Primeira etapa: “Camadas de persistência e controle”

Configurar a conexão com o SQL Server no NetBeans:

1. Na aba "Serviços", divida "Banco de Dados", clique com o botão direito em "Drivers" e escolha "Novo Driver".
2. Na janela que abrir, clique em "Adicionar", selecione o arquivo "mssql-jdbc-12.2.0.jre8.jar" e confirme com "OK".
3. Preencha os campos "Database", "User" e "Password" com os valores correspondentes do seu banco de dados SQL Server.
4. No campo "JDBC URL", utilize o seguinte formato:

```
jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;
```

5. Clique em "Testar Conexão" e, se tudo estiver correto, finalize a configuração.

Criar os projetos do tipo Ant..Java Enterprise..Enterprise Application:

1. No NetBeans, vá para "Arquivo" -> "Novo Projeto...".
2. Selecione "Java Enterprise" na categoria e "Aplicativo Corporativo" como tipo de projeto. Clique em "Próximo".
3. Dê um nome ao seu projeto e selecione o local onde deseja salvá-lo. Clique em "Próximo".
4. Escolha "Ant" como Construtor do Projeto e clique em "Finalizar".

Implementar os projetos CadastroEE-ejb e CadastroEE-war:

1. Para o projeto CadastroEE-ejb:
   - Crie as entidades JPA (PessoaFisica e PessoaJuridica) com suas respectivas anotações de mapeamento.
   - Crie as classes de sessão EJB para cada entidade, onde você implementará as regras de negócio.

- Configure a conexão com o banco de dados usando a anotação

`@DataSourceDefinition`.

- Implemente os métodos para interagir com o banco de dados, utilizando JPA para persistência.

2. Para o projeto CadastroEE-war:

- Crie os servlets e JSPs para lidar com as requisições e respostas da interface web.

- Utilize o Bootstrap para melhorar o design da interface.

- Implemente as chamadas aos EJBs para acessar as regras de negócio.

- Implemente a lógica para exibir e inserir dados do banco de dados nas páginas JSP.

Certifique-se de configurar corretamente as dependências entre os projetos e de que o arquivo EAR está configurado para incluir os EJBs e o módulo WAR. Uma vez que tudo esteja configurado e implementado corretamente, você terá um aplicativo corporativo Java EE funcionando com persistência JPA, regras de negócio EJB e uma interface web usando Servlets, JSPs e Bootstrap.

Criar as entidades JPA através de New Entity Classes from Database:

1. No NetBeans, clique com o botão direito no pacote `cadastroee.model`.

2. Escolha "Novo" -> "Entidade" -> "Entidades JPA a partir do Banco de Dados".

3. Selecione `jdbc/loja` como Data Source e marque todas as tabelas desejadas.

4. No próximo passo, defina o pacote como `cadastroee.model` e marque a opção para criar o arquivo `persistence.xml`.

b. Adicionar os componentes EJB ao projeto:

1. Clique com o botão direito no pacote `cadastroee.controller`.

2. Escolha "Novo" -> "Sessão" -> "Beans para Entidades JPA".

3. Selecione todas as entidades e marque a geração da interface local.

4. Defina o nome do pacote como `cadastroee.controller`.

c. Efetuar pequenos acertos no projeto para uso do Jakarta:

1. Adicione a biblioteca "Jakarta EE 8 API" ao projeto CadatroEE-ejb.

2. Modifique TODAS as importações de pacotes `javax` para `jakarta` em todos os arquivos do projeto CadastroEE-ejb.

3. Na entidade `Produto`, mude o tipo do atributo `precoVenda` para `Float` no lugar de `BigDecimal`.

4. Modifique o arquivo `persistence.xml` para o seguinte:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<persistence version="1.0"

xmlns="http://java.sun.com/xml/ns/persistence"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/ xml/ns/persistence/persistence_1_0.xsd">

<persistence-unit name="CadastroEE-ejbPU" transaction-type="JTA">

<jta-data-source>jdbc/loja</jta-data-source>

<exclude-unlisted-classes>false</exclude-unlisted-classes>

<properties/>

</persistence-unit>

</persistence>
```

Após realizar esses passos, o projeto CadastroEE-ejb estará configurado com as camadas de persistência e controle utilizando JPA e EJBs, conforme as especificações fornecidas. Certifique-se de revisar e testar o projeto para garantir que tudo funcione corretamente.

1. Criar o Servlet:

1. No NetBeans, clique com o botão direito no pacote `cadastroee.servlets`.

2. Escolha "Novo" -> "Servlet".

3. Defina o nome do Servlet como `ServletProduto` e o nome do pacote como `cadastroee.servlets`.

4. Marque a opção "Adicionar informações ao descritor de implantação".

2. Adicionar a referência para a interface do EJB:

No código do Servlet (`ServletProduto.java`), adicione a seguinte linha para fazer a injeção do EJB:

```java
import cadastroee.controller.ProdutoFacadeLocal; // Verifique o pacote correto

@EJB

ProdutoFacadeLocal facade;
```

3. Modificar a resposta do Servlet:

No método `doGet` ou `doPost` do Servlet, utilize o `facade` para recuperar os dados e apresentá-los na forma de lista HTML.

