

# Universidade da Beira Interior

## Departamento de Informática



Departamento de  
Informática

### ***Bakery App***

Elaborado por:

**António Miguel Massano Morais**

Orientador:

**Professora Paula Prata**

8 de julho de 2024



# ***Agradecimentos***

Este trabalho foi a aplicação de todos os conhecimentos adquiridos ao longo da minha jornada acadêmica, e não seria possível sem contar com a ajuda de todas as pessoas que me acompanharam neste caminho.

Em primeiro lugar quero agradecer á minha namorada, aos meus pais, irmãos e família, por todos os sacrifícios, paciência, apoio e incentivo para a conclusão da licenciatura, com um especial agradecimento ao meu irmão José Morais por toda ajuda e conversas que permitiram o melhor desenvolvimento deste trabalho.

Dedico a conclusão deste trabalho ao meu pai, que apesar de já não estar entre nós, sempre me apoiou e incentivou a concluir a licenciatura, e sem as palavras e ensinamentos dele não estaria aqui.

Quero também agradecer a Professora Paula Prata por toda a ajuda e pela a orientação para conseguir desenvolver a aplicação.



# Conteúdo

<b>Conteúdo</b>	<b>iii</b>
<b>Lista de Figuras</b>	<b>v</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Enquadramento . . . . .	1
1.2 Motivação . . . . .	1
1.3 Objetivos . . . . .	2
1.4 Organização do Documento . . . . .	2
<b>2 Engenharia de Software</b>	<b>3</b>
2.1 Introdução . . . . .	3
2.2 Requisitos Funcionais . . . . .	3
2.2.1 Menu Inicial . . . . .	3
2.2.2 Produtos . . . . .	4
2.2.3 Rotas e Pontos de Entrega . . . . .	4
2.2.4 Encomendas . . . . .	5
2.2.5 Outros Requisitos . . . . .	5
2.3 Requisitos Não Funcionais . . . . .	5
2.4 Diagramas de Caso de Uso . . . . .	6
2.4.1 Diagrama do Menu Produto . . . . .	6
2.4.2 Diagrama do Menu Rota e Pontos de Entrega . . . . .	6
2.4.3 Diagrama do Menu Encomenda . . . . .	7
2.5 Conclusões . . . . .	8
<b>3 Tecnologias e Ferramentas Utilizadas</b>	<b>9</b>
3.1 Introdução . . . . .	9
3.2 Visual Studio Code . . . . .	9
3.3 HyperText Markup Language (HTML) . . . . .	10
3.4 Cascading Style Sheets (CSS) . . . . .	10
3.5 Hypertext Preprocessor (PHP) . . . . .	10
3.6 XAMPP . . . . .	10
3.7 Conclusões . . . . .	11

---

<b>4</b>	<b>Implementação e Testes</b>	<b>13</b>
4.1	Introdução . . . . .	13
4.2	Criação da Base de Dados . . . . .	13
4.3	Implementação do <i>Login</i> . . . . .	15
4.4	Implementação de um <i>Header</i> e <i>Footer</i> . . . . .	15
4.5	Implementação do ficheiro config . . . . .	17
4.6	Implementação da Página Inicial . . . . .	17
4.7	Implementação das operações . . . . .	19
4.7.1	Implementação do <i>add_produto</i> . . . . .	19
4.7.2	Implementação do <i>display_produto</i> . . . . .	22
4.7.3	Implementação do <i>modify_produto</i> . . . . .	24
4.7.4	Implementação do <i>update_produto</i> . . . . .	28
4.7.5	Implementação do <i>delete_produto</i> . . . . .	29
4.7.6	Implementação do <i>select_pe</i> . . . . .	29
4.7.7	Implementação do <i>display_pe</i> . . . . .	31
4.8	Conclusões . . . . .	34
<b>5</b>	<b>Conclusões e Trabalho Futuro</b>	<b>35</b>
5.1	Conclusões Principais . . . . .	35
5.2	Problemas Encontrados . . . . .	35
5.3	Trabalho Futuro . . . . .	36
	<b>Bibliografia</b>	<b>37</b>

## ***Lista de Figuras***

2.1	Diagrama Menu Produto . . . . .	6
2.2	Diagrama Menu Rotas e Pontos de Entrega . . . . .	7
2.3	Diagrama Menu Encomenda . . . . .	7
4.1	Diagrama entidade-relação . . . . .	14
4.2	Menu <i>Login</i> . . . . .	15
4.3	Página inicial . . . . .	18
4.4	Formulário para adicionar um produto . . . . .	21
4.5	Tabela de produtos . . . . .	24
4.6	Formulário pré-preenchido para atualizar produto . . . . .	24
4.7	Tabela para selecionar o ponto de entrega . . . . .	30
4.8	<i>display_rotas</i> . . . . .	34
4.9	<i>display_pe</i> . . . . .	34





# ***Acrónimos***

**HTML** HyperText Markup Language

**UBI** Universidade da Beira Interior

**IDE** Integrated Development Environment

**PHP** Hypertext Preprocessor

**CSS** Cascading Style Sheets



## **Capítulo**

# 1

## **Introdução**

### **1.1 Enquadramento**

Este relatório descreve o trabalho realizado no âmbito da unidade curricular Projeto integrado no curso de Engenharia Informática da Universidade da Beira Interior (UBI). O projeto consiste no desenvolvimento de uma aplicação web que permite a gestão de vendas numa padaria.

### **1.2 Motivação**

A ideia e o desenvolvimento da aplicação veio da minha experiência pessoal como funcionário de padaria.

Atualmente trabalho como auxiliar de produção mas já exerci o trabalho de distribuidor/vendedor ambulante. Estes cargos foram exercidos em duas padarias diferente mas que apresentavam a mesma dificuldade, saber de uma forma mais rápida qual a quantidade de pão vendida num determinado dia e qual a quantidade total de pão necessária para satisfazer as encomendas recebidas, facilitando as contagens do mesmo permitindo assim otimizar os consumos dos materiais necessários para o fabrico evitando desperdícios.

Outro motivo para a realização deste projeto foi o custo de uma aplicação deste género. Comprar uma aplicação destas para uma padaria de nível industrial compensa bastante devido ao grande volume de trabalho existente, mas para uma padaria mais pequena como um negocio familiar não é rentável comprar uma aplicação destas devido ao elevado custo. Daí a ideia de criar uma aplicação mais simples minimizando assim o custo para um negócio mais pequeno e que possa cumprir com as funcionalidades pretendidas.

### 1.3 Objetivos

Como já foi mencionado anteriormente, o objetivo principal deste projeto foi o desenvolvimento de uma aplicação web que permitisse uma melhor gestão das vendas e das encomendas.

- A Primeira Tarefa ,**T1**, consistiu em identificar todas as dificuldades do processo das encomendas e contagens.
- A Segunda Tarefa ,**T2**, após a identificação dos problemas, foi necessário pensar como ultrapassar estes problemas da forma mais simples, lógica e mais fácil de implementar.
- A Terceira Tarefa ,**T3**, foi a projeção da *Web App* de maneira a ser simples e de fácil uso. Foi necessário pensar em aspetos como a organização dos conteúdos e uma navegação intuitiva para melhorar a experiência do utilizador.
- A Quarta Tarefa ,**T4**, foi o processo de implementação através da análise de outras aplicações do estilo pretendido.

