

Relatório de trabalho prático

Gestor de ocorrências de catástrofes naturais

António Oliveira

Aluno nº 18833

Trabalho realizado sob a orientação de:

Luís Ferreira

**Linguagens de Programação II**

**Licenciatura em Engenharia de Sistemas Informáticos**

Barcelos, Maio de 2020

Índice

[1 Introdução 3](#_Toc42289996)

[1.1 Contextualização 3](#_Toc42289997)

[1.2 Resumo 3](#_Toc42289998)

[2 Problema 4](#_Toc42289999)

[3 Desenvolvimento 5](#_Toc42290000)

[3.1 Sistema de controlo de versões 5](#_Toc42290001)

[3.2 Classes 6](#_Toc42290002)

[3.2.1 Estrutura por camadas NTier 6](#_Toc42290003)

[3.2.2 Funcionalidades 7](#_Toc42290004)

[4 Objetivos 8](#_Toc42290005)

[5 Conclusão 9](#_Toc42290006)

Lista de Figuras

Figura 1: Diagrama de classes 6

# Introdução

## Contextualização

Na disciplina de Linguagem de Programação II, o professor Luís Ferreira, desafiou-me a aplicar os conhecimentos adquiridos na sua disciplina, desenvolvendo uma aplicação de tema livre em C#, ou seja, eu poderia escolher o tema, mas também a abordagem do mesmo e o rumo, contando com a sua supervisão e acompanhamento durante o processo.

## Resumo

Este projeto consiste numa aplicação que permite registar e gerir ocorrências de catástrofes naturais, guardando os dados relativos às mesmas e bem como feridos e óbitos.

# Problema

É necessário que o programa permita registar e gerir ocorrências numa classe ocorrências que contem uma estrutura de dados em que guarda os objetos “ocorrência”. Dos vários dados exigidos é necessário existir o distrito, que também necessita da sua própria classe e de uma classe na qual seja guardada os vários distritos, e da catástrofe que irá ter também uma classe para guardar os dados de cada catástrofe e uma que guarde todas as catástrofes.

Para registar os feridos e óbitos de cada ocorrência é necessário a criação de uma classe pai “pessoa” da qual as classes filho “óbito” e “ferido” vão herdar propriedades comuns. Os feridos e óbitos serão guardados em duas estruturas de dados na classe “ocorrência” pois cada objeto ocorrência pode ter feridos e óbitos.

# Desenvolvimento

O desenvolvimento do projeto teve em conta as boas práticas de programação e documentação bem como as normas CLS e os pilares da POO.

Após analisar o problema foi observado que apenas eram necessárias duas heranças, entre a classe Pessoa e Ferido e entre Pessoa e Óbito.

## Sistema de controlo de versões

Durante o desenvolvimento do projeto foi utilizado Git como sistema de controlo de versões e o Git workflow como modelo de trabalho, facilitando desta forma o controlo e gestão de versões. Resumidamente, a branch “master” é a branch com a versão mais atualizada e estável da aplicação, já a branch “develop” é a branch de desenvolvimento, ou seja, recebe através de pull request e consequentemente merge, as novas funcionalidades vindas de uma branch “feature”, sendo então uma branch menos estável e mais propensa a bugs. Sempre que é necessário criar uma nova funcionalidade cria-se uma branch a partir da develop e depois quando terminada faz-se um pull request e merge para a develop. Quando o código estiver estável na develop é lançada uma “release”, ou seja, uma nova versão da aplicação, através de um merge da “develop” para a “master”.

O GitHub foi utilizado como sistema de web hosting para o projeto, visto ser uma plataforma bem consolidada e estável com uma versão gratuita, fornecendo todas as ferramentas necessárias.

O repositório pode ser acedido através deste link <https://github.com/SimaoOliveira99/18833_LP2>.

