EPGaia

Escola Profissional de Gaia

Curso Programador de Informática

Trabalho Prático de Avaliação

Brayan Gabriel Constâncio Amorim Nº 03
Miguel Filipe Dias de Oliveira Nº 17
Mateus Costa Nº 16

Vila Nova de Gaia, 06 de Maio 2025

PESSOAS 2030 PORTUGAL 2030 Cofinanciado pela União Europeia
Os Fundos Europeus mais próximos de si.

Índice

Introdução

Este relatório tem como finalidade apresentar o desenvolvimento de um projeto de programação centrado na criação de uma loja online. Este trabalho, realizado no âmbito da disciplina de TPL, teve como principal objetivo simular um sistema de comércio eletrónico funcional e dinâmico. A loja foi concebida com diversas funcionalidades essenciais, incluindo gestão de stock, sistema de login para clientes, aplicação de cupões de desconto, e integração de diferentes formas de pagamento. Ao longo do projeto, foram aplicados conhecimentos técnicos adquiridos sobre lógica de programação e estruturação de dados

Cofinanciado por: POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Desenvolvimento

Começamos o projeto planejando como iríamos fazer o código, dividindo o código em etapas, e dividimos uma tarefa para cada um e depois foi juntar tudo e trabalhar em conjunto para finalizar e corrigir o código.

Funcionalidades

Este projeto tem como objetivo simular e experiência de um site simples de compras online, nele nós somos capazes de fazer login e registo, também adicionar produtos ao carrinho, adicionar cupons de desconto, métodos de pagamento, endereço de entrega e finalização de compra. As funcionalidades do nosso código estão abaixo:

1. Registro de Utilizador

```
void registrar() {
    string nome, senha;

    cout << "-REGISTRO!!\n";
    cout << "Nome de usuario: ";
    cin >> nome;
    cout << "Senha: ";
    cin >> senha;

    ofstream escrever("registros.txt", ios::app);
    escrever << nome << " " << senha << endl;
    escrever.close();

    cout << "Registro efetuado com sucesso!!\n\n";
}
```

Esta função permite ao utilizador criar uma conta. Ao preencher o nome e a senha, o programa guarda esses dados num ficheiro chamado "registros.txt". Ou seja, o sistema cria um registo simples do utilizador, que depois poderá ser usado para login.

Cofinanciado por: POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

2. Fazer login do utilizador

```
void fazerlogin() {
    string nome, senha, rnome, rsenha;
    bool encontrado = false;

    cout << "-Informe o Nome e a Senha do Usuario\n";
    cout << "USUARIO: ";
    cin >> nome;
    cout << "SENHA: ";
    cin >> senha;

    ifstream ler("registros.txt");
    while (ler >> rnome >> rsenha) {
        if (rnome == nome && rsenha == senha) {
            encontrado = true;
            break;
        }
    }
    ler.close();

    if (encontrado) {
        cout << "Login realizado com sucesso!!!\n\n";
        menuDoSistema();
    }
    else
        cout << "Erro! Usuario ou senha incorretos!\n\n";
}
```

Aqui, o utilizador introduz o nome e a palavra-passe. O sistema vai verificar no ficheiro "registros.txt" se existe esse registo. Se encontrar, o login é bem-sucedido e o utilizador entra no sistema. Caso contrário, aparece uma mensagem de erro.