### 1.4 Organização do Documento

Este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.
2. O segundo capítulo – **Engenharia de Software** – apresenta todas as funcionalidades a serem implementadas para o correto desenvolvimento da aplicação.
3. O terceiro capítulo – **Tecnologias e Ferramentas Utilizadas** – descreve os conceitos mais importantes no âmbito deste projeto, bem como as tecnologias utilizadas durante do desenvolvimento da aplicação.
4. O quarto capítulo – **Implementação** – apresenta o processo de implementação, detalhes de implementação e a sua descrição, de forma a possibilitar a avaliação subjetiva como a sua análise;
5. O quinto capítulo – **Conclusões e Trabalho Futuro** – resume os principais resultados e limitações do projeto e apresenta algumas sugestões de melhoria que poderiam ser implementadas numa trabalho futuro.

## **Capítulo**

# 2

## **Engenharia de Software**

### **2.1 Introdução**

Neste capítulo é feita uma análise de como o projeto foi abordado face aos problemas em questão. Também é feito um levantamento de requisitos funcionais 2.2 e requisitos não funcionais 2.3 tendo como objetivo definir as funcionalidades presentes. É feita uma apresentação dos diagramas de casos de uso 2.4 da aplicação com o objetivo de representar o seu funcionamento. E por fim, será realizada uma breve conclusão do capítulo.

### **2.2 Requisitos Funcionais**

Requisitos Funcionais são os aspetos do planeamento do sistema que se preocupam com ações e/ou funcionalidades que o sistema desenvolvido deve executar de modo a garantir que todas as necessidades são atendidas. Estes ajudam os *developers* a projetar a lógica por trás das funcionalidades a serem desenvolvidas, para além de auxiliarem na esquematização da própria aplicação. Para este trabalho, foram definidos os seguintes requisitos funcionais de modo a elaborar a solução para o tema proposto:

#### **2.2.1 Menu Inicial**

A aplicação deve permitir ao utilizador:

- Aceder ao menu "Produtos";
- Aceder ao menu "Rotas e Pontos de Entrega";

- Aceder ao menu "Encomendas";

### **2.2.2 Produtos**

O menu Produtos permite administrar quais os artigos fabricados na padaria, e possíveis de encomendar. O utilizador ao ser encaminhado para este menu deve conseguir:

- Consultar os produtos existentes;
- Adicionar um novo produto;
- Modificar os campos de um produto;
- Eliminar um produto.

### **2.2.3 Rotas e Pontos de Entrega**

O menu Rotas e Pontos de Entrega serve para definir as rotas e localidades onde as encomendas podem ser entregues. O utilizador ao ser encaminhado para este menu deve conseguir:

- Consultar as rotas existentes;
- Adicionar uma nova rota;
- Modificar os campos de uma rota;
- Eliminar uma rota.

As rotas são constituídas por vários pontos de entrega, que também é possível serem definidos pela aplicação, pelo que deve apresentar as seguintes opções:

- Adicionar um ponto de entrega a uma rota;
- Consultar os pontos de entrega de uma determinada rota;
- Modificar os campos de um ponto de entrega;
- Eliminar um ponto de entrega.

### **2.2.4 Encomendas**

As encomendas são uma combinação entre os produtos e as rotas, na medida que o utilizador pode escolher a quantidade de um ou mais produtos para serem entregues num determinado ponto de entrega que está inserido numa rota. Para esse efeito a aplicação deve:

- Permitir ao utilizador escolher o ponto de entrega mais conveniente;
- Permitir escolher diversos produtos com a respetiva quantidade;
- Permitir consultar os detalhes da encomenda efetuada.

### **2.2.5 Outros Requisitos**

A aplicação deve saber gerir os tipos de utilizador (administrador, distribuidor, funcionário), visto que cada um tem privilégios diferentes.

Para tal pretende-se implementar sessões de utilizador, um recurso oferecido pelo Hypertext Preprocessor (PHP) que permite monitorizar os utilizadores que estão a utilizar as páginas da aplicação das quais têm acesso.

## **2.3 Requisitos Não Funcionais**

### **1. Intuitiva e Consistente:**

- A aplicação tem uma interface organizada, fácil de usar e de entender;
- Não necessita de instruções detalhadas;
- Existirem padrões de uso consistentes em toda a aplicação;

### **2. Responsiva:**

- A aplicação deve ser rápida a efetuar as operações pedidas;

### **3. Compatibilidade:**

- Deve permitir o uso em vários e diferentes dispositivos;
- Deve ser possível de usar em diferentes sistemas operativos;

### **4. Acessibilidade:**

- Pode ser utilizada por uma diversidade de utilizadores, ou seja, com níveis de habilidade diferentes;

5. Segura:

- Verificação dos dados introduzidos nos formulários para prevenir a injeção de código/*scripts* maliciosos que possam prejudicar a base de dados mantendo a sua integridade;

## 2.4 Diagramas de Caso de Uso

Nesta secção são apresentados os casos de uso que descrevem as funcionalidades dos principais menus do sistema.

### 2.4.1 Diagrama do Menu Produto

O diagrama de caso de uso do Menu Produto mostra que o utilizador pode adicionar, modificar, eliminar e consultar os produtos disponíveis.

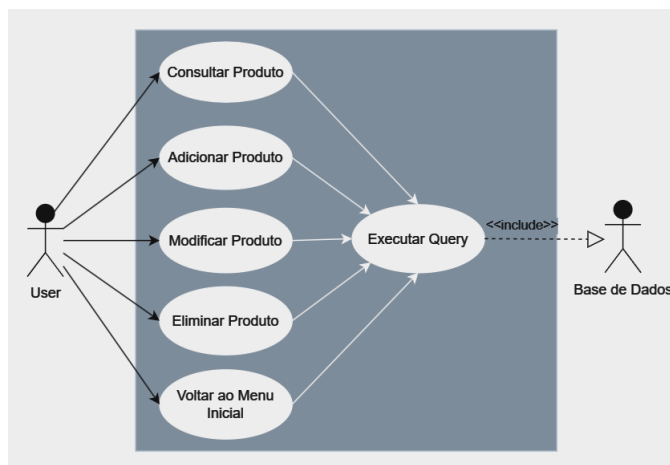


Figura 2.1: Diagrama Menu Produto

### 2.4.2 Diagrama do Menu Rota e Pontos de Entrega

O diagrama do Menu Rotas e pontos de entrega demonstra que é possível gerir as rotas e os pontos de entrega de cada uma delas.



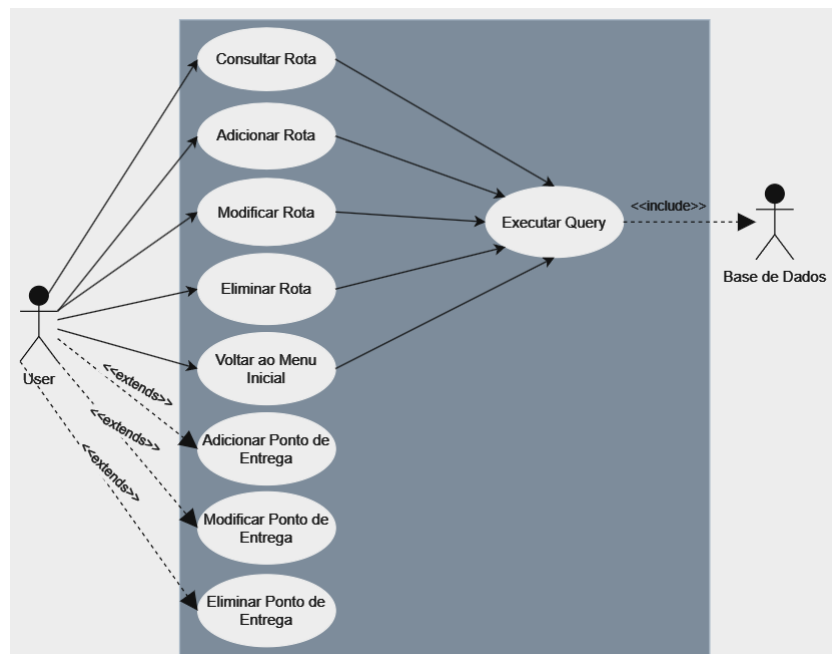


Figura 2.2: Diagrama Menu Rotas e Pontos de Entrega

### 2.4.3 Diagrama do Menu Encomenda

O diagrama do Menu Encomenda mostra que é possível adicionar uma encomenda bem como consultar os detalhes da mesma.