```java
import java.io.IOException;
import java.io.PrintWriter;
import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.List;
import cadastroee.model.Produto; // Verifique o pacote correto

@WebServlet(name = "ServletProduto", urlPatterns =
{"/ServletProduto"})
public class ServletProduto extends HttpServlet {

    @EJB
    ProdutoFacadeLocal facade;

    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Produtos</title>");
            out.println("</head>");
```

```
            out.println("<body>");
            out.println("<h1>Lista de Produtos</h1>");
            out.println("<ul>");
            List<Produto> produtos = facade.findAll();
            for (Produto produto : produtos) {
                out.println("<li>" + produto.getNome() + "</li>");
            }
            out.println("</ul>");
            out.println("</body>");
            out.println("</html>");
        }
    }
}
```

4. Efetuar novos acertos no projeto para uso do Jakarta:

1. Adicione a biblioteca "Jakarta EE Web 8 API" ao projeto CadatroEE-war.

Após realizar esses passos, o Servlet `ServletProduto` estará pronto para recuperar os dados dos produtos usando o EJB `ProdutoFacadeLocal` e apresentá-los em uma lista HTML. Certifique-se de revisar e testar o Servlet para garantir que tudo funcione corretamente.

Modificar as importações de pacotes:

1. No NetBeans, clique com o botão direito no projeto `CadastroEE-war`.
2. Escolha "Ferramentas" -> "Conversor de Pacote" -> "Convert to jakarta.*".
3. Selecione todas as opções de conversão e clique em "OK".

Modificar o arquivo web.xml:

Substitua o conteúdo atual do arquivo `web.xml` pelo seguinte:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/ javaee/web-app_4_0.xsd">

    <servlet>

        <servlet-name>ServletProduto</servlet-name>

        <servlet-class>cadastroee.servlets.ServletProduto</servlet-
class>

    </servlet>

    <servlet>

        <servlet-name>ServletProdutoFC</servlet-name>

        <servlet-class>cadastroee.servlets.ServletProdutoFC</servlet-
class>

    </servlet>

    <session-config>

        <session-timeout>30</session-timeout>

    </session-config>

</web-app>
```

Certifique-se de revisar e testar o projeto para garantir que todas as modificações foram feitas corretamente e que o projeto continua funcionando como esperado após as alterações.

Segunda etapa: Interface Cadastral com Servlet e JSPs

1. ServletProdutoFC.java:

```java
import cadastroee.controller.ProdutoFacadeLocal; // Verifique o pacote correto

import java.io.IOException;

import javax.ejb.EJB;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;



@WebServlet(name = "ServletProdutoFC", urlPatterns = {"/ServletProdutoFC"})

public class ServletProdutoFC extends HttpServlet {



    @EJB

    ProdutoFacadeLocal facade;



    protected void processRequest(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        String acao = request.getParameter("acao");

        String destino = "";



        if (acao != null) {

            switch (acao) {
```

```
                case "listar":
                    request.setAttribute("produtos",
facade.findAll());
                    destino = "ProdutoLista.jsp";
                    break;
                case "formIncluir":
                    destino = "ProdutoDados.jsp";
                    break;
                case "formAlterar":
                    // Capturar o id fornecido como parâmetro do
request
                    // Consultar a entidade via facade e adicionar
como atributo da requisição
                    // Redirecionar para o formulário de alteração
                    break;
                case "excluir":
                    // Capturar o id fornecido como parâmetro do
request
                    // Remover a entidade via facade
                    // Atualizar a lista de produtos
                    // Redirecionar para a página de lista
                    break;
                case "alterar":
                    // Capturar o id fornecido como parâmetro do
request
                    // Consultar a entidade via facade
                    // Preencher os demais campos com os valores
fornecidos no request
                    // Alterar os dados via facade
                    // Atualizar a lista de produtos
                    // Redirecionar para a página de lista
```

```
                    break;

                case "incluir":

                    // Instanciar uma entidade do tipo Produto

                    // Preencher os campos com os valores fornecidos
no request

                    // Inserir via facade

                    // Atualizar a lista de produtos

                    // Redirecionar para a página de lista

                    break;

            }

        }



        request.getRequestDispatcher(destino).forward(request,
response);

    }



    @Override

    protected void doGet(HttpServletRequest request,
HttpServletResponse response)

            throws ServletException, IOException {

        processRequest(request, response);

    }



    @Override

    protected void doPost(HttpServletRequest request,
HttpServletResponse response)

            throws ServletException, IOException {

        processRequest(request, response);

    }

}
```

```

### 2. ProdutoLista.jsp:

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Lista de Produtos</title>
    </head>
    <body>
        <h1>Lista de Produtos</h1>
        <table border="1">
            <tr>
                <th>ID</th>
                <th>Nome</th>
                <th>Quantidade</th>
                <th>Preço</th>
                <th>Ações</th>
            </tr>
            <%-- Recuperar a lista de produtos enviada pelo Servlet --
%>
            <%
                java.util.List<cadastroee.model.Produto> produtos =
(java.util.List<cadastroee.model.Produto>)
request.getAttribute("produtos");
                for (cadastroee.model.Produto produto : produtos) {
```
```

