Escola Secundária Ferreira Dias

Curso Profissional Técnico de Gestão e Programação de Sistemas Informáticos

2023/ 2026

Desenvolvimento de uma aplicação em PHP e MySQL

Relatório do projeto

Danylo Duma

2ºGI

Agualva-Cacém, maio de 2025

Tema

Desenvolvimento de uma aplicação em PHP e MySQL

Orientador

Prof.ª Rute Alves

_____

(assinatura)

Siglas e Acrónimos

PHP: Hypertext Preprocessor (Linguagem de programação para Back-End).

MySQL: My Structured Query Language (Sistema de Gestão de Base de Dados).

CRUD: Create, Read, Update and Delete (Operações básicas em bases de dados).

SGBD: Sistema de Gestão de Base de Dados.

HTML: HyperText Markup Language (Linguagem de programação para Front-End).

CSS: Cascading Style Sheets (Linguagem para estilização e efeitos visuais).

PK: Primary Key (Chave Primária).

FK: Foreign Key (Chave Estrangeira).

Glossário

Back-End: A parte de uma aplicação web que lida com a lógica do servidor, bases de dados e a gestão de dados, não visível para o utilizador final.

Front-End: A parte de uma aplicação web com a qual o utilizador interage diretamente, incluindo a interface gráfica, elementos de design e a forma como a informação é apresentada.

Modelo Lógico: Uma representação abstrata da estrutura de uma base de dados, que mostra as entidades, os seus atributos e os relacionamentos entre elas, independentemente do SGBD utilizado.

Modelo Físico: Uma representação concreta da estrutura de uma base de dados, que detalha como os dados serão armazenados no SGBD específico, incluindo tabelas, colunas, tipos de dados, chaves primárias e estrangeiras.

Cookies: Pequenos arquivos de texto armazenados no navegador do utilizador que podem ser usados para guardar informações sobre o utilizador ou a sessão.

Array: Uma estrutura de dados que armazena uma coleção de elementos.

Índice

Índice de Figuras

1. Introdução

1.1 Enquadramento

O projeto consiste em desenvolver uma aplicação Web com a linguagem de programação PHP, juntamente com MySQL.

1.2 Apresentação do projeto

O projeto consiste na construção de uma aplicação Web com a linguagem de programação PHP, que irá funcionar em simultâneo com MySQL para imprimir os dados de um modelo de base de dados e simular um CRUD num servidor local.

1.3 Fundamentação da escolha do projeto

A escolha deste projeto baseia-se no conhecimento e experiência adquiridos na área da programação Web, bem como na vertente de Back-End, bem como na manipulação de Sistemas de Gestão de Base de Dados (SGBD) que funcionam em conjunto com aplicações Web. Além disso, o domínio desta área da programação é essencial na maioria dos setores do mercado de trabalho relacionados com sistemas informáticos. Este conhecimento permite aos alunos evoluir em vários aspetos profissionais e pode abrir portas para um futuro mais promissor.

1.4 Finalidades

O principal objetivo deste projeto é demonstrar, de forma prática, o funcionamento de uma aplicação Web com Back-End integrado e interligada a um SGBD. Além disso, serve como uma ferramenta de aprendizagem sólida que os alunos poderão aplicar em contextos reais no mundo profissional.

1.5 Tecnologias utilizadas

As tecnologias utilizadas neste projeto incluem linguagens de programação para o Front-End: HTML, CSS, Bootstrap e JavaScript. Para o Back-End, foram utilizadas as linguagens PHP e JavaScript. Como Sistema de Gestão de Base de Dados foi utilizado o MySQL. O projeto foi desenvolvido e testado num servidor local configurado com o XAMPP.