3. Adicionar produto ao carrinho (exemplo da placa-mãe)

```
void mostrarPlacaMae() {
    system("cls");
    string opcao;
    cout << "Adicionar o produto ao Carrinho\n\n";
    cout << "1. " << estoquePlacasMae[0].nome
        << " | " << estoquePlacasMae[0].preco << " Euros"
        << " | Em estoque: " << estoquePlacasMae[0].quantidade << "\n";
    cout << "2. Voltar\n";
    cin >> opcao;

    if (opcao == "1") {
        if (estoquePlacasMae[0].quantidade > 0) {
            Produto pMae;
            pMae.nome = estoquePlacasMae[0].nome;
            pMae.preco = estoquePlacasMae[0].preco;
            pMae.quantidade = 1;

            carrinho[totalItensCarrinho] = pMae;
            totalItensCarrinho++;

            estoquePlacasMae[0].quantidade--;

            cout << "Produto adicionado ao carrinho!\n\n";
        }
        else {
            cout << "Produto fora de estoque!\n\n";
        }
    }
    else if (opcao == "2") {
        cout << "Voltando ao menu de componentes...\n\n";
    }
    else {
        cout << "Opção inválida!\n\n";
    }
}
```


Cofinanciado por: POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Esta função mostra ao utilizador uma opção para comprar uma placa-mãe. Se ele escolher a opção “1”, o produto é adicionado ao carrinho e o estoque disponível é reduzido. Se já estiver esgotado, o programa avisa.

4. Aplicar cupom de desconto

```
double aplicarCupom(double val1, string val2) {
    if (val2 == "BRAYAN15") return val1 * 0.85;
    else if (val2 == "MATEUS05") return val1 * 0.95;
    else if (val2 == "MIGUEL10") return val1 * 0.90;
    else {
        return val1;
    }
}
```

Esta função verifica se o cupão inserido é válido. Se for, será aplicado um desconto ao valor total da compra. Por exemplo, o cupom "BRAYAN15" dá 15% de desconto. Se o cupom não for válido, o valor total mantém-se igual.

5. Escolher método de pagamento

```
void escolherMetodoPagamento() {
    string opcao2;
    bool continuar2 = true;

    while (continuar2) {
        cout << "Métodos de Pagamento\n\n";
        cout << "1. Google Pay\n";
        cout << "2. PayPal\n";
        cout << "3. MB Way\n";
        cout << "4. Voltar\n";
        cout << endl;
        cout << "Escolha uma opção: ";
        cin >> opcao2;
        cout << endl;

        if (opcao2 == "1") {
            cout << "\nVocê escolheu Google Pay.\nPagamento processado com sucesso!\n\n";
            continuar2 = false;
            cout << endl;
        }
        else if (opcao2 == "2") {
            cout << "\nVocê escolheu PayPal.\nPagamento processado com sucesso!\n\n";
            continuar2 = false;
            cout << endl;
        }
        else if (opcao2 == "3") {
            cout << "\nVocê escolheu MB Way.\nPagamento processado com sucesso!\n\n";
            continuar2 = false;
            cout << endl;
        }
        else if (opcao2 == "4") {
            cout << "\nVoltando ao menu anterior...\n\n";
            cout << endl;
            return;
        }
        else {
            cout << "\nOpção inválida. Tente novamente.\n\n";
        }
    }
    inserirEnderecoEntrega();
    continuar2 = false;
}
```

O utilizador escolhe aqui o método de pagamento preferido. Pode optar por Google Pay, PayPal ou MB Way. Depois de escolhido, o sistema confirma o pagamento e segue para pedir o endereço de entrega.

Cofinanciado por: POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

6. Inserir endereço de entrega

```
void inserirEnderecoEntrega() {
    string endereco, codigoPostal, pais;

    cout << "==============================\n\n";
    cout << "Insira seu endereço para a compra ser enviada.\n\n";

    cout << "Endereço: ";
    cin.ignore();
    getline(cin, endereco);

    cout << "Código Postal: ";
    getline(cin, codigoPostal);

    cout << "País: ";
    getline(cin, pais);

    cout << "\nEndereco registrado com sucesso!\n";
    cout << endl;
    cout << "Seu item será enviado para:\n";
    cout << endl;
    cout << "Endereço: " << endereco << "\n";
    cout << "Código Postal: " << codigoPostal << "\n";
    cout << "País: " << pais << "\n";
    cout << endl;
    system("cls");
    mostrarRecibo();
    cout << endl;
}
```

Depois do pagamento, o utilizador fornece o endereço onde a encomenda será entregue. O sistema guarda e confirma os dados, e de seguida apresenta o recibo da compra.

