



INSTITUTO POLITÉCNICO  
DO CÁVADO E DO AVE  
ESCOLA SUPERIOR DE TECNOLOGIA

**RELATÓRIO DE TRABALHO PRÁTICO**

# **Trabalho Prático Fase 3**

---

**DIOGO ROCHA**

**ALUNO Nº 18855**

Trabalho realizado sob a orientação de:  
Luís Ferreira

**Linguagens de Programação II**

**Licenciatura em Engenharia de Sistemas Informáticos -PL**

**Barcelos, Maio de 2020**

## Resumo

Esta fase 2 do trabalho prático, da unidade curricular de **Linguagens de Programação II**, tem como objetivo implementar uma ou várias soluções na linguagem C#.

Nesta segunda fase do trabalho prático, com o tema de “Gestão de Infecionados”, pretende-se corrigir erros apontados pelo professor **Luís Ferreira** e aplicar os últimos capítulos do plano curricular, abordados nas aulas.

Pretendeu-se nesta segunda fase, aplicar novas estruturas de dados, organizar as classes em bibliotecas e guardar os dados em ficheiros.

## Índice

<b>1.CONTEXTUALIZAÇÃO .....</b>	<b>1</b>
1.2. Motivação e objetivos .....	1
<b>2.ANÁLISE AO PROBLEMA .....</b>	<b>2</b>
<b>3.IMPLEMENTAÇÃO .....</b>	<b>3</b>
3.1. Classes utilizadas .....	3
3.2. Exceptions .....	4
3.3. Generics .....	4
3.4. Ficheiros .....	5
<b>4.CONCLUSÃO .....</b>	<b>6</b>

## Lista de Figuras

<b>1.CONTEXTUALIZAÇÃO .....</b>	<b>1</b>
1.2. Motivação e objetivos .....	1
<b>2.ANÁLISE AO PROBLEMA .....</b>	<b>2</b>
<b>3.IMPLEMENTAÇÃO .....</b>	<b>3</b>
3.1. Classes utilizadas .....	3
3.2. Exceptions .....	4
3.3. Generics .....	4
3.4. Ficheiros .....	5
<b>4.CONCLUSÃO .....</b>	<b>6</b>



## **1. Contextualização**

Esta fase 3 do trabalho prático, da unidade curricular de **Linguagens de Programação II**, tem como objetivo implementar uma ou várias soluções na linguagem C#, neste caso um gestor de infeccionados, algo que a DGS (Direção Geral de Saúde) tem feito nestes últimos meses. O programa terá uma abordagem um bocado similar ao que a DGS faz.

### **1.2. Motivação e objetivos**

Esta fase 3, tem como objetivo aplicar os últimos capítulos lecionados durante as aulas como também melhorar onde possa ser melhorado. A motivação é sempre poder fazer bem e melhor sempre que possível. Pesquisar alguns temas de forma autónoma será muito importante para a conclusão desta segunda fase.

## 2. Análise ao problema

Para este trabalho prático havia vários “temas” a escolher. Devido á situação que está a ocorrer globalmente, foi selecionado o tema, que tem como objetivo gerir infectados a nível nacional, numa situação de pandemia.

Na **primeira fase** o objetivo foi de implementar uma estrutura básica com poucos métodos, **orientada a objetos**, de maneira a identificar casos de uma forma geral por região ou outra característica. Foi criado uma classe “Caso”, onde estavam implementos os objetos de casos. Um caso tem definido, um género, idade e região.

Nesta **segunda fase**, a tarefa passou em mudar a estrutura de dados para “**generics**”, organizar o projeto por bibliotecas, guardar os dados em ficheiro binário e adicionar “**Exceptions**”.

Foram criadas as seguintes classes:

- Pessoa (define uma pessoa);
- Caso (define um caso de infeção);
- Recuperados (define uma pessoa recuperada)
- Óbito (define uma pessoa morta)

Cada uma delas com os eu métodos de inserção, consulta e de contabilização.

### 3. Implementação

#### 3.1. Camadas

Para este projeto foi implementada a programação por camadas, ou seja, foram criados 3 projetos, classes, Dgs2 e Regras.

