# CADEIA DE FARMÁCIAS

## Algoritmos e Estruturas de Dados

Relatório da 1ª Parte do Projeto

23/11/2018

Grupo 3 - Turma 5

Márcia Teixeira – 201706065, up201706065@fe.up.pt; Pedro Esteves – 201705160, up201705160@fe.up.pt; Rita Mota - 201703964 , up201703964@fe.up.pt

# Índice

# Descrição do Tema

O objetivo do trabalho desenvolvido é criar uma aplicação que permita a gestão eficiente de uma cadeia de farmácias.

Uma cadeia é composta por farmácias, funcionários e clientes. Cada farmácia tem um conjunto de produtos que pode vender, cada funcionário exerce um cargo numa determinada farmácia e cada cliente tem um historial de vendas associado.

# Implementação / Classes implementadas

## CadeiaFarmacias

Um objeto da classe CadeiaFarmacias representa uma cadeia de farmácias, armazenando todas as farmácias, clientes e funcionários (organizados em vetores). Uma cadeia de farmácias é caracterizada pelo seu nome. Esta classe contém principalmente métodos para alterar ou obter os seus atributos e adicionar, remover e ordenar as farmácias, funcionários e clientes.

## Data

Um objeto da classe Data corresponde a uma data, caracterizada pelo dia, mês e ano. A data pode tanto ser definida pelo utilizador como corresponder à data atual do sistema (usando a biblioteca ctime).

## Hora

Um objeto da classe Hora corresponde a uma data, caracterizada pela hora, minutos e segundos. A hora pode tanto ser definida pelo utilizador como corresponder à hora atual do sistema (usando a biblioteca ctime).

## Farmacia

Um objeto da classe Farmacia corresponde a uma farmácia, caracterizada por um nome, morada, gerente e diretor técnico. Cada farmácia tem uns certos produtos em stock (map produtosVender, com os produtos e as respetivas quantidades disponíveis), e um historial de vendas (vetor vendas). Esta classe contém métodos para alterar e obter os seus atributos, obter informações sobre estes (por exemplo, número de vendas ou total ou em datas específicas) e adicionar, remover e ordenar as vendas.

## Pessoa

Um objeto da classe Pessoa corresponde a uma pessoa, caracterizada pelo seu nome, morada e número de contribuinte. Esta classe contém métodos para alterar e obter os seus atributos.

## Funcionario

A classe Funcionario é derivada publicamente da classe Pessoa. Um objeto desta classe representa um funcionário, que tem como atributos, para além dos da classe base, o seu salário, farmácia em que trabalha e cargo exercido. Esta classe contém principalmente métodos para alterar e obter os seus atributos.

## Cliente

A classe Cliente é derivada publicamente da classe Pessoa. Um objeto desta classe representa um cliente, que tem como atributos, para além dos da classe base, o seu historial de compras. Esta classe contém principalmente métodos para alterar e obter os seus atributos, bem como informações sobre os mesmos (por exemplo, número de vendas).

## Produto

Um objeto da classe Produto representa um produto, caracterizado pelo seu código, nome, preço e descrição. Um produto pode ser ou não passível de ter receita e, caso seja, pode ou não ser possível a sua venda sem receita, e a sua venda com receita tem associado um valor de comparticipação. Esta classe contém métodos para alterar e obter os seus atributos.

## Receita

Um objeto da classe Receita representa uma receita, caracterizada pelo seu número, pelo nome do médico que a prescreveu e pelo cliente ao qual a receita foi prescrita. A receita tem produtos receitados em determinada quantidade. Esta classe contém métodos para alterar e obter os seus atributos e informação sobre os mesmos.

## Venda

Um objeto da classe Venda representa uma venda, caracterizada pelo seu código, data e hora. Uma venda tem associado um cliente que realizou a venda e pode ter também associada uma receita. Uma venda tem vários produtos vendidos, numa determinada quantidade e com determinados valores de IVA e comparticipação (este valor será 0 se o produto não for passível de receita ou se, caso seja e seja possível a sua venda sem receita, não conste na receita associada à venda). Esta classe contém métodos para alterar e obter os seus atributos e informações sobre os mesmos, bem como um método para adicionar produtos à venda.