```
            %>

            <tr>

                <td><%= produto.getId() %></td>

                <td><%= produto.getNome() %></td>

                <td><%= produto.getQuantidade() %></td>

                <td><%= produto.getPreco() %></td>

                <td>

                    <a href="ServletProdutoFC?acao=formAlterar&id=<%=
produto.getId() %>">Alterar</a>

                    <a href="ServletProdutoFC?acao=excluir&id=<%=
produto.getId() %>">Excluir</a>

                </td>

            </tr>

            <% } %>

        </table>

        <a href="ServletProdutoFC?acao=formIncluir">Incluir Novo
Produto</a>

    </body>

</html>
```

Certifique-se de ajustar os pacotes e os métodos no Servlet conforme necessário para atender às especificações da sua aplicação. Esses códigos servem como ponto de partida e podem precisar de ajustes adicionais de acordo com a sua implementação.

Criação da página de cadastro “ProdutoDados.jsp”

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>
```

```
    <head>
        <meta charset="UTF-8">
        <title>Cadastro de Produto</title>
    </head>
    <body>
        <h1>Cadastro de Produto</h1>
        <form action="ServletProdutoFC" method="post">
            <%-- Recuperar a entidade enviada pelo Servlet --%>
            <% cadastroee.model.Produto produto =
(cadastroee.model.Produto) request.getAttribute("produto"); %>

            <%-- Definir a variável acao --%>
            <% String acao = (produto == null) ? "incluir" :
"alterar"; %>

            <%-- Incluir um campo hidden para envio do valor de acao -
-%>
            <input type="hidden" name="acao" value="<%= acao %>">

            <%-- Incluir um campo hidden para envio do id --%>
            <% if (acao.equals("alterar")) { %>
                <input type="hidden" name="id" value="<%=
produto.getId() %>">
            <% } %>

            <%-- Incluir os campos para nome, quantidade e preço de
venda --%>
            Nome: <input type="text" name="nome" value="<%= (produto
!= null) ? produto.getNome() : "" %>"><br>
            Quantidade: <input type="text" name="quantidade"
value="<%= (produto != null) ? produto.getQuantidade() : "" %>"><br>
```

```
            Preço de Venda: <input type="text" name="precoVenda"
value="<%= (produto != null) ? produto.getPrecoVenda() : "" %>"><br>



            <%-- Concluir o formulário com um botão de envio --%>

            <input type="submit" value="<%= (acao.equals("incluir")) ?
"Incluir" : "Alterar" %>">

        </form>

    </body>

</html>
```

Terceira etapa: Melhorando o Design da Interface

ProdutoLista.jsp:

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

    <head>

        <meta charset="UTF-8">

        <title>Lista de Produtos</title>

        <!-- Incluir Bootstrap via CDN -->

        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css">

    </head>

    <body>

        <div class="container">

            <h1>Lista de Produtos</h1>

            <table class="table table-striped">
```

```
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Nome</th>
                        <th>Quantidade</th>
                        <th>Preço</th>
                        <th>Ações</th>
                    </tr>
                </thead>
                <tbody>
                    <!-- Recuperar a lista de produtos enviada pelo
Servlet -->
                    <% java.util.List<cadastroee.model.Produto>
produtos = (java.util.List<cadastroee.model.Produto>)
request.getAttribute("produtos");
                    for (cadastroee.model.Produto produto : produtos)
{ %>
                    <tr>
                        <td><%= produto.getId() %></td>
                        <td><%= produto.getNome() %></td>
                        <td><%= produto.getQuantidade() %></td>
                        <td><%= produto.getPreco() %></td>
                        <td>
                            <a
href="ServletProdutoFC?acao=formAlterar&id=<%= produto.getId() %>"
class="btn btn-primary">Alterar</a>
                            <a
href="ServletProdutoFC?acao=excluir&id=<%= produto.getId() %>"
class="btn btn-danger">Excluir</a>
                        </td>
                    </tr>
                    <% } %>
```