1. Classes -> Objetos de negócios
2. Dgs2 -> Leitura e consulta
3. Regras -> Faz validações

#### 3.2. Classes utilizadas

Foram utilizadas as seguintes classes no digrama a que se segue:

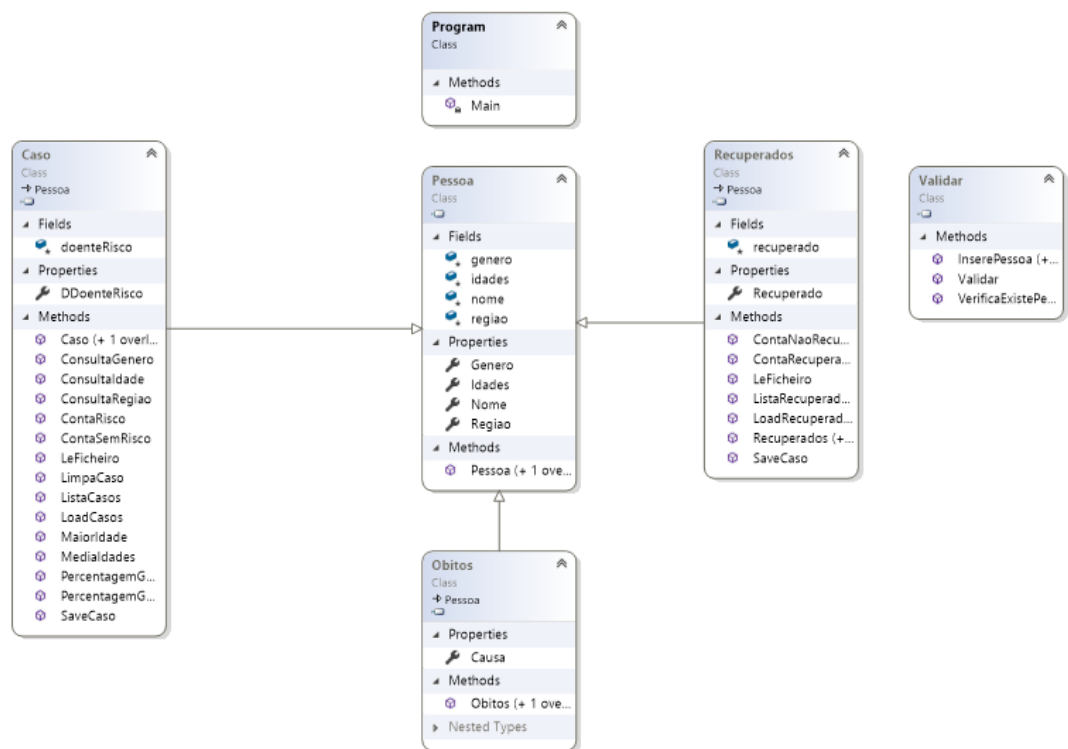


Figure 1-Diagrama de Classes

Figura 1 - Diagrama de classes

No digrama de classes, pode-se observar que a classe “pai” é a classe “**Pessoa**”.

As classes “**Caso**”, “**Óbitos**”, e **Recuperados** herdam de “**Pessoa**”.

“**Caso**”, uma pessoa infetada, com a observação de ser ou não ser doente de risco.

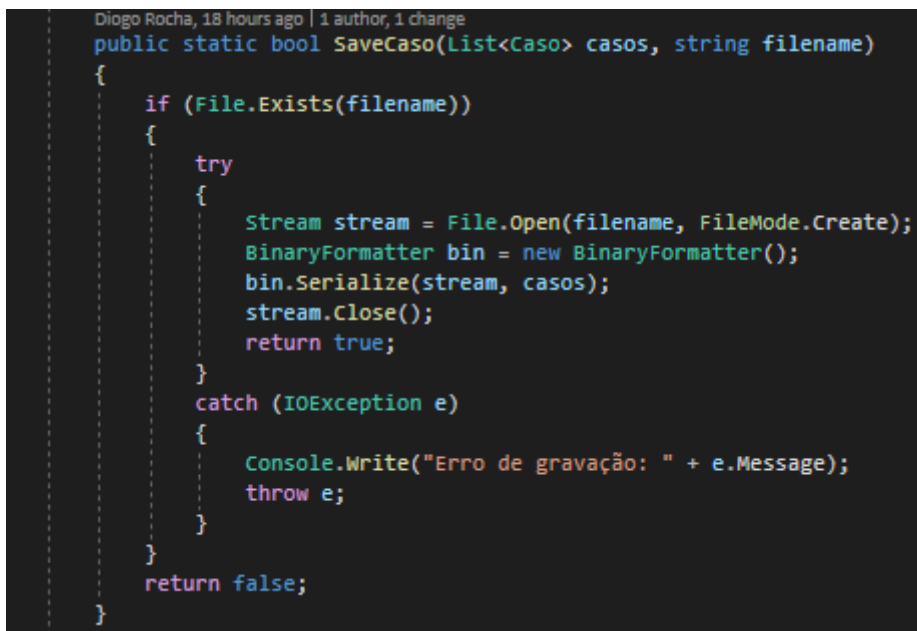
“Recuperados”, representa uma pessoa recuperada ou não.

