U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

BUY NOW

João Andrade, up201905589 | Sérgio Estevão, up201905680 | Sofia Germer,up201907461
Turma 2MIEIC03

PROBLEMA - Venda de Produtos Online

- Registar clientes
- Transações
- Repor stock da loja online caso este seja inferior a um stockMin
  - Procurar loja física com maior stock
  - Caso não encontre nenhuma, compra a fornecedor
- Aceitar diferentes formas de pagamento

Fonte: Google Images, https://pxhere.com/en/photo/1440159

SOLUÇÃO

MENU

- 3 modos de utilização: Clients mode, Manager mode, Option to exit;
- 3 opções no modo Client: Initial Info, More About Us, Option to exit
-3 opções no modo Manager: Change Profict Margin, Check Shop Statistics, Go Back

SUPPLIER

-Informação sobre o preço de aquisição de produtos
-A partir deste preço, multiplicando por um valor (profitMargin) a definir pelo Manager será definido o preço de cada produto

SHOP

-Gere clientes e produtos
-Cria vetor de objetos da Real Shop através da leitura de files todas as lojas físicas dessa loja

REAL SHOPS

- Informações sobre as lojas, visualizadas caso selecionada opção More About Us: adress, postal code, telephone
- Vetor de produtos, que caso seja necessário e possível repõe stock na loja online

CLIENT

-Dados: clientName, NIF
- Pedindo email e utilizando ID sequencial, é dada a opção ao cliente de se tornar membro da **CLASSE REGISTERED CLIENT** (derivada publicamente da classe Client)