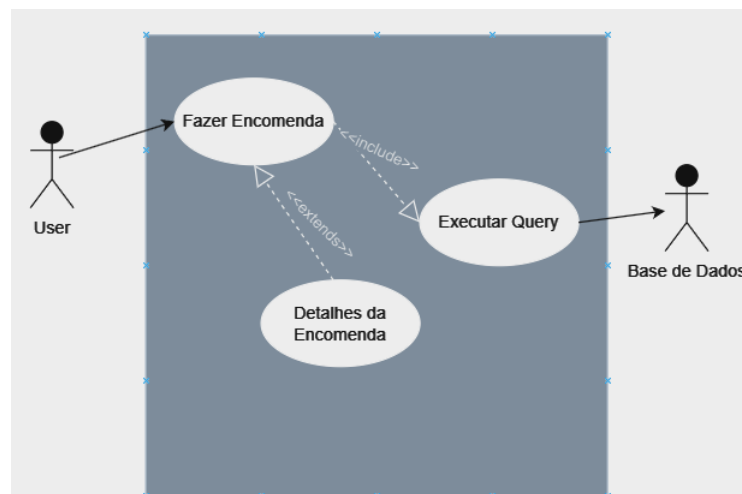


Figura 2.3: Diagrama Menu Encomenda

## **2.5 Conclusões**

Este capítulo tem como propósito definir todas as funcionalidades que foram implementadas na aplicação.

## **Capítulo**

# 3

## ***Tecnologias e Ferramentas Utilizadas***

### **3.1 Introdução**

Durante a elaboração deste projeto foi utilizado um conjunto de ferramentas e tecnologias, cada uma delas com o seu propósito e função a desempenhar na construção do website. Ao longo das seguintes subsecções serão apresentados estes instrumentos e a razão pela qual foram utilizados no âmbito deste projeto.

### **3.2 Visual Studio Code**

O *Visual Studio Code*[1] é um *Integrated Development Environment (IDE)* desenvolvido pela Microsoft. Esta ferramenta tem suporte para muitas linguagens de programação (incluindo as que foram utilizadas neste projeto), tem a possibilidade de instalar extensões que agilizam a escrita de código e desenvolvimento de software e tem suporte para softwares de controlo de versões de código. Este IDE foi escolhido tanto pela familiarização pessoal com a sua utilização, como pelas vantagens já descritas. *visual studio code* foi o ambiente de programação ideal, tem uma interface intuitiva onde foi possível importar a pasta do projeto e ter um acesso rápido e eficaz a todos os ficheiros e componentes.

### 3.3 HyperText Markup Language (HTML)

*HTML*[2] é uma linguagem de marcação utilizada para criar os elementos e definir a estrutura de páginas web. O navegador interpreta o código HTML e mostra ao utilizador a página web resultante. É uma das principais tecnologias usadas na criação de sites e aplicações web. Foi com o uso dos formulários do HTML que foi possível obter dados introduzidos pelo utilizador e enviar esses dados para o servidor onde são respetivamente tratados. A estrutura das páginas da área divididas em cabeçalho, corpo e rodapé foi também definida através do HTML.

### 3.4 Cascading Style Sheets (CSS)

HTML[3] é a linguagem de estilo usada para definir a forma como são apresentadas as páginas web. Permite especificar diversas propriedades de elementos HTML como cor, *layout*, tipo de fonte, tamanho, entre outras, bem como adaptar a apresentação da página ao tipo de dispositivo onde é exibida. Neste projeto, o CSS é utilizado para definir o estilo do *header* e do *footer*.

### 3.5 PHP

*PHP*[4] é uma linguagem de programação *open source* que permite adicionar uma lógica de programação a páginas web, e é geralmente utilizada em conjunto com outras tecnologias para criar sites e aplicações web dinâmicas. O PHP é executado no lado do servidor, o que significa que o código é processado antes de ser enviado para o navegador do utilizador.

### 3.6 XAMPP

O XAMPP[5] é um pacote de software livre e multiplataforma que fornece um ambiente de desenvolvimento web local. Ele inclui o Apache (servidor web), o MySQL, o PHP e o Perl, criando um servidor local que permite testar e desenvolver aplicações web em um ambiente semelhante ao de um servidor real. O XAMPP facilita a configuração e a execução de aplicações web no computador local, agilizando o processo de desenvolvimento o que serviu de grande utilidade.

## **3.7 Conclusões**

Neste capítulo foram analisados todos os tipos de ferramentas necessárias para atingir o objetivo desejado. Todas estas ferramentas e tecnologias foram escolhidas, de modo, a proporcionar a capacidade de desenvolver o tipo de aplicação web desejada.



## Capítulo

# 4

## ***Implementação e Testes***

### **4.1 Introdução**

Neste capítulo será apresentada a implementação e processo de realização das diversas funcionalidades presentes no projeto. De modo a fazer esta demonstração, serão apresentados pequenos excertos de código com uma breve descrição de forma a explicar a minha implementação.

### **4.2 Criação da Base de Dados**

Para a correta implementação da aplicação, começou por se criar uma base de dados denominada db\_bakery. Para isto, utilizou-se o phpMyAdmin que é uma das ferramentas que o Xampp 3.6 disponibiliza para uma melhor e mais fácil gestão de uma base de dados. Foi também necessária a criação de um utilizador para que se pudesse garantir que teria acesso a todos os privilégios que o phpMyAdmin oferece.

Após isto, foram definidas 5 tabelas para a nossa aplicação que são:

- Tabela produto - possui um identificador único, bem como um nome, a sua composição que define os ingredientes do mesmo, o seu preço e a taxa de iva praticada;
- Tabela rota - possui um identificador único, um nome e tem um distribuidor associado. Serve para agrupar pontos de entrega;
- Tabela ponto\_entrega - serve para guardar as informações acerca dos locais de entrega das encomendas;

- Tabela encomenda - guarda a informação relevante para uma encomenda, como o nome do cliente, o NIF, a data da encomenda e o local a ser entregue;
- Tabela linha\_ne - contém a informação da encomenda relativa aos produtos e respetivas quantidades;

O diagrama da figura 4.1 representa as relações entre as tabelas da base de dados:

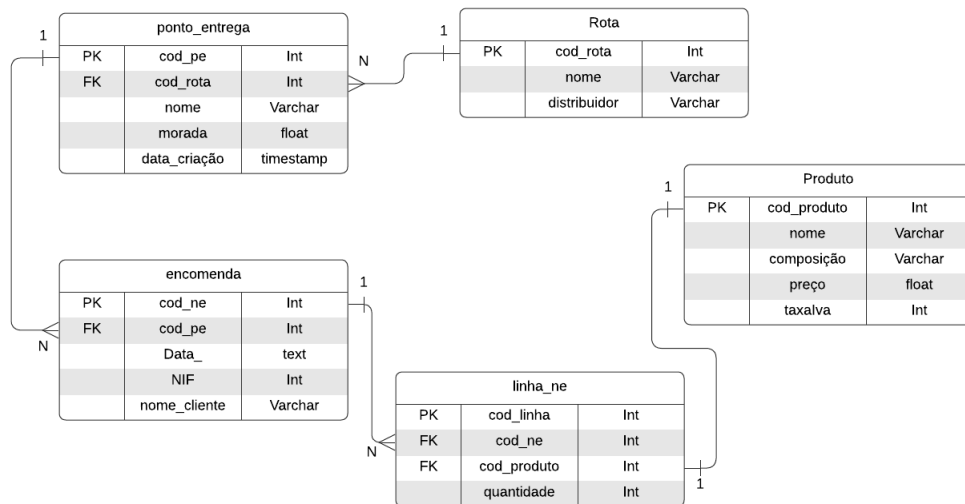


Figura 4.1: Diagrama entidade-relação



## 4.3 Implementação do *Login*

Na implementação do *login* foram definidos dois tipos de *login*, o *login* para administrador que redireciona para a página inicial que será descrita mais a frente. E o *login* para o distribuidor que recebe como *username* o valor do campo distribuidor presente na tabela rota e como *password* o código da rota que lhe pertence, redirecionando para uma tabela com as encomendas atribuídas aos pontos de entrega da mesma rota.