Para além dos métodos referidos, todas as classes têm o seu overload do operador <<, usado para exportar a cadeia de farmácias para um ficheiro, e a maioria das classes tem uma função usada como comparador, necessária para ordenar objetos do seu tipo. Os algoritmos usados para ordenar encontram-se implementados no ficheiro util.h.

# Diagramas UML

Os diagramas de UML realizados foram diagramas de classes. Foram realizados dois diagramas, um no início do projeto para organizar a estrutura que iríamos implementar, e outro no final, para ilustrar a estrutura implementada e comparar as diferenças com o objetivo inicial. O primeiro diagrama foi realizado com recurso à ferramenta Visual Paradigm. Ambos os diagramas vão ser também enviados em anexo devido à possível dificuldade de visualização dos mesmos neste relatório.

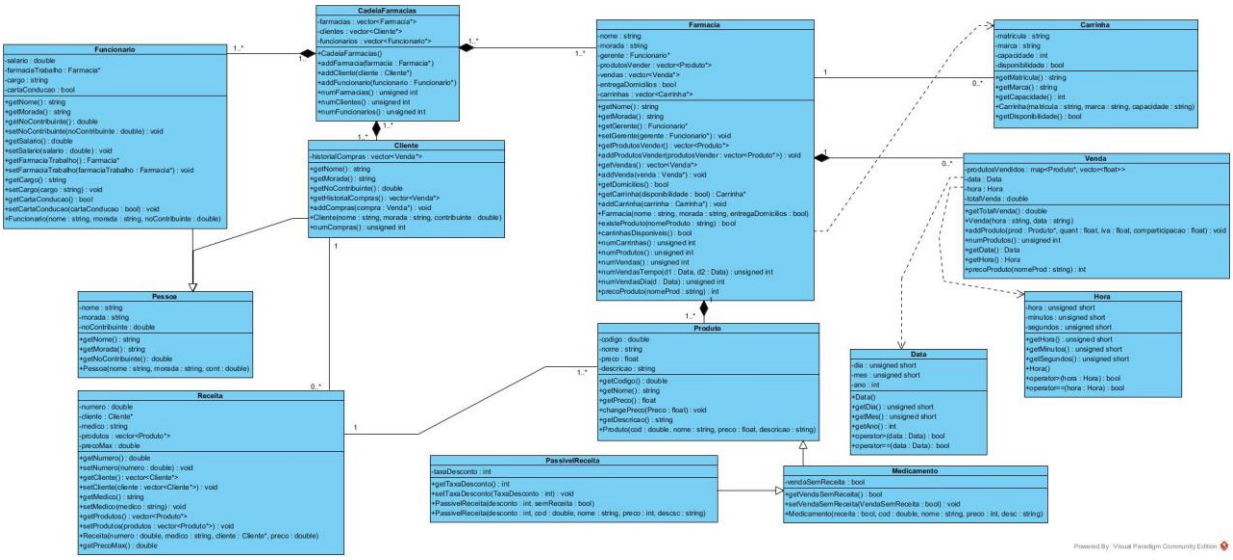


***Diagrama Inicial***

No último diagrama realizado deparamo-nos com um grave problema da ferramenta utilizada: ao converter o código C++ para o diagrama de UML, não foram incluídos os tipos de variáveis, de retorno de métodos e ainda de parâmetros de funções. Uma vez que este tipo de erros aconteceram, utilizamos outra ferramenta, o Umbrello, para gerar o uml.