2. Modelo lógico

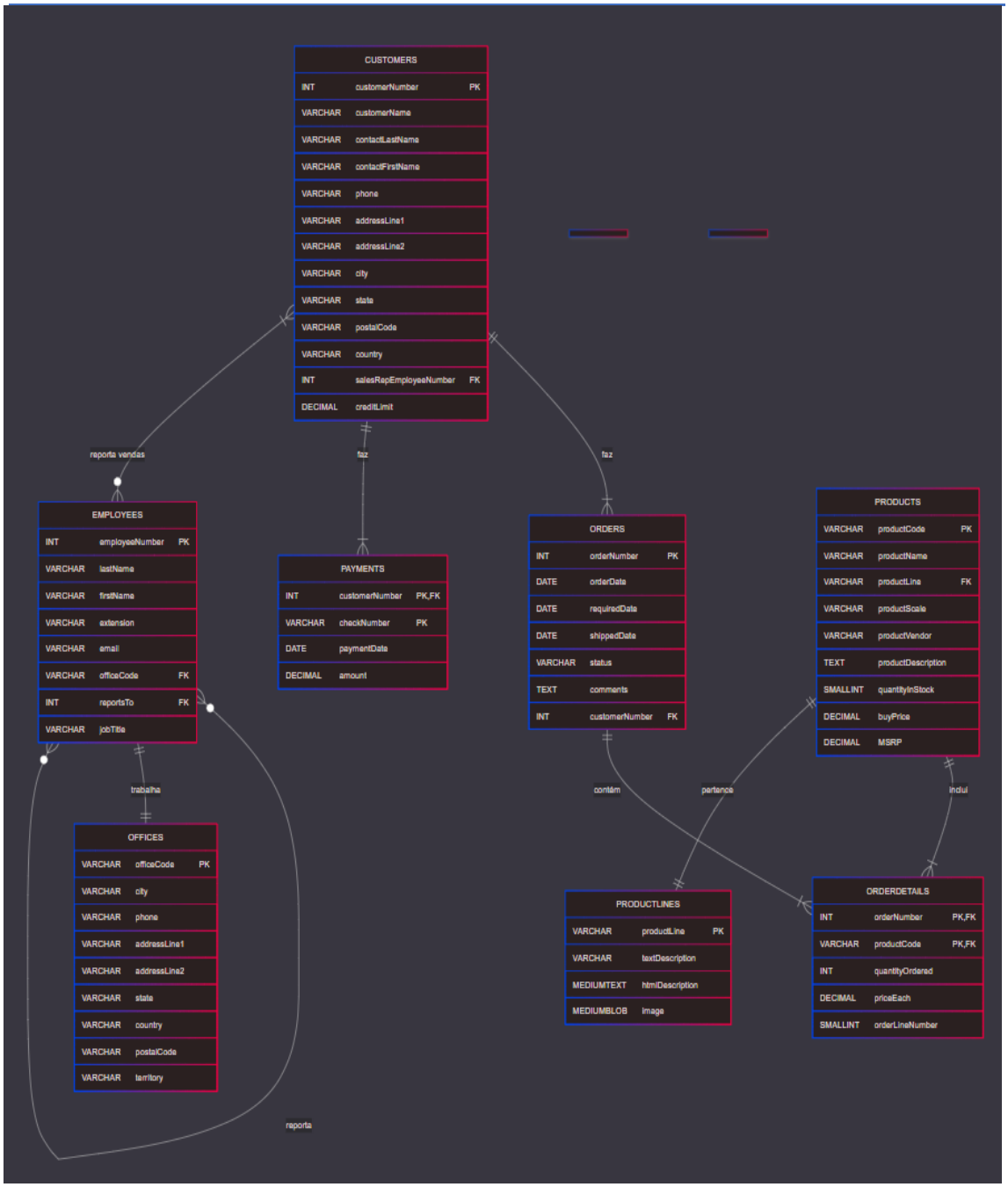

Fig. 1 – Modelo lógico da base de dados

3. Modelo Físico

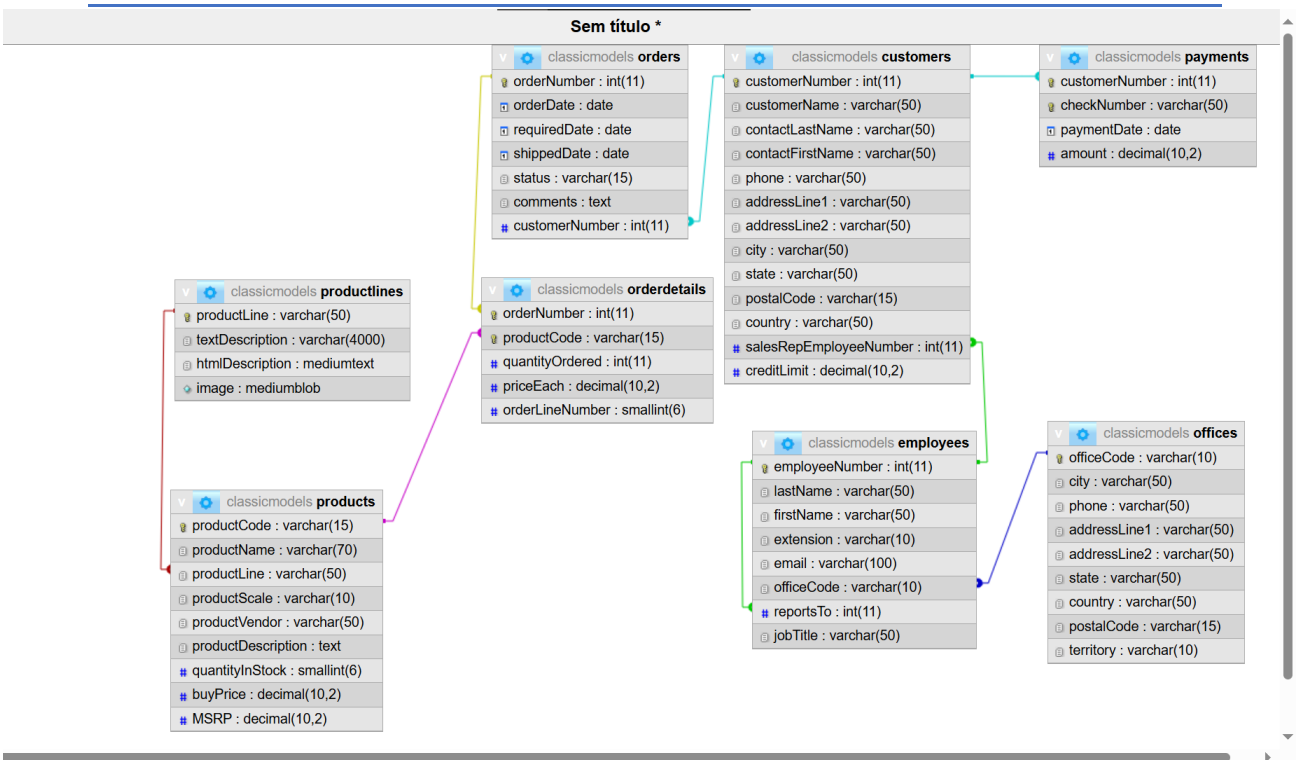

Fig. 2 – Modelo físico da base de dados

4. Desenvolvimento

O principal foco e objetivo deste projeto é o desenvolvimento de uma aplicação Web que forneça informações sobre uma base de dados chamada ClassicModels. Através desta base de dados, a aplicação disponibiliza uma página de login e um sistema CRUD (Create, Read, Update e Delete).

Este projeto será designado como PHP_ClassicModels.

Estrutura do PHP_ClassicModels

A estrutura do projeto encontra-se dividida nas seguintes partes:

Pastas:

- **crud/** - Processamento de ações na base de dados. Responsável pelas funcionalidades de adicionar, editar e apagar registos. Os arquivos dentro da pasta crud/ são:
  - ***add.php*** – Responsável pela inserção de dados;
  - ***edit.php*** – Responsável pela edição de dados;
  - ***delete.php*** – Responsável pela remoção de dados.

- **css/** - Personalização e efeitos visuais. Responsável pela aparência do WebSite. Contém apenas um arquivo .css da qual estiliza os detalhes do WebSite e que o Bootstrap não é capaz de estilizar. O arquivo dentro da pasta css/ é:
  - ***style.css* –** Responsável pela aparência de certos detalhes do WebSite.