PRODUCTS

- Dados: productName, price, stock

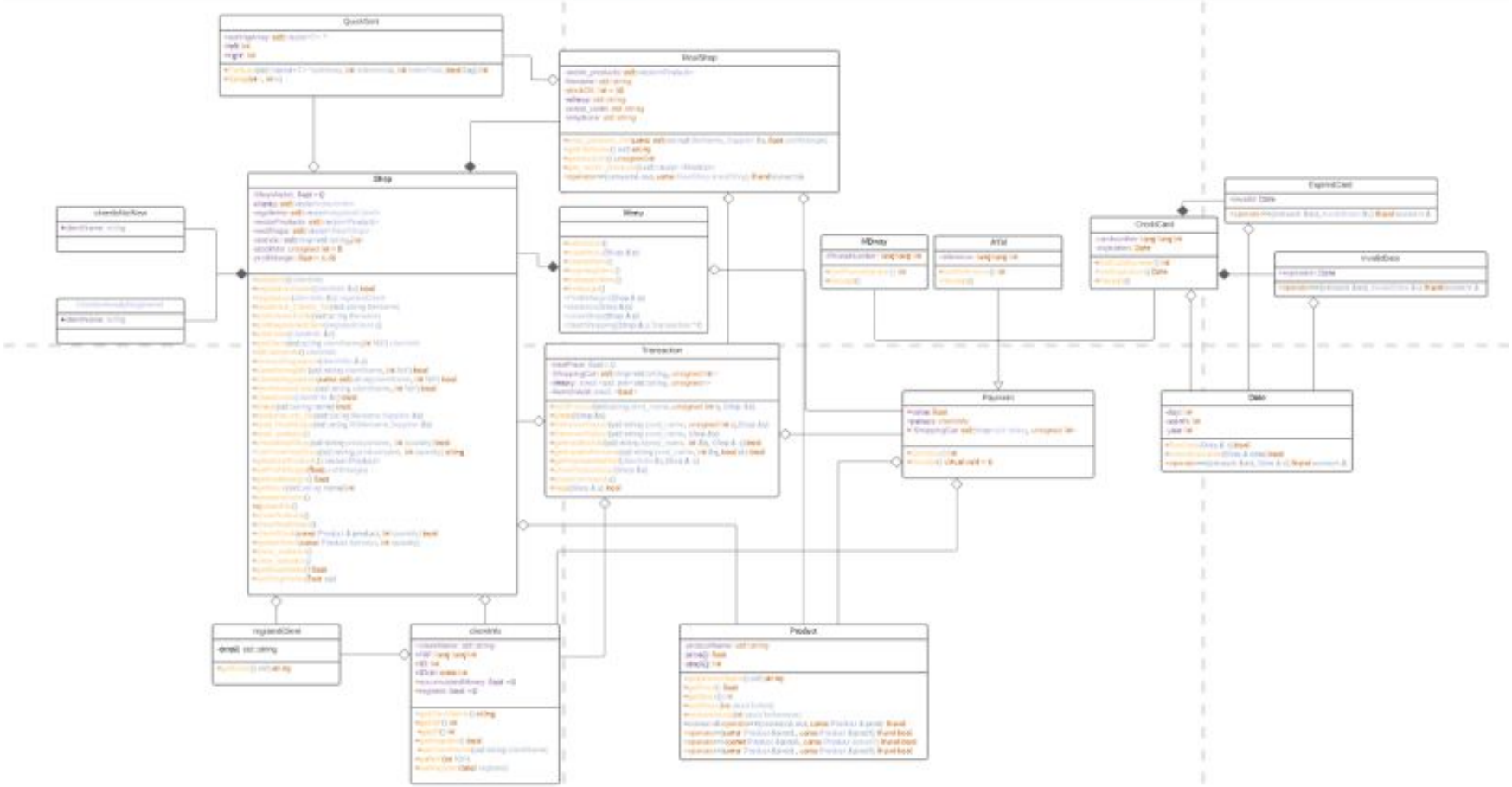
TRANSACTIONS

- Dados: preço total da transação, carrinhos de compras
- Constituída por um loop que permite ao cliente gerir as compras (ex: adicionar/ remover produtos)

PAYMENT

-Chamada dentro da classe Transactions, servindo para efetuar o pagamento da compra
-3 modos de pagamento: MBWAY, ATM, Credit Card

DIAGRAMA DE CLASSES



FICHEIROS

dados:
-nome loja online
-morada
-código-postal
-contacto telefónico

dados:
-nome produto;
-stock nessa loja

- Nota: existe um destes files para cada loja físisca

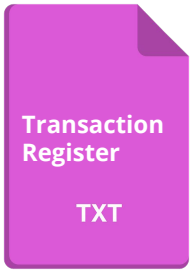
dados:
-nome do produto
-preço de compra a fornecedor

dados:
all clients
-nome do cliente
-NIF
- bool (0 if not registered, 1 if registered)

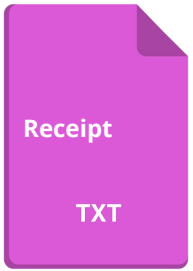
---

registered clients
- ID
-email

dados:
-lucro da loja
- percentagem de compra de cada produto
- quantidade vendida de cada produto



dados:
-lista de todos os talões



dados:
-lista de todos os produtos comprados e a sua quantidade
-nome do cliente, NIF, método de pagamento (telemóvel/referência/cartão de crédito)

TRATAMENTO DE EXCEÇÕES

1	**CLIENT ALREADY REGISTERED**	• Exceção lançada quando o Cartão de Crédito tem data expirada
2	**CLIENT IS NOT NEW**	• Exceção lançada quando o Cliente já existe
3	**EXPIRED CARD**	• Exceção lançada quando o Cartão de Crédito tem data expirada
4	**INVALID DATE**	• Exceção lançada quando a data introduzida pelo cliente

CRUD

- ★ CREATE
- ★ READ
- ★ UPDATE
- ★ DELETE

Create Files (completa):

- ❖ Cria file Receipt para cada compra e adiciona esse talão ao file Transaction
  - ➢ void MBway::Receipt()
  - ➢ void ATM::Receipt()
  - ➢ void CreditCard::Receipt()

Read files(completa) :

- ❖ Ler file dos clientes iniciais quando a loja é inicializada
  - ➢ void Shop::readInitial_Clients_file(const string &fileName);
- ❖ Ler file das real_shops e criamos objeto
  - ➢ void Shop::read_statistics()
- ❖ Ler file dos produtos e criamos objetos da classe Product que serão adicionados ao vetor de produtos
  - ➢ void Shop::readproducts_file(const string& fileName, Supplier &s) (da loja online)
  - ➢ void read_products_file(const string& fileName, Supplier &s, float profitMargin) (para cada loja real)
- ❖ Ler cada file das lojas reais e criar objeto da classe RealShop para cada uma
  - ➢ void Shop::read_RealShops(string RS_filename, Supplier &s)

Update files (completa):

- ❖ Update das transactions : após cada compra se for necessário dar restock na loja online
  - ➢ void Shop::updateStock():
- ❖ Update dos clientes
  - ➢ void Shop::addClientsToFile(const string &filename);
- ❖ Dá update ao file dos produtos da loja online, ao file de produtos de cada loja Real e ao file de estatísticas
  - ➢ void Shop::updateFiles();

Delete files:

- ❖ Não tivemos necessidade de criar nenhuma função que desse delete diretamente a um file, uma vez que quando fazemos o update do file reescrevemos o file por inteiro