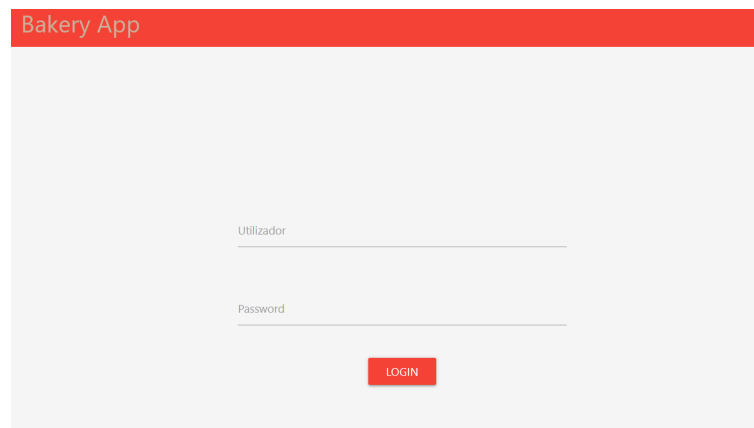


Figura 4.2: Menu *Login*

## 4.4 Implementação de um *Header* e *Footer*

Após a criação da base de dados passou-se para a implementação *web* propriamente dita. O primeiro passo foi criar um *header* e um *footer* para a nossa aplicação *web*.

```
<head>
  <title>Bakery App</title>
  <!-- Compiled and minified CSS -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
  <style type="text/css">
    .brand{
      background: #cbb09c !important;
    }
    .brand-text {
      color: #cbb09c !important;
    }
  }

  form{
```

```
        max-width: 460px;
        margin: 20px auto;
        padding: 20px;
    }
</style>

</head>
<body class="grey lighten-4">
    <nav class="red z-depth-0">
        <div class="container">
            <a href="index.php" class="brand-logo brand-text">Bakery
                App</a>
            <ul id="nav-mobile" class="right hide-on-small-and-down"
                >
                <!-- <li><a href="index.php" class="btn brand z-depth
                    -0">HOME</a></li>-->
            </ul>
        </div>
    </nav>
```

#### Excerto de Código 4.1: Implementação do Header

Este *header* é extremamente simples. Primeiramente definiu-se o título com a tag `<title>`. Em seguida importou-se o arquivo CSS do Materialize utilizando o link mencionado no excerto de código 4.1 para criar o design *front-end* da página. Depois disto, definimos o estilo interno com:

- **.brand** - define a cor de fundo como #cbb09c (um tom de bege);
- **.brand-text** - que define a cor do texto como #cbb09c;
- **form** - define o estilo dos formulários. Neste caso os formulários terão uma largura máxima de 460px, margens automáticas centralizadas e um preenchimento interno de 20px.

Por fim, falta definir o corpo da página. Comecei por definir a cor de fundo da página como cinza. Em seguida, foi criada uma barra de navegação com fundo vermelho. Defini um container para centralizar todo o conteúdo da barra de navegação, um link que funciona como logótipo da aplicação e que redireciona para a página inicial. Foi definido também um container para itens de navegação adicionais, que ficarão alinhados à direita. Neste caso, está comentado.

Para a do *footer*, referido no excerto de código 4.2, utilizei uma implementação muito simples em que foi definida uma secção do tipo *footer* que vai estar centralizado na página e a cor cinza como podemos ver no excerto de código seguinte:

```
<footer class="section">
  <div class="center grey-text">Bakery App 2024</div>
</footer>
</body>
```

Excerto de Código 4.2: "Implementação do Footer"

## 4.5 Implementação do ficheiro config

Como foi mencionado anteriormente, houve a necessidade da criação de um utilizador no phpMyAdmin com a finalidade de ter acesso a todos os privilégios que este oferece. Este utilizador vai ser também utilizado para fazer a ligação a base de dados no código desenvolvido, presente no excerto 4.3.

O ficheiro config foi desenvolvido para que cada vez que seja necessário ligar a base de dados não se tenha de definir a ligação do zero. Com este ficheiro basta fazer o *include* no ficheiro que quer fazer a ligação.

```
<?php
//Ligacao a Base de Dados pelo mySQLi
//Dados para ligacao:
//Host: localhost
//Username: tomorais
//Password: morais123
//Database: db_bakery
$ligacao = mysqli_connect('localhost', 'tomorais', 'morais123', '
    db_bakery');

//Verificar ligacao
if (!$ligacao) {
    echo 'Erro na ligacao:' . mysqli_connect_error();
}

?>
```

Excerto de Código 4.3: "Implementação do Config"

## 4.6 Implementação da Página Inicial

A pagina inicial foi desenhada para ser simples e direta como podemos ver na seguinte figura:

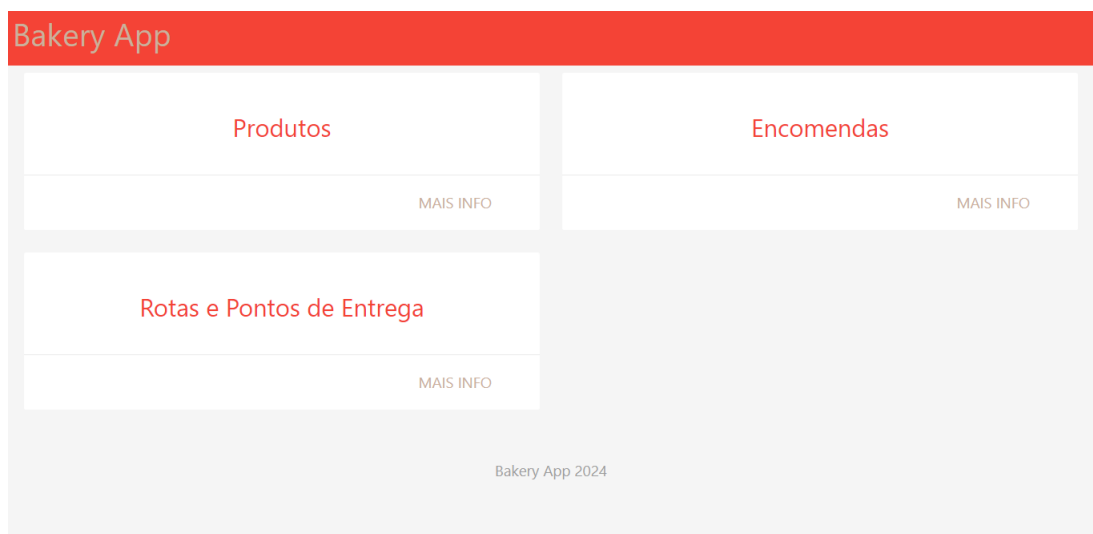


Figura 4.3: Página inicial

Para obter o resultado da figura 4.3 foi implementado um container centralizado que ajusta a largura automaticamente de acordo com o tamanho da tela.