- **includes/** - Utilitários e conexões. Responsável pela conexão à base de dados e representação dinâmica dos dados da ClassicModels. Os arquivos dentro da pasta includes/ são:
  - ***db_columns.php*** – Representa de forma dinâmica as colunas nas tabelas;
  - ***db_connect.php*** – Efetua uma ligação à base de dados Classic Models.

- **Pages/** - Tela principal. Interface dinâmica com CRUD embutido. O arquivo que tem a pasta pages/ é:
  - ***dashboard.php*** – Tela principal com a representação das tabelas da base de dados e CRUD com funcionamento dinâmico.

Arquivos restantes:

- ***index.php*** – Página de login. Primeira página a ser acessada antes do dashboard.php
- ***login.php*** – Validação do utilizador com a ajuda da tabela Employees. Processa a sessão.
- ***logout.php*** – Limpa os cookies, apaga as variáveis de sessão e destrói.

Base de dados ClassicModels

A base de dados ClassicModels é um template pré-definido, que já contém múltiplos registos e relações. Inicialmente foi exportada e, posteriormente, importada para o phpMyAdmin.

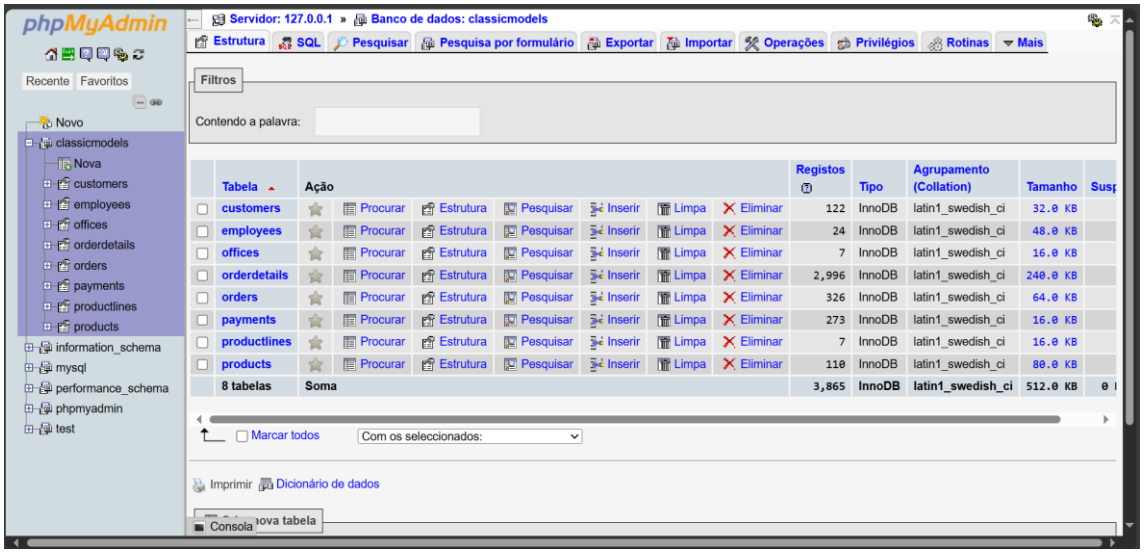

Fig. 3 – Tabelas da ClassicModels

Contém oito tabelas inter-relacionadas, que serão representadas na interface do projeto PHP_ClassicModels.

Representação do PHP_ClassicModels

Ao aceder ao website, o utilizador é direcionado para o index.php, uma página de login que garante segurança no acesso. O sistema de login utiliza a tabela employees, aceitando os campos de e-mail e número do funcionário.

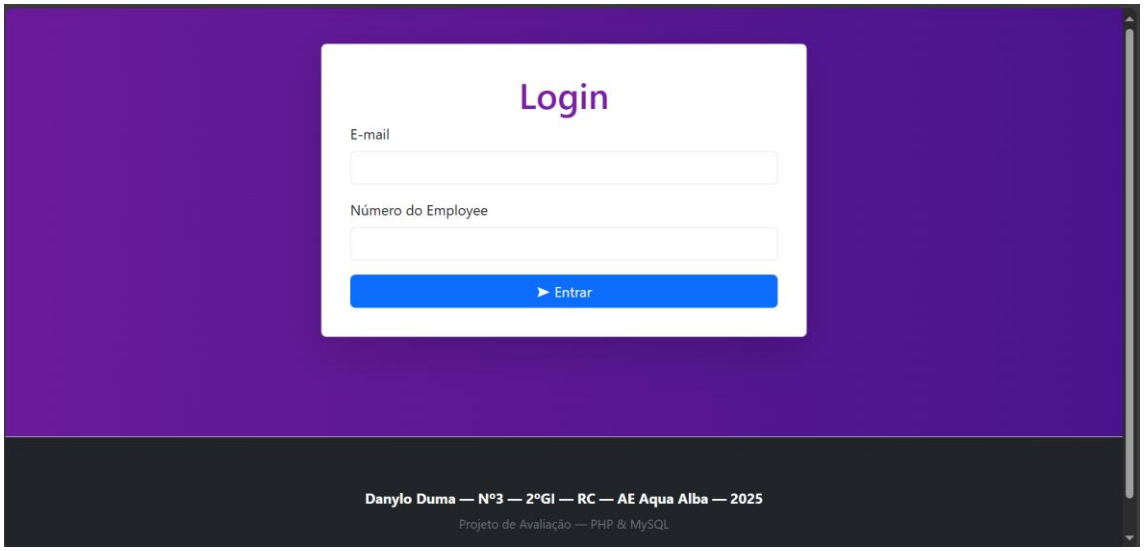

Fig. 4 – Página inicial de login

A interface foi construída com recurso ao Bootstrap, com apoio de um ficheiro CSS personalizado para estilizações adicionais.

```
7   <html lang="pt-pt">
17
18  <body class="bg-light">
19    <div class="container">
20      <div class="row justify-content-center">
21        <div class="col-md-6">
22          <div class="card shadow-lg p-3 mb-5 bg-white rounded">
23            <div class="card-body">
24              <h1 class="text-center">Login</h1>
25              <form action="login.php" method="post">
26                <div class="mb-3">
27                  <label for="email" class="form-label">E-mail</label>
28                  <input type="email" class="form-control" id="email" name="email" required>
29                </div>
30                <div class="mb-3">
31                  <label for="employeeNumber" class="form-label">Número do Employee</label>
32                  <input type="password" class="form-control" id="employeeNumber" name="employeeNumber" re
33                </div>
34                <button type="submit" class="btn btn-primary w-100">Entrar</button>
35              </form>
36              <?php if (isset($_GET['error'])): ?>
37                <div class="alert alert-danger mt-3" role="alert">
38                  <?php echo htmlspecialchars(string: $_GET['error']); ?>
39                </div>
40              <?php endif; ?>
41            </div>
42          </div>
43        </div>
44      </div>
45    </div>
```

