INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

RELATÓRIO DE TRABALHO PRÁTICO

Gestão de Doenças e Infeções

JOÃO CARVALHO 18831

VLADYSLAV SAFRONOV 18824

Trabalho realizado sob a orientação de:

Luís Ferreira

Linguagens de Programação II

Licenciatura em Engenharia de Sistemas Informáticos

Barcelos, 26 de Abril de 2020

Índice

1. Introdução

1.1 Contextualização

O Presente relatório surgiu pela necessidade descrever o conjunto com os das unidades curriculares de Linguagens de Programação II.

O tema é uma necessidade corrente, principalmente na área hospitalar, onde todos os profissionais de saúde têm problemas a encontrar e controlar casos positivos de doenças e pandemias, como é o nosso tema corrente do nosso planeta.

Todas as pessoas, agora mais que nunca, vêm necessidade de controlar os casos de doenças, mortos e infetados no país. Por isso, esta ferramenta tenta colmatar e ajudar a fazer algo para controlar os casos e o tipo de doenças encontradas.

1.2 Motivação e Objetivos

O nosso projeto foi desenvolvido com o intuito de contabilizar e manusear os casos infetados em todo o nosso país. Pensamos que podia ser uma boa ideia a implementar, visto a nossa situação atual, e conseguir perceber a nossa importância no meio social em que nos encontramos.

Conseguimos perceber que, com a nossa ajuda, podemos automatizar sistemas de deteção de casos, e fazer com que, por exemplo, médicos e enfermeiros, a todas as horas, consigam saber quem são os casos positivos no nosso país e a sua ficha médica.

2. Descrição do problema

O nosso projeto baseia-se em automatizar a forma como controlamos o número de infetados com uma determinada doença no nosso país. Para isso, pensamos que, para as nossas necessidades mais fundamentais para a execução deste plano, precisávamos das seguintes ferramentas:

- Uma biblioteca de “Objetos de Negócio” – A nossa library “ObjetoNegocio” irá guardar as classes ”Infeção”, “Doentes” e “Pessoa”. Guardamos numa classe “Pessoa” as informações referentes a um ser. No nosso sistema, achamos indispensáveis os seguintes atributos para caracterizar uma pessoa:

Tabela 1 - Classe Pessoa

Nome	Descrição	Tipo
Idade	Idade de uma pessoa	int
Nome	Nome de uma pessoa	string
Região	Região (Norte/Centro/Sul)	Regiao (enum)
ID	ID único de uma pessoa	int
Sexo	Sexo(Masculino/Feminino/Outro)	Sexo (enum)
Data de Nascimento	Data de Nascimento da pessoa	DateTime
Profissão	Profissão de uma pessoa	Profissao (enum)

Depois de guardarmos as informações refentes a uma pessoa, o nosso próximo objetivo foi criar uma classe que guardasse as informações sobre um doente. Esta classe tem uma herança com a classe Pessoa, já que um doente recebe todas as informações sobre um tipo e um nome;

Uma infeção, no nosso problema, foi descrita como tendo um tipo e um nome. O tipo descreve que doença é (vírus, bactéria, tumor, por exemplo) e o nome guarda o nome da doença (covid-19 e salmonela são nomes de exemplos de um tipo de vírus e uma bactéria, repetivamente;

- Uma biblioteca de “Dados” – esta library irá guardar uma única classe, de momento, chamada “Hospital”. Contabilizamos os casos inseridos num Hospital que,

futuramente, pode funcionar para registar todos os casos de infeções num país. Guardamos também um conjunto de utentes, neste caso, uma lista de doentes.

- Uma biblioteca “Regras de Negócio” – a biblioteca “RegrasNegocio” irá guardar apenas uma classe “Regra” que apresenta métodos como adicionar uma infeção,

3. Implementação do problema

Na primeira fase do nosso trabalho, criamos as nossas infeções. Se temos de gerir as pessoas infetadas, conseguimos também gerir qual é a infeção que têm. É uma classe simples, pelo que podemos ainda acrescentar campos nos atributos, para conseguir ter informação ainda mais detalhada sobre uma infeção quando a usamos.

Depois de ter a descrição de uma infeção, criamos um doente. Este doente pode ser criado por defeito com o nosso construtor, em que é atribuída uma infeção e é “ativado” o nosso booleano que controla a atividade ou inatividade da doença no paciente. O outro construtor recebe mais informações que o construtor por defeito. Ela recebe todos os dados necessários sobre uma pessoa e doente, guardando a informação dos atributos deste nos respetivos campos, e passando os restantes dados para a classe pai através do “base”. Atribui isso ao seu respetivo método, localizado na classe pessoa.

A nossa classe pessoa, como já foi referida na descrição do problema, guarda todos os nossos atributos refentes a uma pessoa. Pensamos na profissão, na região e no sexo como enumerações. Esta classe é muito importante na perspetiva de herança. É uma classe fundamental na hierarquia dos dados.