Após isto foi definida uma linha para agrupar colunas, ou seja, a linha vai garantir que as colunas estejam corretamente alinhadas e em seguida foram definidos os tamanhos da coluna para ecrãs pequenos e médios.

Estando definidas as medidas das colunas, criei um *card* no qual centraliza o seu conteúdo, sendo este um título a cor vermelha. Ainda dentro do *card*, tem uma ação com o nome Mais Info que redireciona para as páginas *display* correspondente.

```
<!DOCTYPE html>
<html>
  <?php include('templates/header.php') ?>

  <!-- PRODUTOS -->
  <div class="container">
    <div class="row">
      <div class="col s6 md3"> <!-- BOX PRODUTOS -->
        <div class="card z-depth-0">
          <div class="card-content center">
            <h5 class="center red-text">Produtos</h5>
          </div>
          <div class="card-action right-align">
            <a href="display_produto.php" class="brand-text">
              Mais Info</a>
            </div>
        </div>
      </div>
    </div>
  </div>
```

```

        </div>
    </div>

    <div class="col s6 md3 "> <!-- BOX ENCOMENDAS -->
        <div class="card z-depth-0">
            <div class="card-content center">
                <h5 class="center red-text">Encomendas</h5>
            </div>
            <div class="card-action right-align">
                <a href="display_encomendas.php" class="brand-text">
                    Mais Info</a>
            </div>
        </div>
    </div>

    <div class="col s6 md3 "> <!-- BOX ROTAS -->
        <div class="card z-depth-0">
            <div class="card-content center">
                <h5 class="center red-text">Rotas e Pontos de
                    Entrega</h5>
            </div>
            <div class="card-action right-align">
                <a href="display_rota.php" class="brand-text">Mais
                    Info</a>
            </div>
        </div>
    </div>

    </div>
</div>

<?php include('templates/footer.php') ?>
</html>

```

Excerto de Código 4.4: "Implementação da Página Inicial"

## 4.7 Implementação das operações

Para efeitos futuros, irei demonstrar como foram implementadas as operações relativas a tabela produto, uma vez que todas as operações das outras tabelas funcionam de forma semelhante.

### 4.7.1 Implementação do *add\_produto*

Para adicionar um produto é exibido um formulário com os campos a serem preenchidos como podemos ver na 4.4

```
<!DOCTYPE html>
```

```

<html>
  <?php include('templates/header.php') ?>

  <section class="container grey-text">
    <h4 class="center">Adicionar Produto</h4>
    <form action="" class="white" action="add_produto.php" method="
      POST">
      <label>Nome:</label>
      <input type="text" name="nome" value="<?php echo
        htmlspecialchars($nome)?>">
      <div class="red-text"><?php echo $erros['nome']; ?></div>
      <label>Composição (Separar por virgula):</label>
      <input type="text" name="composicao" value="<?php echo
        htmlspecialchars($composicao)?>">
      <div class="red-text"><?php echo $erros['composicao']; ?></
        div>
      <label>Preço:</label>
      <input type="text" name="pre_o" value="<?php echo
        htmlspecialchars($pre_o)?>">
      <div class="red-text"><?php echo $erros['pre_o']; ?></div>
      <label>taxaIVA(%):</label>
      <input type="text" name="taxaIVA" value="<?php echo
        htmlspecialchars($taxaIVA)?>">
      <div class="red-text"><?php echo $erros['taxaIVA']; ?></div>
      <div class="center">
        <input type="submit" name="submit" value="Adicionar"
          class="btn brand z-depth-0">
        <a href="display_produto.php" class="btn brand z-depth-0"
          ">Voltar</a>
      </div>
    </form>
  </section>

  <?php include('templates/footer.php') ?>
</html>

```

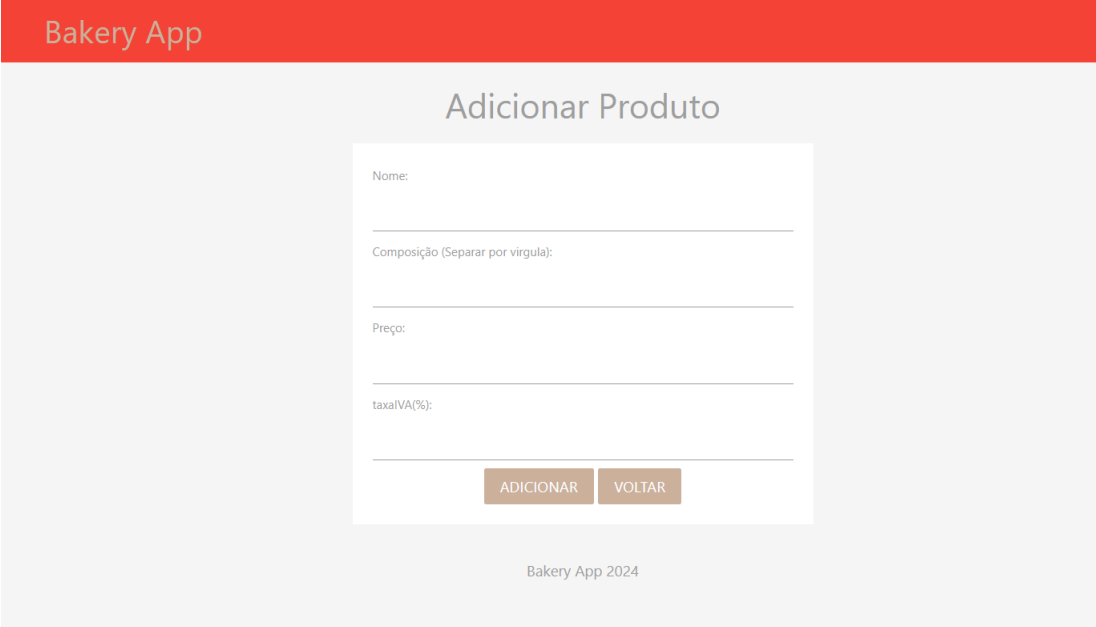
Excerto de Código 4.5: "Formulário para adicionar produto"

Após o preenchimento os dados introduzidos são verificados. Para fazer essa verificação foi utilizada a função *htmlspecialchars* para converter os dados para entidades HTML, ou seja, converte em strings seguras para caracteres especiais.

Essa verificação, começa por ver se o campo está vazio e caso esteja é emitida uma mensagem de erro no formulário dizendo que o campo é obrigatório. Caso tenha algum tipo de dados, fiz uso da função *preg\_match*, e com o auxílio de expressões regulares, defini qual o formato de dados aceites. No caso de um produto, está definido que:

- `\'/^[a-zA-Z\s]+\$/'` - esta expressão é referente ao campo nome e restringe os dados introduzidos a apenas conterem caracteres entre 'A' e 'Z', maiúsculas e minúsculas, e espaços.
- `/^[a-zA-Z0-9\s]+(,\s*[a-zA-Z0-9\s]*)*\$/` - esta expressão refere-se ao campo composição e aceita quaisquer *strings* compostas por letras, algarismos e espaços, separada por vírgulas
- `/^\d+(\.\d{1,2})?\$/` - esta expressão refere-se ao campo preço, que aceita a escrita de dois números separados com um '.'. O primeiro número pode ser qualquer algarismo, e o segundo apenas pode ser composto por números com o máximo de dois algarismos.
- `/^(100[1-9]?[0-9])\$/` - esta expressão diz respeito ao campo taxaIVA e restringe o *input* para números entre 0 e 100.