7. Mostrar o recibo final da compra

```
void mostrarRecibo() {
    cout << "\n---- Recibo de Compra ----\n\n";

    for (int i = 0; i < totalItensCarrinho; i++) {
        cout << carrinho[i].nome << " | Preço: " << carrinho[i].preco
             << " Euros | Quantidade: " << carrinho[i].quantidade << "\n";
    }

    cout << "\nTotal da Compra: " << totalFinalCompra << " Euros\n";

    if (cupomUsado != "Nenhum" && !cupomUsado.empty()) {
        cout << "Cupom aplicado: " << cupomUsado << "\n";
    }

    cout << "Obrigado pela compra! Volte sempre!\n\n";
    cout << "-----------------------------\n";
    cout << endl;
}
```

No final, o sistema apresenta um recibo com todos os produtos comprados, o total gasto, e se algum cupão foi usado. É uma confirmação final para o utilizador daquilo que comprou.

Cofinanciado por: POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

8. Usabilidade de um Dono de Loja Online.

O processo de adicionar novos produtos e renovar o estoque foi projetado para ser simples, permitindo ao dono da loja online gerenciar facilmente o inventário.. Além disso, a estrutura clara de menus e a interação com os produtos tornam a experiência do administrador mais eficiente.

```
void adicionarProdutoCategoria() {
    string grupo, sub;
    Produto novo;

    bool continuar = true;
    bool continuar2 = true;

    cout << "\n=== Adicionar Produto ===\n";
    cout << "1. Componentes\n";
    cout << "2. Periféricos\n";
    cout << "3. Eletrodomésticos\n";
    cout << "4. Consolas\n";
    cout << "5. Voltar\n";
    cout << "Escolha o grupo da categoria: ";
    cin >> grupo;
    cout << endl;

    if (grupo == "5") {
        continuar = false;
    }

    while (continuar) {

        if (grupo == "5") {
            continuar = false;
            }

        cin.ignore();
        cout << "Nome do produto: ";
        getline(cin, novo.nome);
        cout << "Preço: ";
        cin >> novo.preco;
        cout << "Quantidade: ";
        cin >> novo.quantidade;

        while (continuar2) {

            if (grupo == "1") {
```

```
void renovarEstoqueProduto() {
    int indiceProduto, novaQuantidade;
    string grupo, sub;

    bool continuar = true;
    bool continuar2 = true;

    cout << "\n=== Renovar Estoque ===\n";
    cout << endl;

    cout << "1. Componentes\n";
    cout << "2. Periféricos\n";
    cout << "3. Eletrodomésticos\n";
    cout << "4. Consolas\n";
    cout << "Escolha o grupo da categoria: ";
    cin >> grupo;
    cout << endl;

    while (continuar){

    if (grupo == "1") {
        cout << "1. Placas-mãe\n";
        cout << "2. Processadores\n";
        cout << "3. Voltar\n";
        cout << "Escolha a subcategoria: ";
        cin >> sub;
        cout << endl;

        if (sub == "1") {
            cout << "\n=== Lista de Placas-mãe ===\n";
            cout << endl;
            for (int i = 0; i < 10; i++) {
                if (!estoquePlacasMae[i].nome.empty()) {
                    cout << i + 1 << ". " << estoquePlacasMae[i].nome
                        << " | " << estoquePlacasMae[i].quantidade << " unidades em estoque\n";
                }
            }
            cout << "Escolha o número do produto para renovar o estoque: ";
            cin >> indiceProduto;
```