Fig. 5 – Estrutura de design do login

A página de login foi elaborada com um formulário com método POST, que irá receber os dados do Employee.

```
25              <form action="login.php" method="post">
26                <div class="mb-3">
27                  <label for="email" class="form-label">E-mail</label>
28                  <input type="email" class="form-control" id="email" name="email" required>
29                </div>
30                <div class="mb-3">
31                  <label for="employeeNumber" class="form-label">Número do Employee</label>
32                  <input type="password" class="form-control" id="employeeNumber" name="employeeNumber" required>
33                </div>
34                <button type="submit" class="btn btn-primary w-100">Entrar</button>
35              </form>
```

Fig. 6 – Formulário de login em código

Quando carregado no botão “Entrar”, os dados inseridos irão ser armazenados e logo de seguida encaminhados para o login.php, que irão ser verificados e comparados os dados inseridos com os dados que existem na tabela Employees.

```
1  <?php
2  session_start();
3  include_once('includes/db_connect.php');
4
5  if ($_SERVER["REQUEST_METHOD"] == "POST") {
6      $email = $_POST['email'];
7      $employeeNumber = $_POST['employeeNumber'];
8
9      $sql = "SELECT employeeNumber FROM employees WHERE email = ? AND employeeNumber = ?";
10     $stmt = $conn->prepare(query: $sql);
11     $stmt->bind_param(types: "ss", var: &$email, vars: &$employeeNumber);
12     $stmt->execute();
13     $stmt->store_result();
14
15     if ($stmt->num_rows == 1) {
16         $_SESSION['employeeNumber'] = $employeeNumber;
17         header(header: "Location: pages/dashboard.php");
18     } else {
19         header(header: "Location: index.php?error=Credenciais inválidas");
20     }
21
22     $stmt->close();
23     $conn->close();
24 }
25 ?>
```

Fig. 7 – Funcionamento do login.php

O login.php funciona da seguinte forma: Ele começa a executar o primeiro if() quando é enviado um request com o método POST, de acordo com a linha 5 do código. Ao longo deste if() serão executados as seleções e procuras do registo ao longo da tabela Employees, depois é armazenado o resultado e logo a seguir comparado, caso os valores forem equivalentes o login.php irá encaminhar para o dashboard.php, se não irá

encaminhar de volta para o index.php mas já com a mensagem de erro, como pode ser visto na linha 19 depois do else().

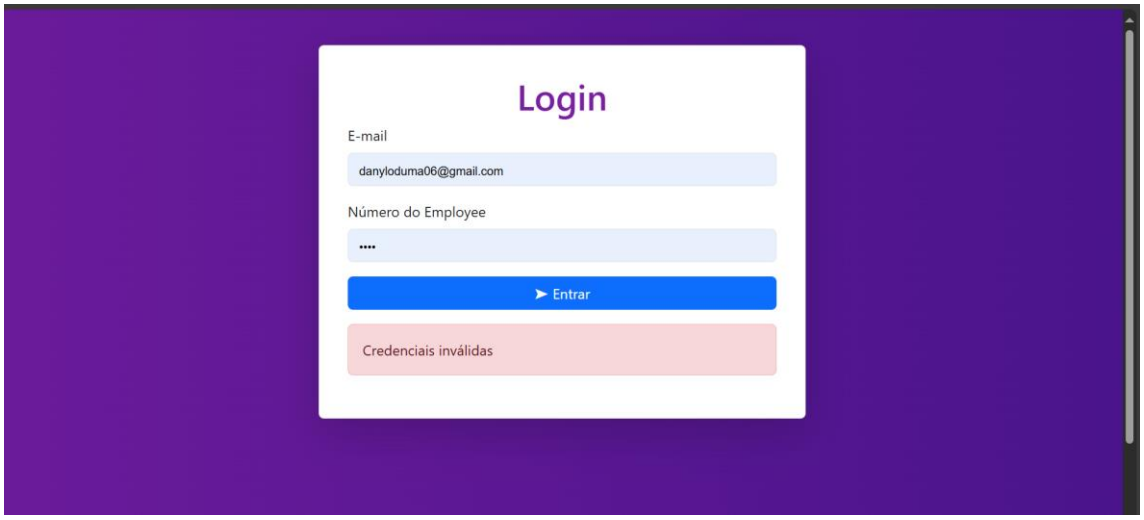

Fig. 8 – Mensagem de erro na página de login

Em simultâneo com o funcionamento do login.php, é executado o arquivo db_connect.php, para a conexão à base de dados ClassicModels. Caso dê erro durante a conexão, irá retornar uma mensagem de erro.

```php
<?php
//Arquivo de conexão à base dados
$servername = "localhost"; //nome do servidor
$username = "root"; //nome do utilizador (root)
$password = ""; //password (nenhum)
$dbname = "classicmodels"; //base de dados selecionada

//conexão à base de dados
$conn = new mysqli(hostname: $servername, username: $username, password: $password, database: $dbname);

//caso dê erro durante a conexão, retorna a mensagem de erro
if ($conn->connect_error) {
  die("Erro na conexão: " . $conn->connect_error);
}
?>
```

Fig. 9 – Processamento da conexão à base de dados

Quando os dados inseridos no formulário forem equivalentes a um dos registos da tabela Employees, teremos acesso a todos os registos da base de dados em dashboard.php.