```
                </tbody>

            </table>

            <a href="ServletProdutoFC?acao=formIncluir" class="btn
btn-success">Incluir Novo Produto</a>

        </div>

    </body>

</html>

```

ProdutoDados.jsp:

```jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

    <head>

        <meta charset="UTF-8">

        <title>Cadastro de Produto</title>

        <!-- Incluir Bootstrap via CDN -->

        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css">

    </head>

    <body>

        <div class="container">

            <h1>Cadastro de Produto</h1>

            <form action="ServletProdutoFC" method="post">

                <% cadastroee.model.Produto produto =
(cadastroee.model.Produto) request.getAttribute("produto"); %>

                <% String acao = (produto == null) ? "incluir" :
"alterar"; %>

                <input type="hidden" name="acao" value="<%= acao %>">
```

```
                <% if (acao.equals("alterar")) { %>

                    <input type="hidden" name="id" value="<%=
produto.getId() %>">

                <% } %>

                <div class="mb-3">

                    <label for="nome" class="form-label">Nome:</label>

                    <input type="text" name="nome" id="nome"
class="form-control" value="<%= (produto != null) ? produto.getNome()
: "" %>">

                </div>

                <div class="mb-3">

                    <label for="quantidade" class="form-
label">Quantidade:</label>

                    <input type="text" name="quantidade"
id="quantidade" class="form-control" value="<%= (produto != null) ?
produto.getQuantidade() : "" %>">

                </div>

                <div class="mb-3">

                    <label for="precoVenda" class="form-label">Preço
de Venda:</label>

                    <input type="text" name="precoVenda"
id="precoVenda" class="form-control" value="<%= (produto != null) ?
produto.getPrecoVenda() : "" %>">

                </div>

                <button type="submit" class="btn btn-primary"><%=
(acao.equals("incluir")) ? "Incluir" : "Alterar" %></button>

