**Relatório**

**Linguagens de Programação II**

**Aluno: Victor Abreu-18841**

**Professor: Luís Ferreira**

**Engenharia de Sistemas Informáticos**

Barcelos, junho, 2020

Resumo

Infect Manager é um programa desenvolvido na linguagem C# para manipulação e tratamento de dados acerca de pessoas infetadas geridos por possíveis diferentes hospitais, cada um com a sua correspondente base de dados. No mesmo é possível registar fichas de utentes, altera-las, guardar os dados em ficheiros do tipo binário, visualizar as fichas contidas na base de dados e disponibiliza também diversas opções de consulta de casos por idade, região, sexo e etc...

No âmbito da disciplina de Linguagens de Programação II o projeto visa explorar diferentes técnicas te tratamentos de dados em C# e tirar proveito do que a presente linguagem de programação pode oferecer, deste modo o programa foi desenvolvido para Consola Windows e oferece interação com utilizador a partir de menu simples e objetivo para melhor visualização das ferramentas e resultados que a experiência com o mesmo pode oferecer.

Índice

[1. Introdução 4](#_Toc38824004)

[1.1. Contextualização 4](#_Toc38824005)

[1.2. Motivação e objetivos 4](#_Toc38824006)

[1.3. Estrutura do Programa 4](#_Toc38824007)

[2. Implementação 5](#_Toc38824008)

[2.1. Descrição do problema 5](#_Toc38824009)

[2.2. Solução 5](#_Toc38824010)

[3. Conclusão 7](#_Toc38824011)

[3.1. Lições aprendidas 7](#_Toc38824012)

[3.2. Apreciação final 7](#_Toc38824013)

# Introdução

## Contextualização

O projeto Infect Manager tem como tema principal o tratamento de dados acerca de pessoas infetadas, e no tratamento de dados das mesmas. O tema foi escolhido e pensado de acordo com a situação atual no meio global no que se refere a pandemia provocada pelo novo vírus Covid-19. Assim com uma contextualização na presente situação mundial, o desenvolvimento do projeto teve como objetivo transportar ao máximo o conceito de produzir de acordo com as necessidades atuais e deste modo fornecer ferramentas e resultado coerente com a envolvente atual.

## Motivação e objetivos

O presente trabalho em linguagem C# têm como objetivo explorar o desenvolvimento de um programa de manipulação de dados funcional, coerente, simples e eficaz.

Foram abordadas técnicas de desenvolvimento por camadas N-Tier para modularização do código e melhor organização de classes e de seus respetivos métodos. Uso de collections, especificadamente Listas Genéricas como estrutura de dados. Manipulação de ficheiro binário, nomeadamente gravação e leitura da estrutura de dado em ficheiro binário. Uso de herança por classes e implementação de exceções a serem usadas em todo o programa.

De modo em geral, foram abordadas técnicas necessárias e de caráter de base para o bom funcionamento e organização do código.

## Estrutura do Programa

A estrutura geral do código está dividida por 4 camadas:

* Front End:

Camada inicial e de interação com o utilizador ao qual localiza-se a função “main” do programa.

Na presente camada é feita a chamada ao método de menu de apresentação que faz ligação com os diferentes tipos de menu de acordo com o tipo de utilizador, que por sua vez faz ligação com o menu principal do programa ao qual distribui as presentes funcionalidades principais de interação.

Tais interações correspondem a visualização de fichas de utentes, inserção de novos utentes na base de dado e consultas de utentes por diferentes opções de escolha.

* Business Layer:

Camada de objetos a serem trabalhados pelo programa em geral.

Nesta camada estão contidas duas classes ao qual uma herda atributos da outra. A classe “pai” de pessoas possui o objeto de negócio pessoa, serializável para tratamento em ficheiro binário, ao qual possui os atributos comuns de uma pessoa, como nome, idade, sexo e região de morada. A classe “filho” de utente, também serializável, herda os atributos de pessoa e adiciona novos atributos exclusivo de um utente como o estado de infeção, se está ativo ou não, e o tipo de infeção.

Assim o programa possui dois objetos de negócio que irão ser tratados em nos métodos e regras e na camada de Dados.

* Business Rules Layer:

Camada de regras de negócio onde estão definidos todos os métodos de tratamento dos objetos de negócio.

Estão implementados nessa camada métodos de inserção e inativação de utentes, métodos de menu de escolha, e métodos do tipo procedimento de pesquisa de utentes por diferentes opções de consulta.

Nesta camada é feita a “ponte” entre os objetos de negócio, a interação com utilizador e a camada de dados.