Listagem

```
--------------------------------PRODUCTS--------------------------------
acucar------------------------------------------------------------ 0.29€
agua-------------------------------------------------------------- 0.87€
amendoins--------------------------------------------------------- 3.34€
arroz------------------------------------------------------------- 1.45€
banana------------------------------------------------------------ 1.74€
batatas----------------------------------------------------------- 2.03€
batatas-fritas---------------------------------------------------- 0.58€
bolachas---------------------------------------------------------- 2.32€
cafe-------------------------------------------------------------- 1.88€
carne------------------------------------------------------------- 2.03€
cerveja----------------------------------------------------------- 2.18€
cha--------------------------------------------------------------- 0.87€
courgette--------------------------------------------------------- 1.45€
couve------------------------------------------------------------- 1.88€
```

void Shop::showProducts()

Display do produto e do seu preço

```
      Shop <---------------- shop_braga
            30 x ovos


Date: Sat Nov 21 18:41:31 2020

------------------------------------------------------------

      Shop <---------------- shop_braga
            30 x cha


Date: Sat Nov 21 20:24:12 2020
```

-Lista de todas as transações da loja

```
--------------------------------STATISTICS--------------------------------
Profit:   534.296€

acucar                |  1% (sold:33)
agua                  ||  2% (sold:64)
amendoins             |  1% (sold:40)
arroz                 |  1% (sold:48)
banana                 0% (sold:0)
batatas                0% (sold:0)
batatas-frita          0% (sold:26)
batatas-fritas         0% (sold:0)
bolachas              |  1% (sold:34)
cafe                  ||  2% (sold:56)
carne                  0% (sold:23)
cerveja               |  1% (sold:27)
cha                   ||  2% (sold:67)
courgette              0% (sold:24)
couve                  0% (sold:26)
farinha               |  1% (sold:32)
laranja               |  1% (sold:35)
leite                 |  1% (sold:32)
mangas                ||||  4% (sold:123)
maracuja              |  1% (sold:35)
massa                 |||||  5% (sold:146)
melancia              |||  3% (sold:86)
nabos                 |||  3% (sold:87)
nozes                 |||  3% (sold:88)
ovos                  ||||  4% (sold:117)
pao                   ||||||||||||||||||||||||||||||||||||  36% (sold:970)
peixe                 ||  2% (sold:78)
pipocas               |||  3% (sold:92)
presunto              |  1% (sold:47)
sumo                  ||||||||  8% (sold:215)
vinho                 |  1% (sold:33)
```

void Shop::show_statistics()

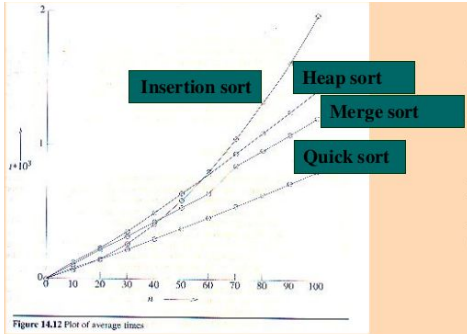
-Dispaly do profit da loja (balanço entre dinheiro recebido e dinheiro gasto em compras ao fornecedor)
-Percentagem de venda de um produto em relação às vendas totais
-Quantidade vendida de cada produto

```
------------------------------------------------------------------------


          Receipt
------------------------------------------------------------------------

Prod.  |  Quant.
mangas  :  32
pao  :  31

Discount:

RegistedClient: -15%
DiscountedValue: -17.3783€


TOTAL: 98.4768€

------------------------------------------------------------------------
Name:    Sofia G
NIF:     452365896
Payment Method: Cartao de credito
Card number: 852365459
Sat Nov 21 22:12:27 2020
```

-Lista de todos os talões

Ordenação - Quick Sort

- Para ordenar alfabeticamente o vetor dos produtos
  - Uma vez que o vetor de Produtos tem mais de 20 elementos, escolhemos o método de quick sort devido a uma questão de eficiência
  - Estando os produtos ordenados alfabeticamente, torna- se mais fácil para o cliente encontrar o produto que procura.
    - **overload do operador < e overload do operador >** que comparam 2 produtos atendendo a sua ordem alfabética
- Para ordenar alfabeticamente o vetor das estatísticas
  - Quando o gerente pretender ver as estatísticas, estando elas ordenadas alfabeticamente torna- se mais fácil encontrar as informações de cada produto
    - **overload do operador < e overload do operador >** que comparam um < pair<string,int>> atendendo à quantidade(int) vendida do produto com nome dado pelo primeiro parametro do map (string)