Caso o dados estejam todos corretos executa a função `mysqli_real_escape_string` que é utilizada para proteger a base de dados contra a injeção de dados maliciosos que podem potencialmente danificar a base de dados. Após a esta verificação vai ser executada a *query* definida para inserir o produto na tabela da base de dados.



The screenshot shows a web application interface for a bakery. At the top, there is a red header bar with the text "Bakery App" in white. Below this, the main content area has a light gray background. In the center, there is a white rectangular form titled "Adicionar Produto" in a medium gray font. The form contains four input fields, each with a label above it: "Nome:", "Composição (Separar por vírgula):", "Preço:", and "taxaIVA(%):". Below the input fields, there are two buttons: "ADICIONAR" and "VOLTAR", both in a light brown color with white text. At the bottom of the form, centered, is the text "Bakery App 2024" in a small gray font.

Figura 4.4: Formulário para adicionar um produto

### 4.7.2 Implementação do *display\_produto*

Após adicionar o primeiro produto a base de dados, apresentou-se a necessidade de verificar se os dados foram introduzidos corretamente sem ser necessário ir ao phpMyAdmin. Para isso implementei uma página para se visualizar os dados de uma dada tabela, neste caso da tabela produto.

Para isso defini uma tabela, que após a correta execução da *query*, recebe e mostra os valores que guardados na base de dados como se pode ver na figura 4.5. Na figura podemos ver ainda que são dadas três opções ao utilizador, a opção de adicionar, alterar ou eliminar um produto.

```
<!DOCTYPE html>
<html>

    <?php include( 'templates/header.php' ) ?>
    <?php
        include( 'config/ligacao_db.php' );

        //Query para obter todos os produtos
        $sql = 'SELECT * FROM produto';

        //Executar a query
        $resultado = mysqli_query($ligacao , $sql);

        //obter as linhas resultantes em forma de array
        $produtos = mysqli_fetch_all($resultado , MYSQLI_ASSOC);

        //Liberta o resultado da memoria
        mysqli_free_result($resultado);

    echo '<table border="10" cellpadding="2" cellspacing="10">
        <tr>
            <td class = center> <font face="Arial">Codigo de Produto</font> </td>
            <td class = center> <font face="Arial">Nome</font> </td>
            <td class = center> <font face="Arial">Composicao</font> </td>
            <td class = center> <font face="Arial">Preco</font> </td>
            <td class = center> <font face="Arial">TaxaIVA</font> </td>
            <td class = center> <font face="Arial">Criado Em</font> </td>
        </tr>';

        //Percorrer o array de produtos
        foreach($produtos as $produto){
            echo '<tr>
                <td class=center> <font face="Arial">'. $produto['
                    cod_produto' ].' </font> </td>
```



```

        <td class=center> <font face="Arial">' . $produto['nome'] . '
        </font> </td>
        <td class=center> <font face="Arial">' . $produto['composicao'] . '
        </font> </td>
        <td class=center> <font face="Arial">' . $produto['preço'] . '
        </font> </td>
        <td class=center> <font face="Arial">' . $produto['taxaIVA'] . '
        </font> </td>
        <td class=center> <font face="Arial">' . $produto['criado_em'] . '
        </font> </td>
        <td class=center> <form align="center" action="
        modify_produto.php" method="post">
        <input type="hidden" name="id" value="' . $produto["
        cod_produto"] . '">
        <input type="submit" class="btn brand z-depth-0" value="
        Alterar" >
        </form>
        </td>
        <td class=center> <form align="center" action="
        delete_produto.php" method="post">
        <input type="hidden" name="id" value="' . $produto["
        cod_produto"] . '">
        <input type="submit" class="btn brand z-depth-0" value="
        Eliminar" >
        </form>
        </td>
    </tr>';
}
echo '</table>';

//Fechar a ligação
mysqli_close($ligacao);

?>
<div class="center">
    <a href="add_produto.php" class="btn brand z-depth-0">Adicionar
    </a>
    <a href="index.php" class="btn brand z-depth-0">Voltar</a>
</div>
<?php include('templates/footer.php') ?>
</html>

```

Excerto de Código 4.6: "*display\_produto*"

Bakery App

Codigo de Produto	Nome	Composicao	Preco	TaxaIVA	Criado Em		
1	Paposeco	Farinha de Trigo T65, Agua, Fermento, Sal, Enzima E170, Enzima E472, Enzima E300	0.24€	6%	2024-07-01 17:19:41	ALTERAR	ELIMINAR
2	Pao de Quartos	Farinha de Trigo T65, Agua, Fermento, Sal, Enzima E170, Enzima E472, Enzima E300	0.95€	6%	2024-07-03 16:13:17	ALTERAR	ELIMINAR

ADICIONAR

VOLTAR

Bakery App 2024

Figura 4.5: Tabela de produtos

4.7.3 Implementação do *modify\_produto*

Utiliza a mesma ideologia do ficheiro *add\_poduto* 4.7.1 com a diferença de que recebe o id do produto selecionado e as variáveis do produto em questão não estão vazias.

O formulário quando apresentado vem já pré-preenchido com os dados do produto a alterar como se pode ver na figura 4.7.

Bakery App

Alterar Produto

Nome:

Pao de Quartos

Composição (Separar por vírgula):

Farinha de Trigo T65, Agua, Fermento, Sal, Enzima E170, En:

Preço:

0.95

Taxa de IVA(%):

6

ALTERAR

VOLTAR

Bakery App 2024

Figura 4.6: Formulário pré-preenchido para atualizar produto

Após os dados serem alterados ou não, serão feitas as verificações referidas anteriormente em 4.7.1 e é feita a atualização da base de dados, através da operação *update\_produto* 4.7.4.

```
<?php
```

```

include('config/ligacao_db.php');
$produtos_modify = [];

if (isset($_POST['id'])) {
    $id_produto = $_POST['id'];
}
else {
    print_r('Variavel nao encontrado');
}

//Obter os dados do produto com o cod_produto passado acima
//Query para obter todos os produtos
$sql_modify = "SELECT * FROM produto WHERE cod_produto = $id_produto
";

//Executar a query
$resultado_modify = mysqli_query($ligacao, $sql_modify);

//obter as linhas resultantes em forma de array
$produtos_modify = mysqli_fetch_assoc($resultado_modify);

mysqli_free_result($resultado_modify);
//mysqli_close($ligacao);

$nome = $produtos_modify['nome'];
$composicao = $produtos_modify['composicao'];
$pre_o = $produtos_modify['pre o'];
$taxaIVA = $produtos_modify['taxaIVA'];
$erros = array('nome' => '', 'composicao' => '', 'pre o' => '', '
    taxaIVA' => '', 'cod_produto' => '');

if(isset($_POST['submit'])) {
    //check nome
    if(empty($_POST['nome'])) {
        $erros['nome'] = '*o nome do produto    obrigat rio <br />
        ';
    } else {
        $nome = $_POST['nome'];
        if(!preg_match('/^[a-zA-Z\s]+$/ ', $nome)) { //A condi o
            diz que o 'nome' tem de come ar por um caracter de a a
            z maiusculo ou minusculo.
            $erros['nome'] = '*o nome do produto deve ser apenas
                letras e espa os <br />';
        }
    }
}

//preg_match - 2 args (express o regular, composicao
    propriamente dita)
// - verifica se o nome apenas contem letras e espa os e