* Data Layer:

Camada de dados do programa, ao qual possui a estrutura de dados utilizada no projeto, e os métodos de inserção de dados, no caso objetos, na estrutura, e de gravação e leitura de dados em ficheiro binário.

Métodos com adicionar um utente na lista e inativar o mesmo são implementados nessa camada e são utilizados em funções da camada de regras de negócio, e métodos que gravam e leem a estrutura de dados em um ficheiro binário.

Deste modo todo o tratamento de registo de dados esta segura e bem, estruturado pois os métodos passam sempre antes por validações e não oferecem qualquer comprometimento dos mesmos.

# Implementação

## Descrição do problema

**Problemáticas enfrentadas no projeto**:

* Qual seria a melhor abordagem para organização do código e sua modularização.
* Qual a estrutura de dados a ser utilizada para tratamento de dados perante ao tema e ao objetivo do programa.
* Tipo de classes e sua herança para melhor organização das mesmas.
* Métodos eficazes para pesquisa, inserção e validação de dados.
* Gravação em ficheiro de modo a manter os dados apos o encerramento do programa.

## Solução

Para a modularização do código foi pensado na organização do mesmo por camada do tipo N-tier, de modo a separar devidamente a diferente parte dos códigos de modo e privar os dados a partir de regras de validação, para manter os mesmo seguros e sem risco de corrupção, e para distribuir o processamento do programa devidamente para torna-lo mais eficaz e leitura simples de código.

A estrutura de dados utilizada foi do tipo *Generic List* pois trata-se de uma estrutura de dados semelhantes aos arrays, devido a possuir posição de dados e poder albergar diferentes tipos de dados, deste modo temos a facilidade de tratamento como em arrays por pesquisa de índice e ao mesmo tem a complexidade necessária para guarda diferentes tipos de dados e inclusive objetos.

A classe Pessoa e principal, foram atribuídas as variáveis para a inserção dos atributos de uma pessoa tais como o seu nome, uma identificação, região de morada, sexo e idade.

A seguir foi definido o construtor de classe com dados definidos exteriormente na classe Main do programa. E por fim suas propriedades para a manipulação dos atributos pessoa utilizando as funções GET e SET.

A classe Utente, foi definida como a classe “filho” de Pessoa, pois essa herda os atributos de uma pessoa e adiciona o tipo de infeção e o id à pessoa, deste modo não é necessária a criação de novos atributos e propriedades, visto os mesmo já foram definidos na classe pai.

O construtor da classe Utentes herda os atributos da classe pessoa e insere o tipo de infeção e seu id. E esses dados adicionais ~são também tratados com suas respetivas propriedades na presente classe.

Os métodos da versão final do programa baseiam-se inserção, validação e inativação de dados em lista genérica, e tais como as suas pesquisas são semelhantes à de um array, visto que funcionam por pesquisa a partir do índice de posição.

Foram implementados também métodos de menu de escolha, para o utilizador ter uma melhor visualização do que o programa pode oferecer em termos de funcionalidade.

Por fim nesta parte final foi implementado um método de criação e gravação da estrutura de dados em um ficheiro binário, de modo a proteger os dados da melhor forma. E também um método de leitura que faz a verificação da existência do ficheiro no início do programa de modo a carregar os dados já existentes quando o programa inicia, e grava as alterações no mesmo quando o programa encerra.

# Conclusão

## Lições aprendidas

Em conclusão da primeira parte do trabalho proposto foi adquiro os conhecimentos acerca da distribuição de dados e métodos por classes, e do tipo de classes no que diz respeito a heranças, sendo assim os pilares para a programação orientada a objetos. Foi também reforçado a prática de manipulação de arrays em C#, da implementação de propriedades de classe e como instanciar no Main as mesmas.

Em conclusão da finalização do projeto, foi adquirido uma nova forma de pensar em termos de organização de código em linguagem orientada a objetos.

Foi possível também fazer o uso de collections para tratamento de dados, e nos métodos correspondentes ao tipo de estrutura escolhida. E por fim foi apreendido técnicas de gravação e leitura em ficheiros binários.

Por fim concluo que o presente projeto contribuiu para o aprendizado das bases de uma boa programação em linguagem C# (orientada a objetos), nas diversas ferramentas de facilidade que a linguagem oferece e na visão de como deve ser distribuído os módulos do programa por camadas, facilitando a leitura e processamento do código.

Bibliografia

Toda a bibliografia utilizada para a resolução do projeto corresponde ao material didático oferecido pelo professor da disciplina de Linguagens de Programação II Luís Ferreira, tais como power points explicativos e códigos implementados em aula.