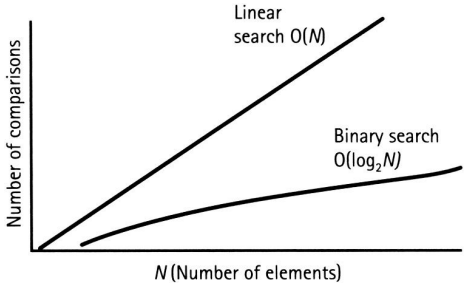
Fonte: Sahni "Data Structures, Algorithms and Applications in C++

```
bool operator<(const Product &prod1, const Product &prod2){
    return prod1.getproductName() < prod2.getproductName();
}

bool operator>(const Product & prod1, const Product & prod2)
{
    return prod1.getproductName() > prod2.getproductName();
}
```

```
bool operator < (const pair<int, int> &a, const pair<int, int> &b) {
    return a.second < b.second;
}

bool operator > (const pair<int, int> &a, const pair<int, int> &b) {
    return a.second > b.second;
}
```

Procura - Binary Search

```
int Shop::special_Binary_search(const vector<Product> prod, string product_name) const {
    int left = 0, right = prod.size() - 1;
    while (left <= right)
    {
        int middle = (left + right) / 2;
        if (prod[middle].getproductName() < product_name)
            left = middle + 1;
        else if (product_name < prod[middle].getproductName())
            right = middle -1;
        else
            return middle;
    }
    return -1;
}
```

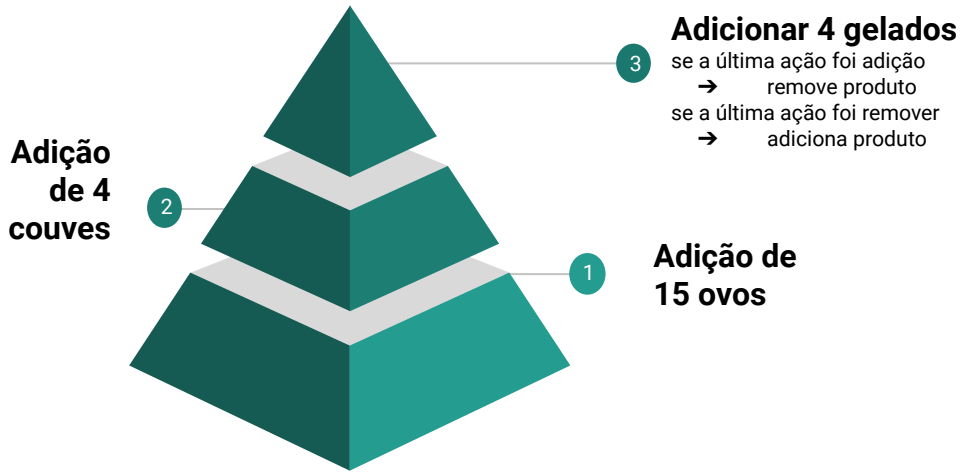


Fonte: Google Images , https://www.techtud.com/sites/default/files/public/user_files/tud39880/linearSearch%20vs%20binary%20search%20diagram_0.jpg

- Como os nossos vetores de Produtos (tanto da online shop como para cada loja real) estão ordenados, escolhemos o método de procura Binary Search, por uma questão de eficiência
- Uma vez que o nosso vetor de Produtos é um vetor de objetos (cada um com o respetivo nome, preço e stock), para utilizarmos Binary Search, criamos uma função especial de pesquisa que procura um Produto pelo seu nome.

DESTAQUE - Função undo()

-Escolhemos apresentar a função void undo() da classe transaction, uma vez que consideramos se destacar devido à sua originalidade
-A Função Undo() da class Transaction permite ao cliente voltar atrás na sua ação mais recente, através do uso conjunto de 2 stacks que guardam as ações do cliente de forma cronológica.
- stack <bool> RemorAdd : guarda false se a ação foi de remoção e true se foi de adição
- stack <pair<Product, unsigned>> history : guarda os dados associados às ações (o produto e a quantidade a adicionada/removida)



DIFICULDADES ENCONTRADAS

1. Sendo o nosso primeiro projeto de programação enquanto um grupo de 3 estudantes, surgiu , naturalmente, uma dificuldade maior em gerir as diferentes versões e conciliar ideias.

2. Após a realização deste projeto, consideramos ter aprofundado os nossos conhecimentos na linguagem C++ bem como nos seguintes tópicos: herança e polimorfismo, gestão de dados através de ficheiros, tratamento de exceções e algoritmos de pesquisa e ordenação.

3. Por último, concluímos que foi um projeto que nos fez desenvolver tanto as hard-skills bem como as soft-skills, nomeadamente o trabalho em equipa.