            </form>

        </div>

    </body>

</html>
```

Com essas modificações, o Bootstrap será incluído via CDN nos seus arquivos JSP `ProdutoLista.jsp` e `ProdutoDados.jsp`, permitindo a utilização dos estilos e componentes do Bootstrap em sua aplicação. Certifique-se de testar para garantir que os estilos e comportamentos do Bootstrap estão funcionando conforme o esperado.

Responda de forma sucinta os seguintes questionamentos:

1. Qual a importância dos componentes de middleware, como o JDBC?
2. Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?
3. Como o padrão DAO melhora a manutenibilidade do software?
4. Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?
5. Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?
6. Como o uso de operador lambda simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?
7. Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static?
8. Como é organizado um projeto corporativo no NetBeans?
9. Qual o papel das tecnologias JPA e EJB na construção de um aplicativo para a plataforma Web no ambiente Java?
10. Como o NetBeans viabiliza a melhoria de produtividade ao lidar com as tecnologias JPA e EJB?
11. O que são Servlets, e como o NetBeans oferece suporte à construção desse tipo de componentes em um projeto Web?
12. Como é feita a comunicação entre os Serlvets e os Session Beans do pool de EJBs?
13. Como funciona o padrão Front Controller, e como ele é implementado em um aplicativo Web Java, na arquitetura MVC?
14. Quais as diferenças e semelhanças entre Servlets e JSPs?
15. Qual a diferença entre um redirecionamento simples e o uso do método forward, a partir do RequestDispatcher? Para que servem parâmetros e atributos nos objetos HttpRequest?
16. Como o framework Bootstrap é utilizado?
17. Por que o Bootstrap garante a independência estrutural do HTML?
18. Qual a relação entre o Boostrap e a responsividade da página?

1. Os componentes de middleware, como o JDBC, são importantes porque fornecem uma camada de abstração entre a aplicação e o banco de dados, permitindo o acesso e a manipulação dos dados de forma padronizada e independente do banco de dados específico.

2. A diferença fundamental entre o uso de Statement e PreparedStatement é que o PreparedStatement é pré-compilado, o que resulta em melhor desempenho e segurança, especialmente quando se trata de consultas SQL parametrizadas.

3. O padrão DAO (Data Access Object) melhora a manutenibilidade do software ao separar a lógica de acesso aos dados da lógica de negócios da aplicação. Isso permite que as alterações na camada de persistência sejam feitas de forma isolada, sem afetar outras partes do código.

4. Quando lidamos com um modelo estritamente relacional, a herança é refletida através de relações de tabelas, como tabelas filhas que têm uma chave estrangeira que referencia a chave primária da tabela pai. Isso é conhecido como mapeamento de herança para tabelas no contexto de mapeamento objeto-relacional.

5. A persistência em arquivo armazena os dados em arquivos no sistema de arquivos do computador, enquanto a persistência em banco de dados armazena os dados em um sistema de gerenciamento de banco de dados, oferecendo recursos avançados como consultas SQL, indexação e transações.

6. O uso de operador lambda simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java, ao permitir que você especifique a lógica a ser executada de forma concisa e inline, reduzindo a necessidade de código boilerplate.

7. Métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static porque o método main é estático e só pode chamar outros métodos estáticos diretamente. Isso ocorre porque ele é chamado antes que qualquer objeto da classe seja criado.

8. Em um projeto corporativo no NetBeans, os diferentes componentes da aplicação, como o front-end, back-end e camada de persistência, são geralmente organizados em diferentes projetos dentro de um projeto principal, conhecido como um projeto de tipo Enterprise Application (EAR).

9. As tecnologias JPA (Java Persistence API) e EJB (Enterprise JavaBeans) desempenham papéis fundamentais na construção de um aplicativo para a plataforma Web no ambiente Java. O JPA é responsável pela camada de persistência, fornecendo uma maneira padrão de mapear objetos Java para tabelas de banco de dados. O EJB fornece serviços de negócios e gerenciamento de transações para aplicativos corporativos Java.

10. O NetBeans viabiliza a melhoria da produtividade ao lidar com as tecnologias JPA e EJB fornecendo suporte integrado para essas tecnologias. Ele oferece assistentes e ferramentas visuais para criar entidades JPA, EJBs de sessão e entidades de bean gerenciado. Além disso, o NetBeans fornece integração com servidores de aplicativos, facilitando o desenvolvimento, teste e depuração de aplicativos que utilizam essas tecnologias.

11. Servlets são componentes Java que são usados para estender a funcionalidade de servidores web. Eles recebem solicitações HTTP e geram respostas dinâmicas com base nessa solicitação. O NetBeans oferece suporte à construção de Servlets em projetos web

Java, fornecendo modelos de Servlets, assistentes para criação de Servlets e integração com o contêiner web para execução e depuração de Servlets.

12. A comunicação entre Servlets e Session Beans do pool de EJBs é feita usando injeção de dependência. Os Servlets podem injetar referências para Session Beans usando a anotação `@EJB`, permitindo que eles acessem os métodos e dados dos Session Beans como objetos locais.

13. O padrão Front Controller é um padrão de projeto usado para centralizar o fluxo de controle em um aplicativo web. Ele é implementado em um aplicativo web Java na arquitetura MVC (Model-View-Controller) colocando um Servlet como controlador frontal. O Servlet recebe todas as solicitações do cliente, determina qual ação deve ser tomada com base na solicitação e, em seguida, despacha essa ação para o componente apropriado para execução.

14. Servlets são componentes Java que recebem e respondem a solicitações HTTP. Eles geralmente são usados para processar solicitações do cliente, executar lógica de negócios e gerar respostas dinâmicas. JSPs (JavaServer Pages) são arquivos HTML com marcações JSP que são usados para criar a interface do usuário de um aplicativo web. Eles geralmente são usados para apresentar dados e interagir com o usuário. A diferença principal entre Servlets e JSPs é que os Servlets são componentes Java que geram conteúdo dinâmico diretamente, enquanto os JSPs são arquivos HTML com inclusões de código Java para gerar conteúdo dinâmico.

15. Um redirecionamento simples (sendRedirect) direciona o navegador do cliente para uma nova URL, enquanto o método forward encaminha a solicitação internamente para outro componente do aplicativo no servidor. Os parâmetros e atributos nos objetos HttpRequest são usados para passar dados entre componentes, como Servlets e JSPs, ou para fornecer informações sobre a solicitação HTTP. O framework Bootstrap é utilizado para criar interfaces de usuário responsivas e estilizadas em aplicativos web. Ele garante a independência estrutural do HTML fornecendo uma biblioteca de estilos CSS e componentes JavaScript que podem ser facilmente aplicados a qualquer HTML. O Bootstrap é projetado para ser responsivo, o que significa que ele se adapta automaticamente a diferentes tamanhos de tela e dispositivos, garantindo uma experiência consistente em dispositivos móveis e desktops.