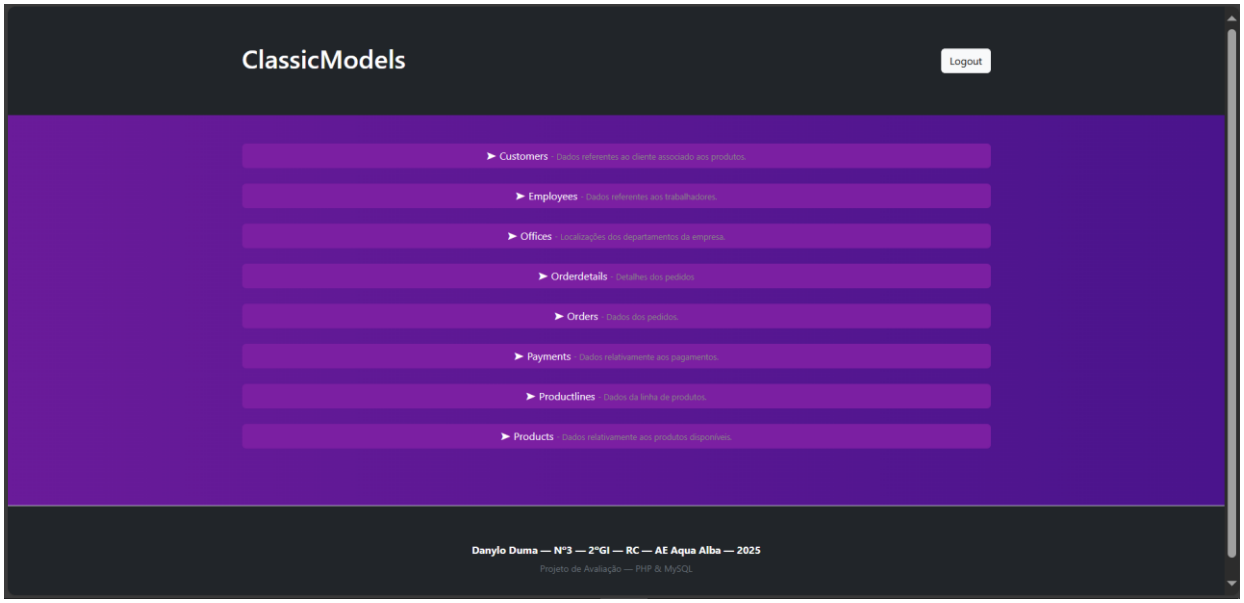

Fig. 10 – Página principal das tabelas do ClassicModels

(Nota: A representação visual do WebSite na imagem anterior foi reduzida pra uma melhor visualização do conteúdo da página.)

A página dashboard.php contém diversas opções das tabelas da ClassicModels, ao carregar em uma das opções, irá funcionar um dropdown com os registos da tabela selecionada. A representação da tabela é feita a com ajuda do db_columns.php, para representar as tabelas de forma dinâmica.

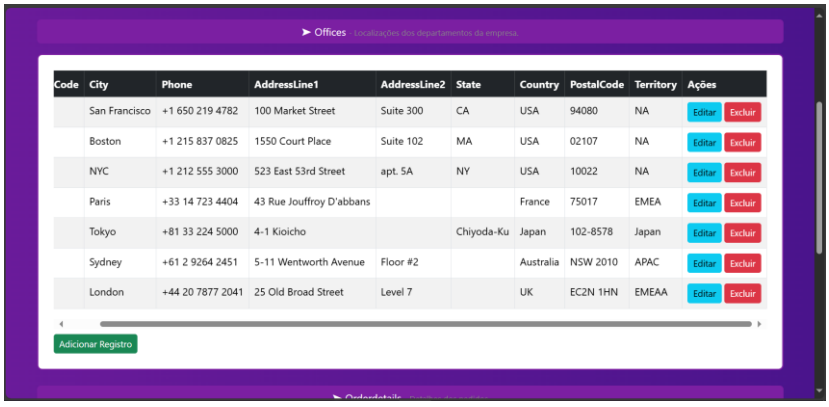

Fig. 11 – Representação das tabelas

```php
<?php
include 'db_connect.php';

$table = $_GET['table'] ?? '';
if (!$table) {
  http_response_code(response_code: 400);
  echo json_encode(value: ['erro' => 'Tabela não especificada']);
  exit;
}

$sql = "SELECT COLUMN_NAME, COLUMN_KEY, EXTRA, COLUMN_TYPE
        FROM INFORMATION_SCHEMA.COLUMNS
        WHERE TABLE_SCHEMA = 'classicmodels' AND TABLE_NAME = ?";
$stmt = $conn->prepare(query: $sql);
$stmt->bind_param(types: "s", var: &$table);
$stmt->execute();
$result = $stmt->get_result();

$columns = [];
while ($row = $result->fetch_assoc()) {
  $columns[] = $row;
}

echo json_encode(value: $columns);
?>
```

Fig. 12 – Armazenamento de informação das colunas

Cada opção representativa da tabela tem a opção de adicionar novo registo. Com isso, é possível preencher dados personalizados pelo utilizador.

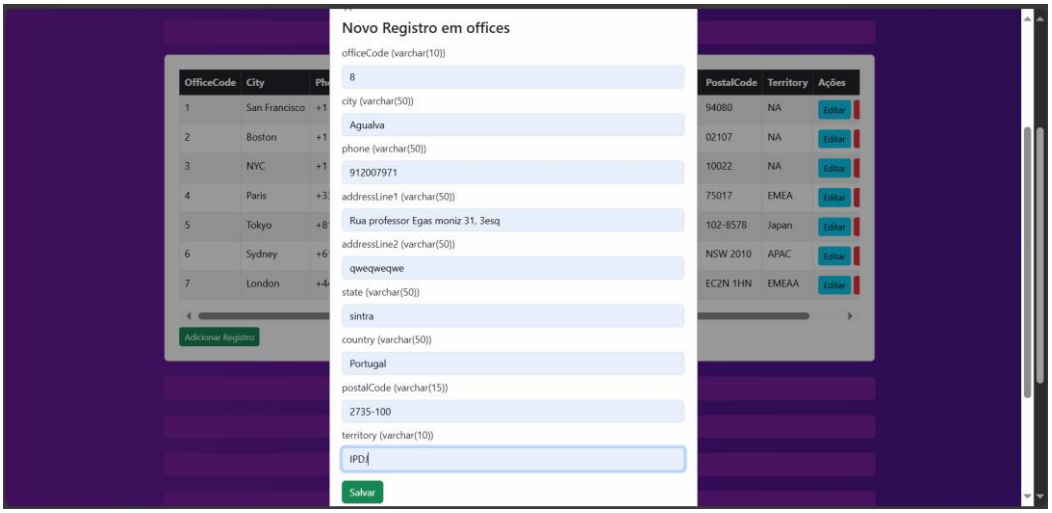

Fig. 13 – Preenchimento de dados para a tabela

Além disso, cada registo tem a opção de editar ou apagar do ClassicModels. Na opção de editar o registo, certos campos de preenchimento estão bloqueados, pois são os campos que têm valores e relações entre chaves-primárias (PK) e chaves-estrangeiras (FK) das tabelas.