Cofinanciado por: POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

9. Histórico de compras

A implementação do histórico de compras por sessão garantiu que o cliente pudesse acompanhar suas compras realizadas enquanto estivesse logado. Utilizamos uma estrutura dedicada para armazenar os produtos comprados temporariamente, sem misturar dados entre usuários.

```
void salvarNoHistorico(double totalFinalCompra, const string& cupomUsado);
void mostrarHistoricoCompras();
void limparHistoricoCompras();
void limparCarrinho();
```

```
void mostrarHistoricoCompras() {
    if (totalCompras == 0) {
        cout << "\nNenhuma compra realizada ainda.\n";
        return;
    }

    for (int i = 0; i < totalCompras; i++) {
        cout << "\n---- Compra #" << (i + 1) << " ----\n";
        for (int j = 0; j < historico[i].quantidadeItens; j++) {
            cout << historico[i].itens[j].nome << " | Preço: " << historico[i].itens[j].preco
                 << " Euros | Quantidade: " << historico[i].itens[j].quantidade << "\n";
        }
        cout << "Total: " << historico[i].total << " Euros\n";
        if (historico[i].cupom != "Nenhum" && !historico[i].cupom.empty()) {
            cout << "Cupom aplicado: " << historico[i].cupom << "\n";
        }
        cout << "-----------------------------\n";
    }
}
```

```
void limparHistoricoCompras() {
    for (int i = 0; i < totalCompras; i++) {
        for (int j = 0; j < historico[i].quantidadeItens; j++) {
            historico[i].itens[j].nome = "";
            historico[i].itens[j].preco = 0.0;
            historico[i].itens[j].quantidade = 0;
        }
        historico[i].quantidadeItens = 0;
        historico[i].total = 0.0;
        historico[i].cupom = "";
    }
    totalCompras = 0;
}
```


Cofinanciado por: POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

10. Restaurar o estoque quando o cliente não finalizar a compra.

O estoque foi modificado para ser reserva temporária durante a adição de itens ao carrinho. Quando um produto é adicionado, a quantidade é diminuída no estoque e salva-se o estado atual do estoque no carrinho. Caso a compra não seja finalizada, o estoque é restaurado ao valor inicial, garantindo que os itens não sejam perdidos. Ao finalizar a compra, a diminuição definitiva do estoque ocorre.

Ao adicionar um produto ao carrinho, a quantidade é reduzida do estoque como uma reserva imediata. Se o cliente sair sem finalizar a compra, o estoque é restaurado a partir dos backups, preservando os dados anteriores.

Para evitar restaurar o estoque após uma compra finalizada, é necessário verificar a variável “booleana finalizarCompra”.

Foi identificado que o backup não estava sendo atualizado após a finalização da compra, o que fazia o sistema restaurar o estoque incorretamente para o estado original.

```
void fazerBackupEstoque() {
    for (int i = 0; i < 10; i++) backupPlacasMae[i] = estoquePlacasMae[i];
    for (int i = 0; i < 7; i++)  backupProcessador[i] = estoqueProcessador[i];
    for (int i = 0; i < 10; i++) backupMonitor[i] = estoqueMonitor[i];
    for (int i = 0; i < 10; i++) backupTeclado[i] = estoqueTeclado[i];
    for (int i = 0; i < 5; i++)  backupFrigorifico[i] = estoqueFrigorifico[i];
    for (int i = 0; i < 5; i++)  backupMaqLavar[i] = estoqueMaqLavar[i];
    for (int i = 0; i < 3; i++)  backupPS5[i] = estoquePS5[i];
    for (int i = 0; i < 3; i++)  backupNS2[i] = estoqueNS2[i];
}

void restaurarEstoque() {
    for (int i = 0; i < 10; i++) estoquePlacasMae[i] = backupPlacasMae[i];
    for (int i = 0; i < 7; i++)  estoqueProcessador[i] = backupProcessador[i];
    for (int i = 0; i < 10; i++) estoqueMonitor[i] = backupMonitor[i];
    for (int i = 0; i < 10; i++) estoqueTeclado[i] = backupTeclado[i];
    for (int i = 0; i < 5; i++)  estoqueFrigorifico[i] = backupFrigorifico[i];
    for (int i = 0; i < 5; i++)  estoqueMaqLavar[i] = backupMaqLavar[i];
    for (int i = 0; i < 3; i++)  estoquePS5[i] = backupPS5[i];
    for (int i = 0; i < 3; i++)  estoqueNS2[i] = backupNS2[i];
}

void atualizarBackupEstoque() {
    for (int i = 0; i < 10; i++) backupPlacasMae[i] = estoquePlacasMae[i];
    for (int i = 0; i < 7; i++)  backupProcessador[i] = estoqueProcessador[i];
    for (int i = 0; i < 10; i++) backupMonitor[i] = estoqueMonitor[i];
    for (int i = 0; i < 10; i++) backupTeclado[i] = estoqueTeclado[i];
    for (int i = 0; i < 5; i++)  backupFrigorifico[i] = estoqueFrigorifico[i];
    for (int i = 0; i < 5; i++)  backupMaqLavar[i] = estoqueMaqLavar[i];
    for (int i = 0; i < 3; i++)  backupPS5[i] = estoquePS5[i];
    for (int i = 0; i < 3; i++)  backupNS2[i] = estoqueNS2[i];
}
```