“Óbitos”, representa uma pessoa morta pelo vírus ou por outra causa.

“Program”, onde são chamados os métodos de todas as classes.

### 3.3.Exceptions

Para identificar erros que podem ocorrer durante a execução do programa, foram implementadas **exceptions** em certas funções. Como por exemplo na **figura 2**, foi implementada numa função de escrita de um ficheiro e em alguns casos foi implementada em certos **inputs** do utilizador. Com esta funcionalidade, foi possível por vezes identificar o erro e solucionar.



```
Diogo Rocha, 18 hours ago | 1 author, 1 change
public static bool SaveCaso(List<Caso> casos, string filename)
{
    if (File.Exists(filename))
    {
        try
        {
            Stream stream = File.Open(filename, FileMode.Create);
            BinaryFormatter bin = new BinaryFormatter();
            bin.Serialize(stream, casos);
            stream.Close();
            return true;
        }
        catch (IOException e)
        {
            Console.WriteLine("Erro de gravação: " + e.Message);
            throw e;
        }
    }
    return false;
}
```

Figura 2-Exemplo de uma **Exception**



### 3.4. Generics

Nesta aplicação foi utilizada uma nova estrutura de dados chamado de **Generics**, esta estrutura de dados permite criar coleções genéricas. Ou seja torna-se mais simples adicionar dados sem que seja necessário percorrer uma lista á procura de uma espaço de memória livre.

```
List<Caso> casos = new List<Caso>();
Caso c = new Caso();

Validar.InserePessoa(casos, new Caso("João", "Norte", 25, "Masculino", "Sim"));
Validar.InserePessoa(casos, new Caso("Ana", "Sul", 25, "Feminino", "Não"));
Validar.InserePessoa(casos, new Caso("Joana", "Sul", 25, "Feminino", "Sim"));
Validar.InserePessoa(casos, new Caso("Mario", "Centro", 32, "Masculino", "Sim"));
Validar.InserePessoa(casos, new Caso("Diogo", "Centro", 8, "Masculino", "Não"));
Validar.InserePessoa(casos, new Caso("Paula", "Sul", 16, "Feminino", "Sim"));
Validar.InserePessoa(casos, new Caso("Cristina", "Sul", 40, "Feminino", "Sim"));
Validar.InserePessoa(casos, new Caso("Isabela", "Norte", 55, "Feminino", "Sim"));
Validar.InserePessoa(casos, new Caso("Sara", "Norte", 65, "Feminino", "Não"));
Validar.InserePessoa(casos, new Caso("Margarida", "Centro", 80, "Feminino", "Não"));
```

Figura 3-Exemplo de uma Lista

### 3.5. Ficheiros

Foi necessário guardar certa informação em ficheiros binários e ler essa mesma informação guardada. Na **figura 4** é possível observar uma função que guarda a informação para um ficheiro.

```
public static bool SaveCaso(List<Caso> casos, string filename)
{
    if (File.Exists(filename))
    {
        try
        {
            Stream stream = File.Open(filename, FileMode.Create);
            BinaryFormatter bin = new BinaryFormatter();
            bin.Serialize(stream, casos);
            stream.Close();
            return true;
        }
        catch (IOException e)
        {
            Console.WriteLine("Erro de gravação: " + e.Message);
            throw e;
        }
    }
    return false;
}
```

Figura 4-Exemplo de uma função que grava

## 4. Conclusão

Esta terceira fase, envolveu consultar o material fornecido pela unidade curricular e alguma pesquisa. Penso ter implementado bem a estrutura de dados “**Generics**”, de acordo de como foi lecionado nas aulas.

Acredito ter seguido todas as normas **CLS**, organizado as minhas classes em bibliotecas e documentado tudo. Foram colocados vários “**Exceptions**”, á medida que iam sendo criados métodos, dos quais em certos casos, ajudou na resolução de problemas. Apesar de estar a pré inserir os dados, foi possível por a gravar para ficheiro binário apesar de não ter sido possível implementar bem o carregar ficheiro.

Apesar do programa ainda estar um pouco incompleto, (não foi possível trabalhar na classe “**Óbitos**” devido ao calendário de testes) penso ter cumprido com o que era pedido no enunciado do trabalho e aplicar a matéria lecionada nesta unidade curricular.