## Classes

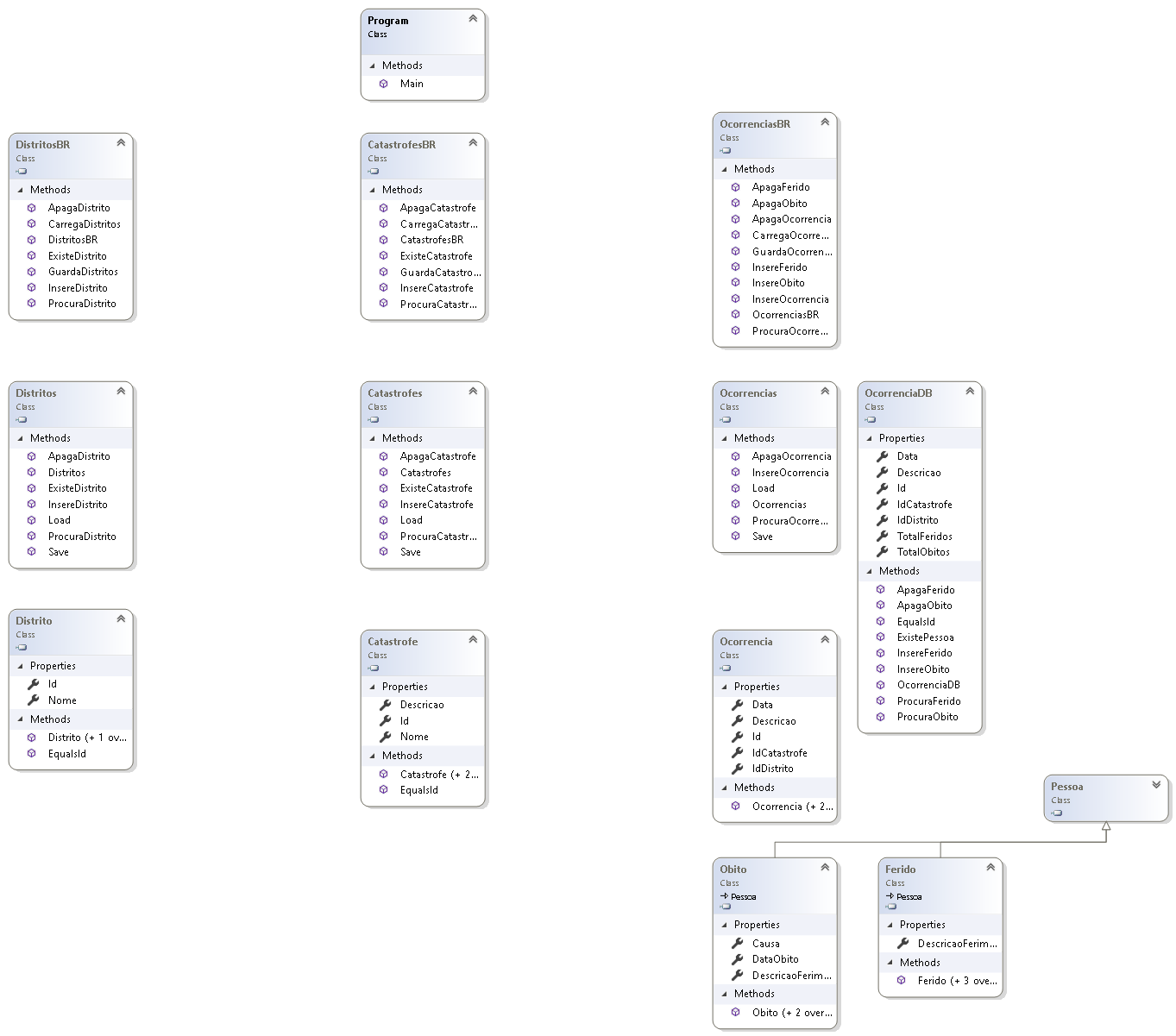


Figura 1: Diagrama de classes

### Estrutura por camadas NTier

Ao desenvolver o projeto foi adotada uma arquitetura de camadas, o NTier, para facilitar e melhor estruturar o projeto. A primeira camada é a camada Front-End, que interage diretamente com o utilizador através de Input/Output (classe “Program”); em contacto direto com esta camada está a camada Business Rules na qual são efetuadas as verificações e validações (classes “DistritosBR”, “CatastrofesBR” e “OcorrenciasBR”), esta camada após as verificações chama as funções e interage com a camada Dados; a camada Dados possui as funções que são capazes de manipular os dados guardados e carregar/guardar dados em ficheiros binários (classes “Distritos”, “Catastrofes”, “Ocorrencias”); por fim, a camada Business Objects, esta camada interage e é utilizada por todas as camadas anteriores e possui os objetos da aplicação (classes “Distrito”, “Catastrofe”, “Ocorrencia”, “Pessoa”, “Obito” e “Ferido”).

### Funcionalidades

O utilizador pode através do programa:

* Distritos
  + Criar
    - Campos: Id e Nome
  + Editar
  + Apagar
* Catástrofes
  + Criar
    - Campos: Id, Nome e Descrição
  + Editar
  + Apagar
* Ocorrências
  + Criar
    - Campos: Id, Data e Descrição
  + Editar
  + Apagar
  + Feridos
    - Inserir feridos na ocorrência
      * Campos: Id, Nome, Idade, Cc, Data de nascimento, Id do distrito e Descrição dos ferimentos
    - Editar feridos de uma ocorrência
    - Apagar feridos da ocorrência
  + Óbitos
    - Inserir óbitos na ocorrência
      * Campos: Id, Nome, Idade, Cc, Data de nascimento, Id do distrito, Data do óbito, Causa da morte e Descrição dos ferimentos
    - Editar óbitos de uma ocorrência
    - Apagar óbitos da ocorrência

# Objetivos

Os objetivos da realização deste projeto era poder aplicar os conceitos aprendidos em aula através do desenvolvimento uma aplicação seguindo as boas praticas CLS e os pilares da POO e estruturada por camadas NTier, que resolvesse o problema pedido, utilizando estruturas de dados eficientes.

# Conclusão

Apesar de ter algumas dificuldades com a estruturação em camadas, com este projeto fui capaz de aplicar heranças entre classes, entender melhor a relação classe e objeto, praticar as boas práticas da POO seguindo as normas CLS e separar as classes em camadas NTier entendendo objetivo por detrás de cada uma.