Cofinanciado por: POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Constrangimentos e Soluções encontradas

1. Sessão de Login

Problema

Não estávamos conseguindo salvar os dados do utilizador ao registrar, então usava um usuário pré-criado no código.

Resolução

Pesquisei sobre gravação em arquivos e contei com ajuda de colegas do projeto.

Observação

Ainda parece que só é possível registrar uma pessoa por vez; ao registrar uma nova, a anterior desaparece (provavelmente está sobrescrevendo o arquivo em vez de adicionar).

2. Cupom de Desconto

Problema

O cupom de desconto não aplicava o valor corretamente ao total da compra.

Resolução

Percebemos que o cálculo era feito antes de aplicar o cupom. Ajustei a ordem das operações e revisei a fórmula para subtrair o valor correto.

3. Método de Pagamento

Problema

Não conseguimos simular a experiência completa de cada forma de pagamento (Google Pay, PayPal, MB Way) no código.

Resolução

Optamos por simular o processo usando mensagens/frases representando cada etapa do pagamento. Assim, o usuário visualiza o que aconteceria sem precisar de integração real com sistemas de pagamento.

Observação

Pretende-se simular, fielmente, a experiência de cada forma de pagamento

4. Utilização do Programa

Cofinanciado por: POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Problema

Como se trata de uma loja online, simular a experiência de um site em um programa terminal é difícil. O código não abre novas páginas, então tudo acontece em sequência e fica corrido.

Resolução

Aceitamos essa limitação e buscamos oferecer a melhor experiência possível com o que sabemos até agora. Usei técnicas como múltiplas opções bem organizadas, um código limpo, separado por funções, e poucas categorias de produtos para facilitar a navegação.

5. Funcionalidade de um Estoque e Carrinho

Problema

Inicialmente, não tínhamos uma ideia concreta de como implementar um sistema de estoque em uma loja online, nem como funcionava exatamente um carrinho de compras.

Resolução

Fizemos pesquisas sobre sistemas de estoque e carrinho em lojas virtuais. Com base nisso, criamos um estoque que se atualiza automaticamente quando um produto é adicionado ao carrinho, diminuindo a quantidade disponível. Também desenvolvemos um carrinho funcional onde os produtos escolhidos são armazenados temporariamente até a finalização da compra.

6. Funcionalidade de adicionar um novo produto por meio da utilização do código (Dono da loja online)

Problema

Inicialmente, o sistema apresentava um erro de duplicação de produtos no estoque. Isso ocorria porque os arrays de produtos não estavam verificando corretamente os espaços vazios antes de adicionar novos itens, resultando na repetição de produtos em múltiplos índices.

