomendada. O STS é uma versão personalizada do Eclipse.

Após a instalação dos programas foi introduzido o conceito de Maven. Apache Maven e Gradle são duas das ferramentas de automação de compilação mais populares no ecossistema Java. Maven é um gestor de projetos e de compilação de software que foi criado pela Apache Software Foundation. É muito usado por muitas empresas e projetos open source devido à sua simplicidade e à sua vasta quantidade de plugins disponíveis.

Começámos por criar um projeto Maven com uma aplicação simples instanciando três pessoas e mostrando os seus dados no ecrã, criou-se a classe, o construtor vazio, o construtor com os parâmetros, criação dos métodos get e set, e criação de método para guardar uma pessoa numa string. De seguida e com o xampp a correr no localhost no PhpMyAdmin criámos uma base de dados MySQL vazia. Depois editámos o arquivo pom.xml e atualizámos o Maven para a versão atual Java, forçámos o Update e incluímos as dependências Maven a serem transferidas. Configurámos o JPA no projeto por meio do arquivo persistence.xml (arquivo de configuração) e posteriormente incluímos os mapeamentos na classe de domínio com o criar/instanciar da classe EntityManager que faz a concessão e o acesso a dados e EntityManagerFactory para criar o EntityManager, assim já há ligação à base de dados e todos os arquivos e

persistência estão implementados. De seguida criamos interação com a base de dados, lemos, criámos, atualizámos, apagámos valores, entre outros. Depois consolidámos com um exercício prático individual.





De seguida abordámos o modelo lógico EJB (Enterprise Java Beans), que é um dos principais componentes da plataforma Java EE e que executa em um container EJB do lado do servidor. Ao utilizar EJB é possível construir aplicações distribuídas, totalmente seguras e com um eficaz tratamento de transações. Além disso, o EJB oferece serviços de persistência de dados, controle de concorrência, envio de mensagens, serviço de agendamento, chamadas a métodos remotos e a web services. EJB é, um conjunto de soluções integradas e centralizadas para suprir

os problemas de desenvolvimento de softwares. Os componentes EJB funcionam bem com a camada web. A JPA, ou Java Persistence API, é uma das APIs mais importantes que integram o framework e fornece ao EJB um mecanismo de persistência de dados poderoso. Além disso, o JCP (Java Community Process) mantém o EJB. Isso torna a API do EJB pública e garante um futuro promissor para a tecnologia. Demos a definição de container EJB, um container EJB pode ser considerado uma JVM poderosa que fornece uma variedade de serviços para aplicações EJB. Também existem servidores de software livre como JBoss e o GlassFish, eles estão disponíveis em containers EJB ou em containers web. A ação de colocar a aplicação EJB dentro de um container é chamada de implantação, ou deploy. Uma vez implantado e o servidor inicializado, os componentes passam a ter acesso aos diversos recursos oferecidos pelo container.

Demos a definição de componentes EJB, os componentes EJB são classes Java onde são desenvolvidas as lógicas de negócio da aplicação. No EJB, existem três tipos de componentes: Beans de Sessão; MDB (Message-Driven Beans); Entity. Os beans de sessão e os MDBs são responsáveis por conter as lógicas de negócio corporativas, enquanto as entities possuem toda a lógica de persistência. As anotações são uma forma de dizer para o container para fazer alguma coisa. Sempre que uma classe, interface, método ou atributo possuir alguma anotação, o container irá executar uma série de instruções de forma transparente para atender o seu propósito, lembrando que toda anotação é precedida de '@'.

Os beans de sessão são os componentes mais utilizados no EJB, pois são neles que a maioria da lógica corporativa é desenvolvida. Os beans de sessão podem ser acedidos de duas formas: Local; Remota; Utilizando as anotações @Local ou @Remote, respetivamente. Até a versão 3 do EJB, todos os beans de sessão precisavam de uma interface (Local ou Remota) para fazerem o acesso ao componente. A partir da versão 3.1, quando o bean for local, não é mais obrigatório existir uma interface. Nem mesmo a anotação @Local precisa ser colocada de forma explícita, e o acesso pode ser feito diretamente pela classe. Lembrando que, quando o acesso for Remoto, a obrigatoriedade de uma interface continua. Para criar componentes, não é necessário instanciar nenhum componente, este trabalho pertence ao container EJB, que cria algumas instâncias de forma transparente e as deixa disponíveis no pool de beans. É necessário apenas injetar a instância no código, com a anotação @EJB:

```
@EJB
Calculadora calculadora;
```

Os beans de sessão do tipo Stateless são responsáveis por conter operações que não necessitam durar mais do que uma chamada. As instâncias ficam em um pool de beans e são invocadas sempre que há uma solicitação. Somente após voltar para o pool, é que ela pode ser usada em outra solicitação.

Estas são algumas das funcionalidades que se encaixam nas características do bean de sessão Stateless, pois não precisam durar mais do que uma única chamada.

Os beans de sessão do tipo Stateful são responsáveis por conter operações que necessitam durar mais do que uma chamada, ou seja, que após a execução do componente o estado dos objetos modificados seja mantido.

O EJB dispõe de um recurso que permite que o programador libere a instância do componente ligado ao utilizador, quando esta não for mais utilizada, basta colocar a anotação @Remove em um método, quando este método for invocado e concluído, a instância em questão será excluída pelo container EJB.