```

```

        separado por virgulas
//check composicao
if(empty($_POST['composicao'])) {
    $erros['composicao'] = '*pelo menos um ingrediente
        obrigatório <br />';
} else {
    $composicao = $_POST['composicao'];
    if(!preg_match('/^([a-zA-Z0-9\s]+)(\s*[a-zA-Z0-9\s]*)*$/ ',
        $composicao)) {
        $erros['composicao'] = '*o nome do produto deve ser
            apenas letras e espaços <br />';
    }
}

//check preço
if(empty($_POST['preço'])) {
    $erros['preço'] = '*o preço do produto obrigatório <br />';
} else {
    $preço = $_POST['preço'];
    if(!preg_match('/^\d+(\.\d{1,2})?$/ ', $preço)) {
        $erros['preço'] = '*o preço do produto deve ser apenas
            números <br />';
    }
}

//check taxaIVA
if(empty($_POST['taxaIVA'])) {
    $erros['taxaIVA'] = '*o taxaIVA do produto obrigatório
        <br />';
} else {
    $taxaIVA = $_POST['taxaIVA'];
    if(!preg_match('/^(100|[1-9]?[0-9])$/ ', $taxaIVA)) {
        $erros['taxaIVA'] = '*o taxaIVA do produto deve ser
            apenas números <br />';
    }
}

//Verifica se existem erros no array
//Se existirem retorna true e irá devolver uma mensagem de erro
//Caso contrário retorna False e irá redirecionar para a página
inicial
if(array_filter($erros)) {
    //echo 'erros no formulário';
} else {
    //Função para proteger a BD de injeção de dados
    maliciosos na BD
    $nome = mysqli_real_escape_string($ligacao, $_POST['nome']);
    $composicao = mysqli_real_escape_string($ligacao, $_POST['

```

```

        composicao' ] );
        $pre o = mysqli_real_escape_string($ligacao , $_POST[ 'pre o
        ' ] );
        $taxaIVA = mysqli_real_escape_string($ligacao , $_POST[ '
        taxaIVA ' ] );
    }
} //Fim do POST check

?>
<!DOCTYPE html>
<html>
    <?php include( 'templates/header.php' ) ?>

    <section class="container grey-text">
        <h4 class="center">Alterar Produto</h4>
        <form action="update_produto.php" class="white" method="POST">
            <input type="hidden" name="id_produto" value="<?php echo
            $id_produto; ?>">
            <label>Nome:</label>
            <input type="text" name="nome" value="<?php echo
            htmlspecialchars($nome)?>">
            <div class="red-text"><?php echo $erros[ 'nome' ]; ?></div>
            <label>Composi o (Separar por virgula):</label>
            <input type="text" name="composicao" value="<?php echo
            htmlspecialchars($composicao)?>">
            <div class="red-text"><?php echo $erros[ 'composicao' ]; ?></
            div>
            <label>Pre o:</label>
            <input type="text" name="pre o" value="<?php echo
            htmlspecialchars($pre o)?>">
            <div class="red-text"><?php echo $erros[ 'pre o ' ]; ?></div>
            <label>Taxa de IVA(%):</label>
            <input type="text" name="taxaIVA" value="<?php echo
            htmlspecialchars($taxaIVA)?>">
            <div class="red-text"><?php echo $erros[ 'taxaIVA' ]; ?></div>
            <div class="center">
                <input type="submit" name="submit" value="Alterar" class="
                btn brand z-depth-0">
                <a href="display_produto.php" class="btn brand z-depth-0">
                Voltar</a>
            </div>
        </form>
    </section>

    <?php include( 'templates/footer.php' ) ?>
</html>

```

Excerto de Código 4.7: "*modify\_produto*"

#### 4.7.4 Implementação do *update\_produto*

Este ficheiro obtém os dados do formulário através da função *mysqli\_real\_escape\_string* pois é usado o *\$POST* que os envia para este ficheiro. Um dos dados que recebe é o *id\_produto* que será validado. Caso o *id\_produto* seja válido, é executada a query contida no excerto de código 4.8 que faz a atualização dos dados na base de dados, atualizando assim os dados da tabela.

```
<?php
include('config/ligacao_db.php');

if (isset($_POST['submit'])) {
    // Obter os dados do formulario
    $id_produto = mysqli_real_escape_string($ligacao, $_POST['id_produto']);
    $nome = mysqli_real_escape_string($ligacao, $_POST['nome']);
    $composicao = mysqli_real_escape_string($ligacao, $_POST['composicao']);
    $preco = mysqli_real_escape_string($ligacao, $_POST['preco']);
    $taxaIVA = mysqli_real_escape_string($ligacao, $_POST['taxaIVA']);

    // Verifica se o ID do produto e valido
    if (empty($id_produto)) {
        echo "ID do produto invalido.";
        exit;
    }

    //Atualizar dados na BD
    $sql_update = "UPDATE produto SET nome = '$nome', composicao = '$composicao', preco = '$preco', taxaIVA = '$taxaIVA' WHERE cod_produto = $id_produto";

    //Guardar na BD e verificar
    if(mysqli_query($ligacao, $sql_update)){
        //Sucesso
        header('Location: display_produto.php');
    } else {
        //Erro
        echo 'Erro na query: ' . mysqli_error($ligacao);
    }

    mysqli_close($ligacao);
} else {
    echo "Acesso invalido.";
}
?>
```

Excerto de Código 4.8: "Ficheiro para atualizar produto"

### 4.7.5 Implementação do delete\_produto

Como foi dito anteriormente, na tabela apresentada no *display\_produto* 4.6 é oferecida a opção de eliminar um produto. O botão disponibilizado obtém o id do produto a eliminar e executa a *query* como podemos ver no excerto de código apresentado a seguir.

```
<?php
    include( 'config/ligacao_db.php' );

    if ( isset($_POST['id']) ) {
        // Obter os dados do formulario
        $id_produto = mysqli_real_escape_string($ligacao , $_POST['id'] );

        // Verifica se o ID do produto e valido
        if ( empty($id_produto) ) {
            echo "ID do produto invalido.";
            exit;
        }

        //Atualizar dados na BD
        $sql_delete = "DELETE FROM produto WHERE cod_produto =
            $id_produto";

        //Guardar na BD e verificar
        if(mysqli_query($ligacao , $sql_delete)){
            //Sucesso
            header( 'Location: display_produto.php' );
        } else {
            //Erro
            echo 'Erro na query: ' . mysqli_error($ligacao);
        }

        mysqli_close($ligacao);
    } else {
        echo "Acesso invalido.";
    }
?>
```

Excerto de Código 4.9: "Ficheiro para eliminar produto"

### 4.7.6 Implementação do select\_pe

Apesar das operações usadas serem todas semelhantes é necessário falar do ficheiro *select\_pe*. Este ficheiro foi criado com o propósito de selecionar o ponto de entrega pretendido quando pretende adicionar uma encomenda nova.

Ou seja, quando o utilizador inicia o processo de fazer uma nova encomenda vai ter de escolher o ponto de entrega da encomenda, associando assim o id da nova encomenda aquele ponto de entrega.