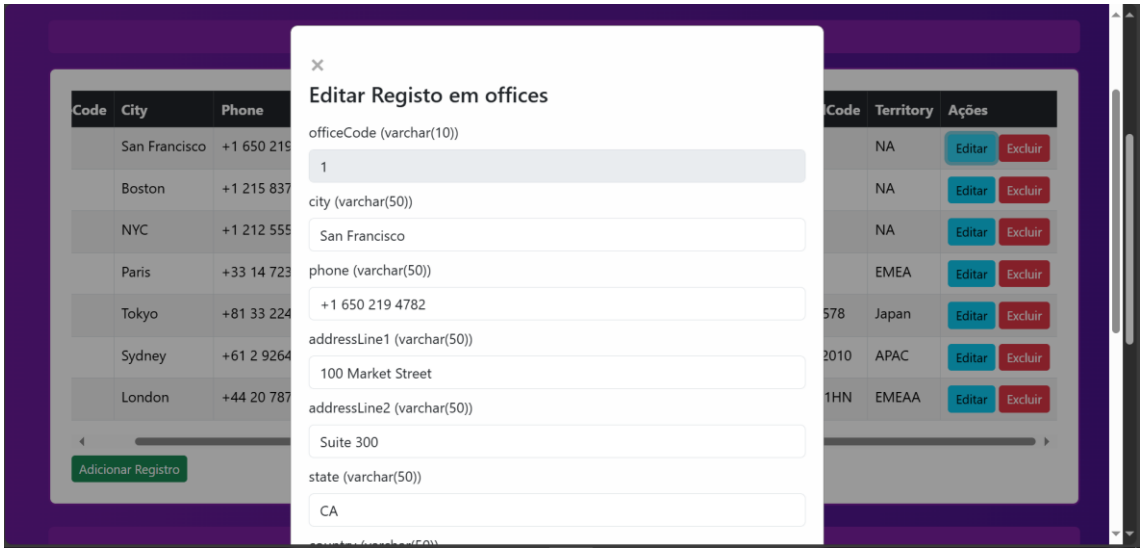

Fig. 14 – Edição de dados da tabela

Ao tentar apagar um registo, irá perguntar se tem a certeza em apagar o registo da base de dados e, ao confirmar a exclusão, irá retornar a mensagem que foi excluída com sucesso e consequentemente apagará o registo.