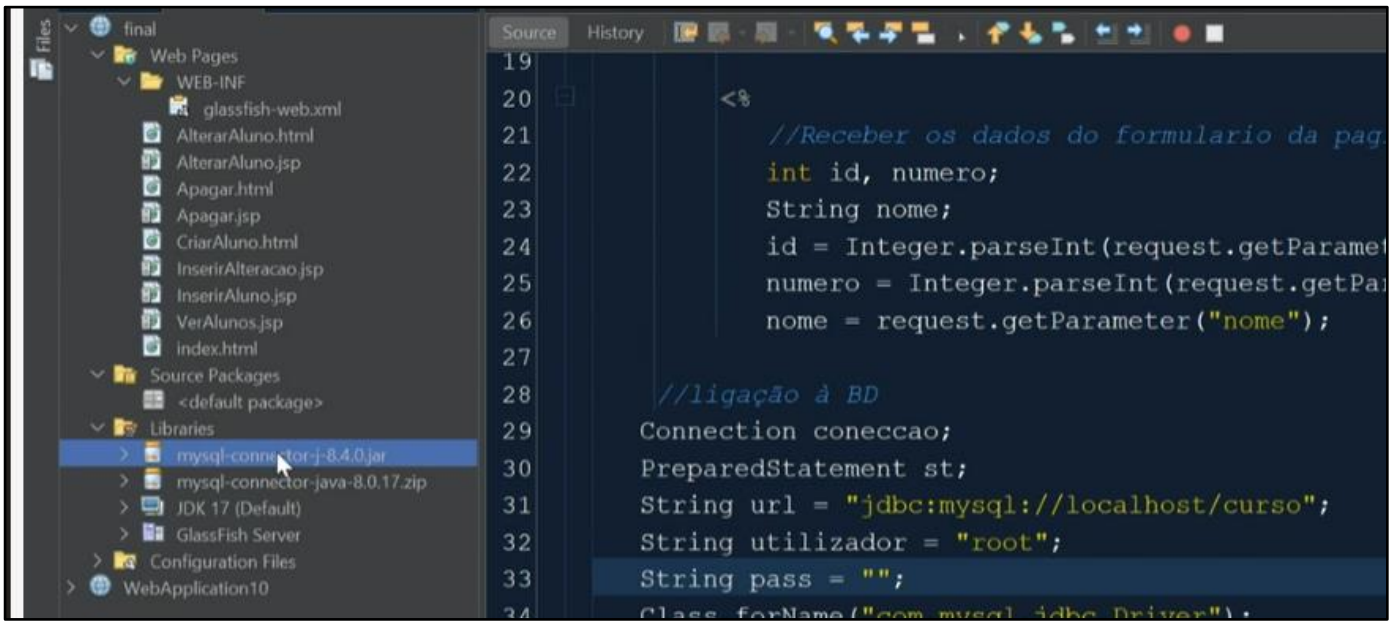
```java
import javax.ejb.Local;
import javax.ejb.Stateful;
import javax.ejb.Remove;

@Stateful
@Local
public class ItensBean {

  public void addItensNoCarrinho(String item) {
    //Lógica para adicionar itens no carrinho
  }

   @Remove
   public void finalizarCompra(){
     //Após a execução deste método, a instancia será destruída pelo container
  }
}
```

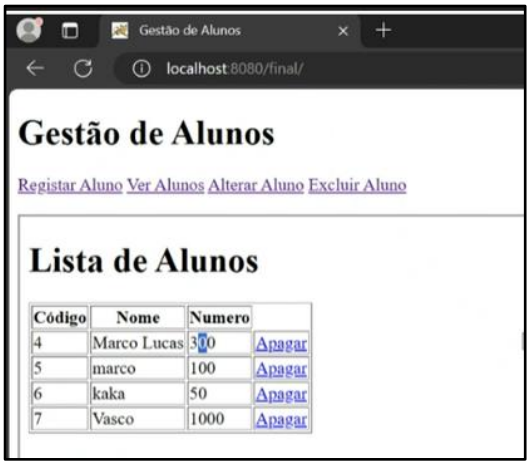
O tipo Singleton é criado apenas uma vez e disponível para todos os utilizadores do container EJB.

Os métodos callback do EJB são invocados pelo container em situações específicas durante o ciclo de vida de um componente. Os métodos callback são: @PostConstruct; @PreDestroy; @PrePassivate; @PostActivate.

Por fim, abordámos as aplicações WEB com Servlets, JSP´s e JSF.

O formador fez um exercício prático no NetBeans IDE19 onde aplicou no seu todo o conteúdo abordado na UFCD incorporando html, JSP (inclusão do JAVA), ligações à base de dados e efetuando as operações de registo, visualização, alteração de eliminação de dados.





Todos os conteúdos teóricos foram acompanhados por prática nas sessões e fichas de trabalho feitas pelos formandos em aulas assíncronas para consolidar conhecimentos. Alguns trabalhos práticos foram apresentados nas aulas, o que permitiu esclarecer dúvidas tanto dos nossos trabalhos como questões de trabalhos de colegas.

No final fizemos um trabalho prático onde aplicamos os conhecimentos adquiridos ao longo da UFCD.

Esta UFCD teve bastante componente teórica e bastante componente prática. Aprendi bastante, tanto na parte de novos conteúdos como na parte da consolidação de conteúdos que já conhecia. Esta UFCD foi complexa e abrangente pois exigia conhecimentos de conteúdos de muitas áreas.

O formador estruturou muito bem a UFCD, aprendi bastante. Bem-haja ao formador pela transmissão de conhecimentos e pela sempre disponibilidade para esclarecer dúvidas.