Bakery App				
Codigo do Ponto de Entrega	Nome	Morada	Codigo da Rota	
1	Hotel da Vila	Rua Dr Esteves de Carvalho	1	SELECIONAR
2	Jose Alberto	Rua Dr Esteves de Carvalho	1	SELECIONAR
3	Restaurante Olival	N232	1	SELECIONAR

Figura 4.7: Tabela para seleccionar o ponto de entrega

A implementação desta operação pode ser vista no excerto de código 4.10

```

<!DOCTYPE html>
<html>

<?php include ( 'templates/header.php' ) ?>
<?php
    include ( 'config/ligacao_db.php' );

    //Query para obter todos os produtos
    $sql = 'SELECT * FROM ponto_entrega';

    //Executar a query
    $resultado = mysqli_query($ligacao , $sql);

    //obter as linhas resultantes em forma de array
    $pes = mysqli_fetch_all($resultado , MYSQLI_ASSOC);

    //Liberta o resultado da memoria
    mysqli_free_result($resultado);

    echo '<table border="10" cellpadding="2">
        <tr>
            <td class=center> <font face="Arial">Codigo do Ponto de
                Entrega</font> </td>
            <td class=center> <font face="Arial">Nome</font> </td>
            <td class=center> <font face="Arial">Morada</font> </td>
    </tr>
    </table>';

```



```

        <td class=center> <font face="Arial">Codigo da Rota</font>
        </td>

    </tr>';

//Percorrer o array de produtos
foreach($pes as $pe){
    echo '<tr>
        <td class=center> <font face="Arial">'. $pe['cod_pe']. '</font> </td>
        <td class=center> <font face="Arial">'. $pe['nome']. ' </font>
        > </td>
        <td class=center> <font face="Arial">'. $pe['morada']. ' </font> </td>
        <td class=center> <font face="Arial">'. $pe['cod_rota']. ' </font> </td>

        <td> <form action="add_encomenda.php" method="post">
            <input type="hidden" name="id_pe" value="'. $pe["cod_pe"] .'">
            <input type="submit" class="btn brand z-depth-0" value="
                Selecionar" >
            </form>
        </td>
    </tr>';
}
echo '</table>';

//Fechar a liga o
mysqli_close($ligacao);

?>
<a href="display_encomendas.php" class="btn brand z-depth-0">Voltar
</a>
<?php include('templates/footer.php') ?>
</html>

```

Excerto de Código 4.10: "select\_pe"

#### 4.7.7 Implementação do *display\_pe*

A operação *display\_pe* tem de ser mencionada pois difere do *display\_produto* em um aspeto.

O *display\_pe* é apresentado como redirecionamento de um botão na página que apresenta as rotas, *display\_rota*. Este botão, apresentado na figura 4.8, vai apresentar uma tabela com os pontos de entrega de uma determinada

rota, ou seja, vai enviar o id da rota selecionada que vai ser recebido na operação *display\_pe* que, como foi dito anteriormente, vai apresentar uma tabela com os pontos de entrega dessa rota, como mostra a figura 4.9. Podemos ver esta implementação no excerto de código 4.11

```
<!DOCTYPE html>
<html>

<?php include( 'templates/header.php' ) ?>
<?php
    include( 'config/ligacao_db.php' );
    $id_rota = '';
    if (isset($_POST['id_rota'])) {
        $id_rota = $_POST['id_rota'];
    }
    else {
        print_r('Variavel nao encontrado');
    }

    //Query para obter todos os produtos
    $sql = "SELECT * FROM ponto_entrega WHERE cod_rota = $id_rota ";
    ;

    //Executar a query
    $resultado = mysqli_query($ligacao, $sql);

    //obter as linhas resultantes em forma de array
    $pes = mysqli_fetch_all($resultado, MYSQLI_ASSOC);

    //Liberta o resultado da memoria
    mysqli_free_result($resultado);

    echo '<table border="10" cellpadding="2">
        <tr>
            <td class = center> <font face="Arial">Codigo do Ponto de
                Entrega</font> </td>
            <td class = center> <font face="Arial">Nome</font> </td>
            <td class = center> <font face="Arial">Morada</font> </td>
            <td class = center> <font face="Arial">Codigo da Rota</font>
                </td>

        </tr>';

    //Percorrer o array de produtos
    foreach($pes as $pe){
        echo '<tr>
            <td class=center> <font face="Arial">'. $pe['cod_pe']. '</
                font> </td>';
```

```

        <td class=center> <font face="Arial">' . $pe['nome'] . ' </font
        > </td>
        <td class=center> <font face="Arial">' . $pe['morada'] . ' </
        font> </td>
        <td class=center> <font face="Arial">' . $pe['cod_rota'] . ' </
        font> </td>

        <td class=center> <form action="modify_pe.php" method="post
        ">
            <input type="hidden" name="id" value="' . $pe["cod_pe"] .
            "'>
            <input type="submit" class="btn brand z-depth-0" value="
            Alterar" >
        </form>
    </td>
    <td class=center> <form action="delete_pe.php" method="post
    ">
        <input type="hidden" name="id" value="' . $pe["cod_pe"] . '
        ">
        <input type="submit" class="btn brand z-depth-0" value="
        Eliminar" >
    </form>
    </td>
</tr>';
}
echo '</table>';

//Fechar a liga o
mysqli_close($ligacao);

?>
<div class="center">
    <a href="add_pe.php" class="btn brand z-depth-0">Adicionar Ponto
    de Entrega</a>
    <a href="display_rota.php" class="btn brand z-depth-0">Voltar</a
    >
</div>
<?php include('templates/footer.php') ?>
</html>

```

Excerto de Código 4.11: "*display\_pe*"

Bakery App					
Codigo da rota	Distribuidor	Nome da Rota			
1	Entregas Manteigas	Cesar	PONTOS DE ENTREGA	ALTERAR	ELIMINAR
2	Rota 1 Manteigas	Cesar	PONTOS DE ENTREGA	ALTERAR	ELIMINAR
3	Rota 2 Manteigas	Fabio	PONTOS DE ENTREGA	ALTERAR	ELIMINAR
			ADICIONAR NOVA ROTA	VOLTAR	
Bakery App 2024					

Figura 4.8: *display\_rotas*

Bakery App					
Rota: Entregas Manteigas					
Codigo do Ponto de Entrega	Nome	Morada	Codigo da Rota		
1	Hotel da Vila	Rua Dr Esteves de Carvalho	1	ALTERAR	ELIMINAR
2	Jose Alberto	Rua Dr Esteves de Carvalho	1	ALTERAR	ELIMINAR
3	Restaurante Olival	N232	1	ALTERAR	ELIMINAR
			ADICIONAR PONTO DE ENTREGA	VOLTAR	

Figura 4.9: *display\_pe*

### 4.8 Conclusões

Neste capitulo foram analisados os códigos de implementação para a aplicação, bem como o resultado final.

## Capítulo

# 5

## ***Conclusões e Trabalho Futuro***

### **5.1 Conclusões Principais**

O desenvolvimento da *Bakery App*, utilizando HTML, PHP e CSS, demonstrou ser uma solução eficiente e prática para melhorar o processo de gestão das rotas e encomendas. Provou ser também uma possível solução mais económica para negócio de pequena dimensão.

Através da integração de tecnologias web, foi possível criar uma interface intuitiva e funcional, que facilita a navegação dos utilizadores e agiliza a administração das encomendas.

O desenvolvimento da aplicação serviu como uma oportunidade de aprendizagem e aplicação prática dos conhecimentos adquiridos durante o curso, evidenciando a importância da integração de diferentes linguagens de programação no desenvolvimento de soluções web.

### **5.2 Problemas Encontrados**

Os problemas encontrados foram:

- **Realização de encomenda** - da forma como a base de dados esta estruturada os produtos apenas podem ser adicionado um de cada vez, sendo que o mais intuitivo e prático seria selecionar as quantidades todas e fazer a adição dos produtos e quantidades desejadas de uma só vez.

### 5.3 Trabalho Futuro

A aplicação tem vários aspetos que podem ser melhorados e novas funcionalidades que podem ser implementadas, como a implementação de notificações automáticas, pagamentos *online*.

## ***Bibliografia***

- [1] Visual studio code. <https://code.visualstudio.com/>.
- [2] Html. <https://www.w3schools.com/html/>.
- [3] Css. <https://www.w3schools.com/CSS>.
- [4] Php. <https://www.w3schools.com/php/>.
- [5] Xampp. <https://www.apachefriends.org/index.html>.