Projeto BD - Parte 3

Nome do Aluno	Número do Aluno	Contribuição (%)	Esforço (horas)
Martim Mendes	102932	34%	60
Lourenço Matos	103432	34%	60
João Correia	103544	32%	60

Número do Grupo: 11
Turno: BD2L03
Nome do Professor do Laboratório: Pedro Miguel Leão Veloso Dias

Arquitetura da Aplicação Web

Layout da Aplicação

Ao iniciarmos a aplicação, somos direcionados para a página `admin_login.html`, que nos permite entrar no sistema como administrador usando a senha: **01**. Após o login bem-sucedido, somos direcionados para a página `admin_dashboard.html`, que atua como a principal interface de usuário para todas as funcionalidades da administração. O painel do administrador está dividido em quatro seções principais:

- **Controle de pedidos:** Colocar ou pagar um pedido.
- **Controle de produtos:** Registar, remover, alterar o preço ou alterar a descrição de um produto.
- **Controle de clientes:** Registar ou remover um cliente.
- **Controle de fornecedores:** Registar ou remover um fornecedor.

Ao clicar em qualquer um dos botões para realizar uma das ações, uma janela modal é exibida, apresentando a(s) tabela(s) relevante para a ação. Após efetuar uma ação, somos levados a uma página de sucesso, que nos apresenta o estado atual da tabela que alterámos e os atributos da ação, ou em caso de erro, uma página a exibir uma mensagem de erro. Ambas as páginas possuem um botão de redirecionamento para a `admin_dashboard.html`.

Estrutura de Diretórios

A estrutura de diretórios da aplicação é a seguinte:

- A pasta `/web` contém `admin_dashboard.html`, `admin_login.html`, `index.html`, bem como subpastas `cgi-bin/` e `impossible/`.
- A subpasta `impossible/to/find/this/` contém o arquivo `not_login.py`, que armazena as credenciais de acesso à base de dados.
- A subpasta `cgi-bin/` contém todos os arquivos CGI que são usados para buscar dados das tabelas e implementar a lógica do programa. O arquivo `python_backend_admin.cgi` atua como o principal back-end que recebe requests e atualiza a base de dados.

CGI e as suas Funções

Os arquivos CGI na pasta `cgi-bin/` incluem:

- `get_clients.cgi` - Este arquivo envia um request para receber os dados da tabela de clientes.
- `get_products.cgi` - Este arquivo envia um request para receber os dados da tabela de produtos.
- `get_orders_non_paid.cgi` - Este arquivo envia um request para receber os pedidos não pagos.
- `get_suppliers.cgi` - Este arquivo envia um request para receber os dados da tabela de fornecedores.
- `python_backend_admin.cgi` - Este arquivo é o principal arquivo de back-end que implementa toda a lógica do programa, recebendo requests e atualizando a base de dados.

Resumindo, a aplicação web está bem estruturada e projetada para ser altamente funcional, com uma clara separação de responsabilidades entre as várias partes da aplicação.

A nossa web application: https://web.tecnico.ulisboa.pt/ist1103432/.

Índices

7.1

Um índice na tabela `product` (coluna `price`) e na tabela `order` (coluna `name`) do tipo composite <price,date> não é tão eficiente uma vez que a igualdade é só testada após o teste de range e assim como teríamos ordenado primeiro por `price` e só depois por `date`, quando chegássemos à parte do índice onde o `price` é maior que 50, teríamos que ainda iterar sobre todas as datas até chegar a 2023. Por esta razão, acreditamos que, para agilizar o `WHERE`, seria mais eficiente termos 2 índices: uma B+ tree na tabela `product` na coluna `price` e uma Hash na tabela `order` para o ano da coluna `date`, já que a condição que nos é dada não é uma comparação exata para a data, mas sim para o ano de 2023. Dessa forma, teríamos o `price` ordenado, o que agiliza as condições de range check, e ao mesmo tempo teríamos a Hash para a condição de igualdade da `date`.
Observação: Não achamos necessária a criação de mais indices uma vez que o resto das operações utilizam os indices implicitos das primary and foreign keys.

```
CREATE INDEX idx_price ON orders (price);
CREATE INDEX idx_year ON orders USING hash (EXTRACT(YEAR FROM date));
```

7.2

Um índice na tabela `product` na coluna `name` do tipo B+ tree unclustered já que iria ordenar os nomes dos produtos (alfabeticamente), o que iria melhorar bastante a condição de range `LIKE`, uma vez que procuramos nomes que começam por "A".
Observação: Não achamos necessária a criação de mais indices uma vez que o resto das operações utilizam os indices implicitos das primary and foreign keys.

```
CREATE INDEX idx_pname ON product (name);
```