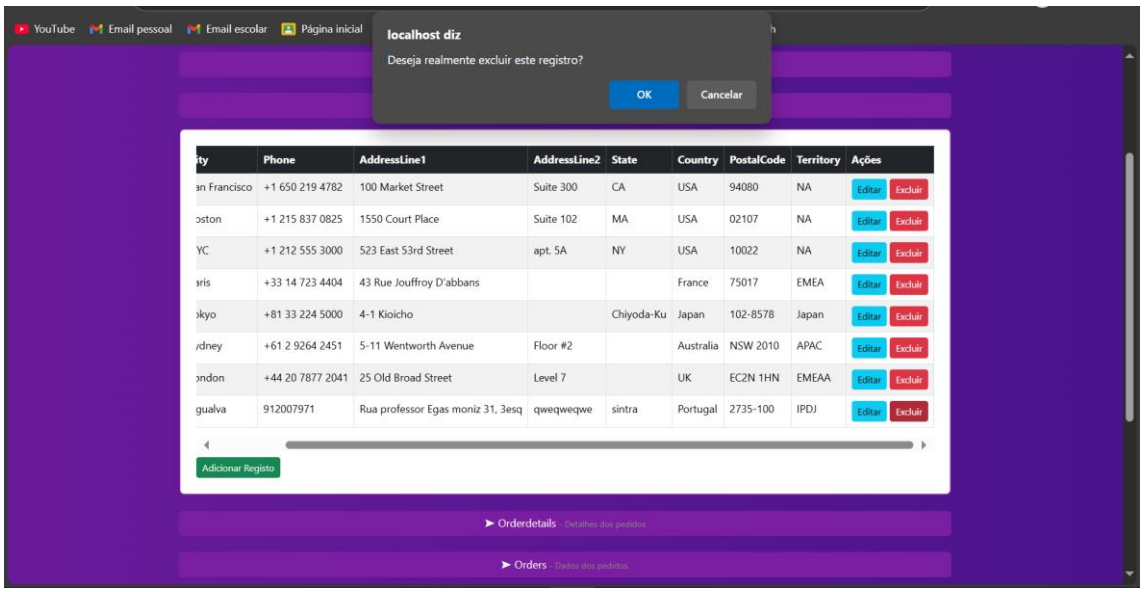

Fig. 15 – Confirmação da exclusão de registos

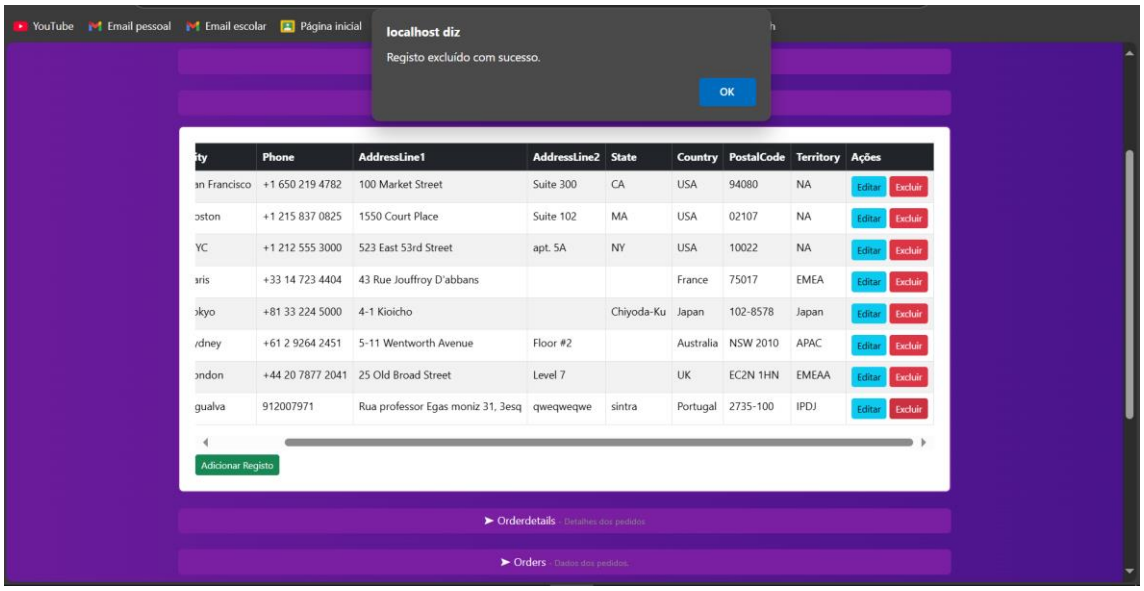

Fig. 16 – Exclusão de dados efetuada com sucesso

Caso o registo tenha relações com os registos das outras tabelas, irá informar que o registo tem relações com outras tabelas.

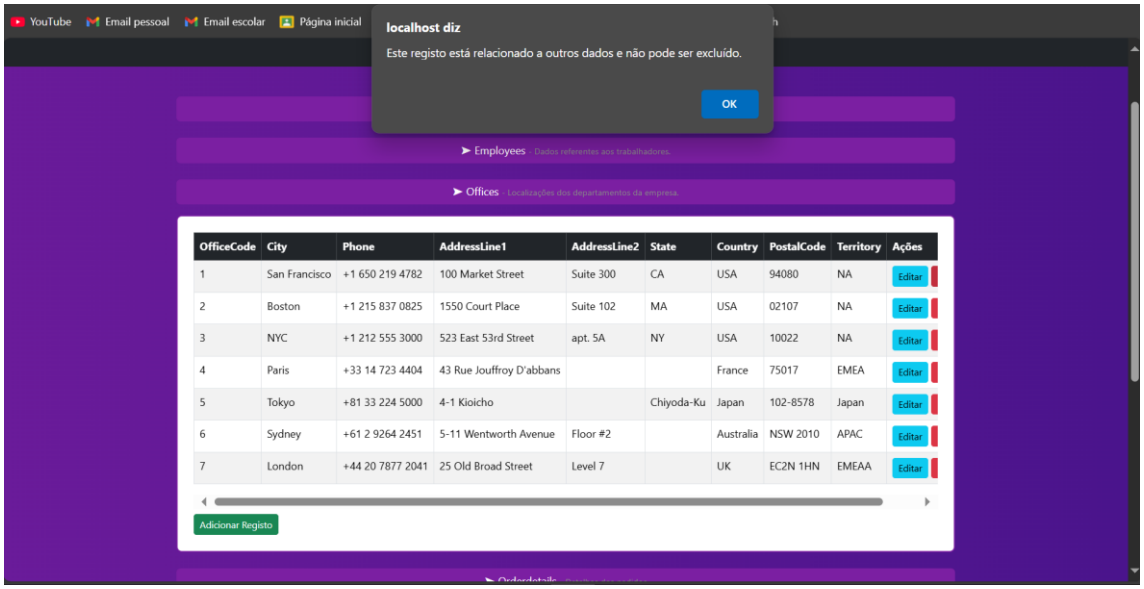

Fig. 17 – Mensagem de erro de exclusão do registo

Na parte do servidor, o processamento destes dados funciona da seguinte maneira:

Quando é acionado “Adicionar novo registo”, serão inseridos diversos dados e logo em seguida enviados para o add.php, que é aonde seram filtrados os dados inseridos e, se a filtragem validar os dados, serão inseridos no ClassicModels.

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $table = $_POST['table'];
  unset($_POST['table']);

  // Remove o campo 'id' se ele não existir na tabela (precaução extra)
  if (isset($_POST['id'])) {
    unset($_POST['id']);
  }

  // Remove campos vazios (ex: auto_increment não preenchido)
  $filtered = array_filter(array: $_POST, callback: function ($v): bool {
    return $v !== '';
  });

  if (empty($filtered)) {
    die('Nenhum dado válido enviado.');
  }
  $columns = array_keys(array: $filtered);
  $values = array_values(array: $filtered);

  $placeholders = implode(separator: ',', array: array_fill(start_index: 0, count: count(value…$values), '?'));
  $columnsList = implode(separator: ',', array: $columns);

  $types = str_repeat(string: 's', times: count(value: $values));