A nossa forma de gerir onde os doentes estão e de os adicionar num hospital, é trota tratada na classe Hospital. Esta classe tem nos seus atributos o máximo de pessoas que um hospital alberga, conta o número de hospital e regista um array de doentes. Nesta classe implementamos também as seguintes funcionalidades:

- Inserimos Doente – Inserimos um doente neste array contido num hospital. Um hospital é uma infraestrutura que recebe indivíduos, pessoas;

- Regista Infeção – Esta função insere uma infeção num determinado doente. Tem alguns métodos necessários para gerir doentes, mas não pensamos em usar nos nossos doentes por serem métodos que podemos reutilizar mais tarde. Alguns dos métodos são:
    - Verificar nome do doente – Com esta função, retornamos true caso exista um determinado nome no conjunto de doentes, false caso contrário;
    - Get Doente – Dado um determinado ID, esta função retorna o doente com aquele ID e null caso não exista;
    - Conta Casos – Esta função conta o número de casos ativos de infeção;
    - Consultar Região – Conta os casos de cada região e imprime-os;
    - Consultar Idade – Conta o número de dados com determinada idade;
    - Consultar Profissão – Conta o número de profissões e imprime-as no ecrã, e o número de cada pessoa em cada profissão;
    - Desativar Infetado – Desativa um infetado quando curado, desativando-o apensar através do seu booleano. Passamos o ID para identificar e desativar, permanecendo, na mesma, no array;
    - Show Ficha – Procedimento que exibe a ficha do determinado utente, através do seu ID;
    - To String – Exibe a ficha de todos os utentes contidos num determinado hospital.
- Uma biblioteca que trata dos dados principais – a biblioteca “Interact” contém a nossa classe “Controller”, que é a classe auxiliar ao main. Apenas temos uma referência para o nosso método “ComeçaProcesso”, que começa o nosso programa.

4. Análise

Para mostrar como funciona o nosso programa, iremos mostrar o output do terminal depois de inserir duas pessoas com duas doenças diferentes.

```
Doente           : joao
Idade            : 20
ID               : 1
Região           : NORTE
Nasceu em        : 22/05/2020 00:00:00
Sexo             : Masculino
Estado Civil     : FUNCIONARIO
Infetado com     : Virus concretamente Covid-19
Ainda infetado   : True

Doente           : Zuca
Idade            : 24
ID               : 2
Região           : CENTRO
Nasceu em        : 22/05/2020 00:00:00
Sexo             : OUTRO
Estado Civil     : DESEMPREGADO
Infetado com     : Bacteria concretamente Cancro
Ainda infetado   : True

Casos Infetados: 2
```

Figura 1 - Ficha do utente através do ID

Fazendo uma espécie de debug ao código, podemos explicar os passos necessários à criação de doenças e pessoas.

Primeiramente, inserimos uma doença. Esta nossa primeira abordagem, obriga a criação de uma doença quando registamos um novo caso. Esta abordagem foi mal pensada e será alterada para a terceira entrega do trabalho, pois pensamos tarde que as infeções podem ser uma lista com já infeções predefinidas.

Depois de ter uma infeção, associamos a uma pessoa, que recebe uma idade, um nome, uma região, sexo, data de nascimento e o tipo de infeção. É atribuído um bool que verifica se ainda está infetado e um ID para ser facilmente encontrado. Depois de ter as pessoas criadas, conseguimos “tratá-las” e alterar o estado do nosso bool, registando a pessoa como já não estando infetada.

5. Conclusão

Com a realização do trabalho, considero que aprendemos mais no que toca ao domínio da linguagem C#. É uma linguagem com imensas aplicações no mundo real, pelo que é bom conseguir melhorar o domínio que temos sobre estas linguagens, aumentando a nossa capacidade e domínio nesta área tão competitiva.

Tendo como missão a resolução de um problema que se foi tornando cada vez mais complicado, principalmente nesta segunda fase, penso que todos os objetivos propostos foram alcançados. O programa responde aos problemas e todos os objetivos iniciais até à data foram cumpridos. Com a nossa próxima entrega, pretendemos melhorar as nossas abordagens. Conseguimos melhorar na clareza do nosso código, na implementações de biblioteca, que tornou tudo também muito mais claro e limpo.

Através de um algoritmo e de um código simples e de fácil compreensão, conseguimos facilmente criar uma pessoa, criar um doente, criar um hospital e conectar todos estes conceitos a uma rede dentro de um hospital que controla todos estes, e todas as suas ações dentro do nosso contexto.

Como em todos os projetos, houve erros e falhas aquando da realização do mesmo. Vimos dificuldades na abordagem de uma pessoa, sendo esta a classe pai do nosso doente. Estes doentes vão ser geridos na nossa classe hospital, e denotamos uma melhoria de algoritmo neste paradigma, comparativamente ao nosso anterior paradigma, o C. Achamos de mais fácil implementação e perceção, pois conseguimos ter todos os constituintes de um projeto separados, bem identificados e todos conectados

Em suma, foi um bom exercício de aprendizagem que melhorou o nosso domínio sobre uma linguagem de programação que tem muitas utilizações no cotidiano. Achamos dificuldades na implementação, mas todas elas resolvidas, maioritariamente através de conversas entre os dois elementos do grupo e consulta aos exercícios feitos em aula.