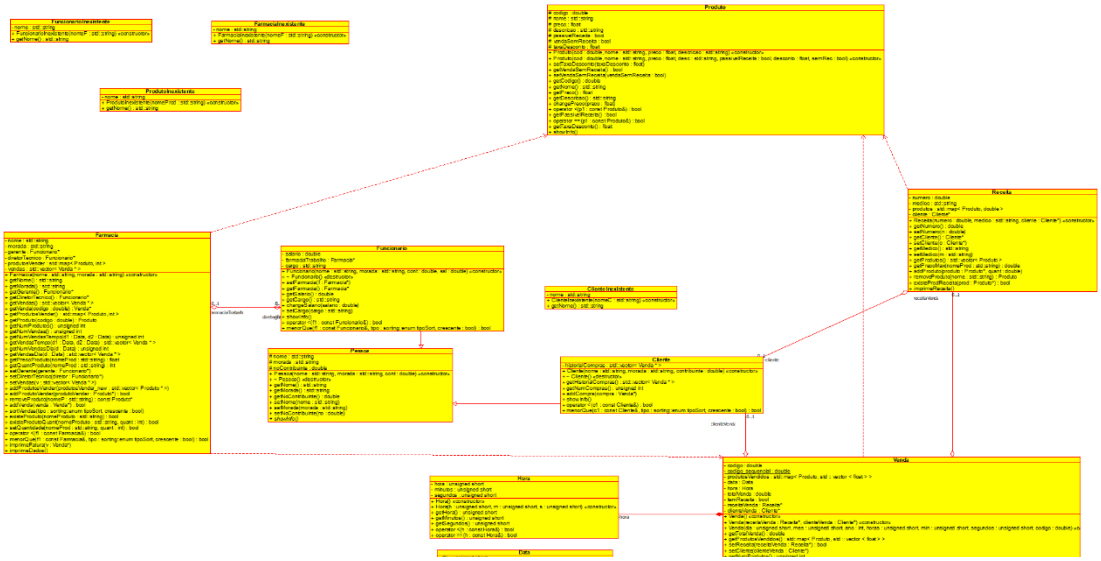

***Diagrama Final***

# Dificuldades

Encontramos algumas dificuldades nos includes dos header files, uma vez que muitas classes eram incluídas noutras que também as incluíam.

Outras dificuldades que encontramos foi na exportação e importação dos dados, principalmente nesta última, uma vez que trabalhamos com muitos apontadores para objetos. Por exemplo, na classe Farmacia, existem apontadores para objetos do tipo Funcionario, nomeadamente o gerente e o diretor técnico da farmácia em causa. No momento de importar estes dados, como ainda não tínhamos importado os funcionários, tornou-se complicado gerir esta informação. Uma simples solução seria importar primeiro os objetos da classe Funcionario, mas, como estes também incluíam apontadores para objetos do tipo Farmacia, que gerou a dificuldade referida em cima, não seria eficiente fazer isto. Para além disso, tivemos também inicialmente dificuldades nos métodos de ordenação para as várias classes usando diferentes critérios.

Por fim, uma das maiores dificuldades prendeu-se com a gestão do tempo, especialmente na fase final do trabalho. Esta má gestão do tempo deveu-se ao facto de termos encontrado vários erros no nosso código durante a implementação do mesmo, o que impediu de avançar no desenvolvimento do trabalho. Partes importantes foram deixadas para as últimas duas semanas, o que resultou nalguns bugs no trabalho final. Uma das partes do trabalho mais afetada foi a interface, devido a ter sido feita num espaço de tempo muito reduzido.

# Casos de aplicação

Ao começar o programa, o utilizador tem duas opções (para além da opção de sair): criar uma nova cadeia de farmácias ou importar uma cadeia já existente num ficheiro. Aí tem mais opções: ver, adicionar, remover e ordenar (por vários critérios e ordem crescente ou decrescente) farmácias, funcionários ou clientes. Pode também escolher ver os dados da cadeia de farmácias, gerir uma farmácia, funcionário ou cliente específico.

Na gestão de uma farmácia é possível ver os dados da farmácia, mudar o gerente ou o diretor técnico, consultar e alterar o stock (adicionar, alterar quantidades e remover produtos), ver historial de vendas (completo ou entre duas datas específicas) e ordená-lo. Pode também realizar uma venda. Esta venda pode ou não ter associada uma receita; caso tenha, são pedidos os dados da mesma.

Na gestão de um funcionário é possível ver os dados do funcionário, alterar a farmácia em que trabalha, cargo, salário ou morada.

Na gestão de um cliente é possível ver os dados do cliente, ver o seu historial de compras e alterar a sua morada.

# Contribuição dos Membros

A implementação das classes CadeiaFarmacias, Data, Hora, Farmacia, Pessoa (e as classes derivadas Funcionario e Cliente), Produto, Receita e Venda e dos métodos do ficheiro util.h foi dividida de forma igual pelo Pedro Esteves e pela Márcia Teixeira. A documentação Doxygen e o diagrama UML inicial foram feitos pelo Pedro Esteves. O relatório foi feito por Márcia Teixeira. O menu/interface foi implementado pela Rita Mota.