Resolução

Fizemos uma alteração no código para garantir que a busca por espaços vazios nos arrays de estoque fosse realizada antes de adicionar um novo produto. Ao verificar corretamente os índices onde o campo “nome” estava vazio, o sistema passou a adicionar os novos produtos apenas nos espaços disponíveis, corrigindo o problema de duplicação.

7. Renovar o estoque dos produtos

Problema:

O processo de renovação do estoque de produtos estava suscetível a erros, como a falta de um mecanismo claro para selecionar o produto desejado e a atualização inadequada das quantidades. Sem uma abordagem direta, o administrador poderia acabar tendo dificuldades para identificar e adicionar os produtos corretamente.

Solução:

A solução foi criar uma funcionalidade que permite ao administrador selecionar diretamente o produto que deseja renovar o estoque. Isso foi feito por meio de um menu de opções, onde o dono da loja escolhe facilmente o produto e atualiza a quantidade de forma rápida e precisa, sem confusão ou erros de seleção.

8. Histórico de Compras

Problema:

O sistema de compras foi implementado com uma estrutura de carrinho temporária, e ao final da compra, os dados são limpos para reutilização. Para evitar que um cliente veja os dados de outro, foi adicionado um método que limpa o histórico de compras ao sair da conta. No entanto, isso causou um problema: quando o cliente realiza compras, sai da conta e depois entra novamente, o histórico dessas compras **não aparece mais**, pois ele foi apagado durante o logout. Isso impede o cliente de visualizar registros anteriores de compras realizadas.

Solução:

A solução adotada foi criar uma estrutura de histórico de compras que armazena os dados enquanto o cliente estiver logado. A função "limparHistoricoCompras()" foi implementada para ser chamada **somente quando o cliente sai da conta**, garantindo que o histórico esteja disponível durante toda a sessão. Dessa forma, o cliente pode visualizar suas compras normalmente enquanto estiver logado, e o sistema ainda garante a segurança e privacidade entre diferentes logins, ao limpar o histórico apenas no logout.

Cofinanciado por: POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

9. Controle de Estoque Existente

Problema:

O sistema reduzia diretamente o estoque ao adicionar produtos ao carrinho, mesmo sem o cliente finalizar a compra.

Além disso, ao sair da conta, o estoque era restaurado para o estado inicial (backup original), mesmo após uma compra ser finalizada. Isso causava inconsistência nos dados de estoque, revertendo corretamente quando o cliente desistia, mas incorretamente quando finalizava a compra.

Os mesmos erros estavam dando para os novos produtos e o novo estoque que são adicionados pelo "dono" na usabilidade do código.

Solução:

Foi implementado um controle com variáveis de backup para restaurar o estoque apenas se a compra não for finalizada.

Adicionou-se uma verificação com "bool finalizarCompra" para distinguir os dois casos.

Também foi corrigido o erro no backup: agora, ao finalizar a compra, o backup é atualizado com o novo estado do estoque, garantindo consistência nas futuras sessões.

Foi preciso corrigir o estoque dos novos produtos que o "dono" adiciona, atualizando o estoque de cada novo produto por meio da função "atualizarBackupEstoque();" para ficarem atualizados como o estoque dos produtos já existentes. Também foi necessário fazer o mesmo passo na função "void renovarEstoqueProduto()", pois ela também não se atualizava e nem guardava o estoque que foi adicionado nos produtos existentes como no novos produtos.

Conclusão

Este trabalho permitiu-nos aplicar de forma prática os conhecimentos adquiridos, através da criação de uma loja online funcional em ambiente de terminal. Apesar de ainda haver espaço para melhorias, especialmente na organização e dinamismo do código, consideramos que foi uma experiência enriquecedora. O trabalho em grupo foi essencial para alcançar um resultado mais completo, promovendo a cooperação e a partilha de ideias.