  $sql = "INSERT INTO $table ($columnsList) VALUES ($placeholders)";
  $stmt = $conn->prepare(query: $sql);

  if (!$stmt) {
    die("Erro no prepare: " . $conn->error);
  }

  $stmt->bind_param(types: $types, ...var: &$values);

  if ($stmt->execute()) {
    echo "success";
  } else {
    echo "Erro ao inserir: " . $conn->error;
  }
```

Fig. 18 – Processamento do add.php

Na parte do editar um registo, certos campos de preenchimento são bloqueados para não interferir nas relações do ClassicModels. Quando os campos são editados, estes dados editados são enviados para o edit.php, que é aonde irão ser armazenados numa array e encaminhados para o registo editado da base de dados para atualizar a informação do registo.

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $table = $_POST['table'];
  $idValue = $_POST['id'];
  unset($_POST['table'], $_POST['id']);

  // Descobre o nome da chave primária real da tabela
  $pkQuery = "SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS
              WHERE TABLE_SCHEMA = 'classicmodels' AND TABLE_NAME = ? AND COLUMN_KEY = 'PRI' LIMIT 1";
  $stmt = $conn->prepare(query: $pkQuery);
  $stmt->bind_param(types: "s", var: &$table);
  $stmt->execute();
  $stmt->bind_result(var: &$primaryKey);
  $stmt->fetch();
  $stmt->close();

  if (!$primaryKey) {
    die("Chave primária não encontrada.");
  }

  // Monta os pares campo = ?
  $set = [];
  $values = [];
  foreach ($_POST as $col => $val) {
    $set[] = "$col = ?";
    $values[] = $val;
  }

  $setStr = implode(separator: ', ', array: $set);
  $values[] = $idValue; // adiciona o valor da PK no final

  $types = str_repeat(string: 's', times: count(value: $values));

  $sql = "UPDATE $table SET $setStr WHERE $primaryKey = ?";
  $stmt = $conn->prepare(query: $sql);

  if (!$stmt) {
    die("Erro ao preparar: " . $conn->error);
  }
```

Fig. 19 – Processamento do edit.php

Por último, na parte de excluir um registo, este pedido será encaminhado para o delete.php, que é aonde irá ser procurado este registo e verificado quanto às relações, caso o registo tenha alguma relação, irá ser retornado a mensagem de erro, caso contrário, irá executar o DELETE para excluir o registo

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  try {
    // Descobre a chave primária
    $pkQuery = "SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS
                WHERE TABLE_SCHEMA = 'classicmodels' AND TABLE_NAME = ? AND COLUMN_KEY = 'PRI' LIMIT 1";
    $stmt = $conn->prepare(query: $pkQuery);
    $stmt->bind_param(types: "s", var: &$table);
    $stmt->execute();
    $stmt->bind_result(var: &$pk);
    $stmt->fetch();
    $stmt->close();

    if (!$pk) {
      throw new Exception(message: "Chave primária não encontrada.");
    }

    // Executa o DELETE com tratamento de exceção
    $sql = "DELETE FROM $table WHERE $pk = ?";
    $stmt = $conn->prepare(query: $sql);
    $stmt->bind_param(types: "s", var: &$id);
    $stmt->execute();

    echo "success";

  } catch (mysqli_sql_exception $e) {
    if (strpos(haystack: $e->getMessage(), needle: 'a foreign key constraint fails') !== false) {
      echo "relacionado";
    } else {
      echo "erro:" . $e->getMessage();
    }
  }
}
```

Fig. 20 – Processamento do delete.php

Quando o utilizador quiser encerrar a sessão, basta carregar no botão “Logout” que se situa no cabeçalho do PHP_ClassicModels. Ao carregar nele, irá encaminhar o pedido de Logout para logout.php, que irá limpar as variáveis de utilizador, limpar os cookies e destruir a sessão.

```php
<?php

session_start();

$_SESSION = array();

if (ini_get(option: "session.use_cookies")) {
  $params = session_get_cookie_params();
  setcookie(
    name: session_name(),
    value: '',
    expires_or_options: time() - 42000,
    path: $params["path"],
    domain: $params["domain"],
    secure: $params["secure"],
    httponly: $params["httponly"]
  );
}

session_destroy();
header(header: "Location: index.php");
?>
```

Fig. 21 – Processamento do logout.php

5. Melhorias e correção de erros

Em termos de desenvolvimento, todas as funcionalidades exigidas no enunciado foram devidamente implementadas. No entanto, poderiam ser adicionados alguns detalhes extra que, embora pequenos, contribuiriam significativamente para destacar e melhorar o funcionamento do projeto.

Exemplos disso incluem a implementação de uma prevenção mais rigorosa contra SQL Injection, a possibilidade de adicionar tabelas dinamicamente através do painel de administração do website, a padronização das mensagens de erro em formato JSON ou com códigos HTTP — que facilita futuras integrações com APIs —, bem como a introdução de um sistema de registos (logs) externos.

Estas e outras melhorias poderão ser exploradas em projetos futuros, com o objetivo de tornar a aplicação mais robusta e segura.

6. Reflexão Crítica

Com este projeto, foi mais divertido e interessante desenvolver e utilizar diversas tecnologias. As tecnologias implementadas são precisamente aquelas que muitas empresas usam nas suas inovações.

Por exemplo, o PHP com MySQL é frequentemente utilizado em várias aplicações web e em sistemas de gestão e processamento de dados. O Bootstrap, por sua vez, é um dos frameworks mais populares e versáteis para a estilização do front-end.

Em outras palavras, trata-se de um conjunto de tecnologias muitas das vezes adotadas no mercado de trabalho, e cuja utilização em projetos académicos proporciona uma valiosa compreensão prática das ferramentas que as empresas usam no seu dia a dia para criar soluções inovadoras.

7. Conclusão

8. Webgrafia

aNotePad – Acedido em 3 de maio 2025 - https://pt.anotepad.com/

Bootstrap – Acedido em 3 de maio 2025 - https://getbootstrap.com/docs/5.3/examples/footers/

Bootstrap – Acedido em 4 de maio 2025 - https://getbootstrap.com/docs/5.3/layout/grid/

Stack Overflow – Acedido em 4 de maio 2025 - https://pt.stackoverflow.com/questions/268124/fetch-vs-xmlhttprequest-vs-jquery-ajax

Bootstrap – Acedido em 29 de abril 2025 - https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js

Mermaid – Acedido em 4 de maio 2025 - https://www.mermaidchart.com/app/projects/b2285f2a-edc5-43a0-97bc-936347c0b47a/diagrams/ba213189-c8fa-480a-9347-8286a03f4e4f/version/v0.1/edit

Nota:

Para a elaboração do projeto foram utilizados recursos de inteligência artificial (ChatGPT, Gemini), exclusivamente com o objetivo de apoio educativo, esclarecimento de dúvidas técnicas, correção da língua portuguesa e melhoria na estruturação do código, não sendo utilizada para fins de plágio ou cópia indevida de trabalhos alheios.