*Uma imagem com texto, Tipo de letra, Gráficos, logótipo

Descrição gerada automaticamente*



**Programação orientada a objetos**

LICENCIATURA ENGENHARIA INFORMÁTICA MÉDICA

2ºano

2023/2024

Docente: Tiago Castro

Grupo 9

No.22845 – Mateus Costa

No.28027 – Houda El Manssouri

No.28040 – Beatriz Fernandes

No.29860 – Júlio Pereira

Índice

[Repositório GitHub com o trabalho 2](#_Toc187089246)

[https://github.com/pumata22/Trabalho\_POO.git 2](#_Toc187089247)

[**Introdução** 2](#_Toc187089248)

[**Modelo de Negócio** 2](#_Toc187089249)

[**Objetivos** 2](#_Toc187089250)

[**Classes Identificadas** 2](#_Toc187089251)

[**Implementação Essencial das Classes** 3](#_Toc187089252)

[**Desafios Encontrados** 6](#_Toc187089253)

[**Desenvolvimento da Lógica das Classes** 6](#_Toc187089254)

[**Exemplos de Utilização** 6](#_Toc187089255)

[Desafios encontrados 7](#_Toc187089256)

[Conclusão 7](#_Toc187089257)

Repositório GitHub com o trabalho

<https://github.com/pumata22/Trabalho_POO.git>

# **Introdução**

Neste relatório demonstramos o nosso desenvolvimento no trabalho prático da disciplina de Programação Orientada a Objetos. No qual tentamos tentar explicar um pouco algumas partes mais importantes do nosso código.

# **Modelo de Negócio**

* O Modelo de Negócio que nos decidimos adotar é o de uma aplicação de gestão de atividades de socorro.

### **Objetivos**

* O nosso objetivo com este trabalho é conseguir planear, projetar uma aplicação de atividades de socorro que funcione e que possa ser utilizada.

# **Classes Identificadas**

Classes: - Classes: --> Bombeiros

--> Inem

--> ForcasArmadas

--> Ocorrencia

--> Pessoa

--> ProtecaoCivil

--> Veiculos

-->TipoOcorrencia

- Abstratas: --> AbstrataVeiculos

- Interfaces: --> IVeiculos

# **Implementação Essencial das Classes**

* **Bombeiros**

**Atributos**

private string nomQuartel;

private int idQuartel;

private string moradaQuartel;

private int contacto;

**Construtores**

public Bombeiros(){}

public Bombeiros(int idQuartel, string moradaQuartel, int contacto, string nomQuartel){}

* **Inem**

**Atributos**

private int qtdMedicos;

private int qtdEnfermeiros;

private int qtdMeios;

private string posicionamento;

**Construtores**

public Inem() {}

public Inem(int qtdMedicos, int qtdEnfermeiros,int qtdMeios, string posicionamento){}

* **ForçasArmadas**

**Atributos**

private string tipoForca;

private string moradaBase;//base ou esquadra da policia por exemplo

Veiculos veiculo = new Veiculos();

**Construtores**

public ForcasArmadas(){}

public ForcasArmadas(string tipoForca, string moradaBase){}

* **Ocorrência**

**Atributos**

private string morada;

private string veiculos;

private string observacoes;

private int data;

private int idade;

private int nVitimas;

private int nOcorrencia;

private int identificadorTipoEmergencia;

private int nVeiculos;

private int coordenadas1, coordenadas2;

private int nOperacionais;

private double horaAlerta, horaFecho;

private double areaArdida;

Pessoa pessoa = new Pessoa();

**Construtor emergências medicas e agressão**

public Ocorrencia(string morada, string veiculos, string observacoes, int nVitimas, int idade, int nOcorrencia, int identificadorTipoEmergencia, int nOperacionais, Pessoa pessoa){}

**Construtor acidente**

public Ocorrencia (string morada, string veiculos, string observacoes, int nVitimas,int nOcorrencia, int identifacadorTipoEmergencia,int coordenadas1, int coordenadas2, int nOperacionais, Pessoa pessoa) {}

**Construtor incendio urbano e industrial**

public Ocorrencia (string morada, string veiculos,int nOcorrencia, string observacoes, int nVitimas, int identifacadorTipoEmergencia, int coordenadas1, int coordenadas2, int nOperacionais) {}

**Construtor incendio florestal**

public Ocorrencia(string morada, string veiculos, string observacoes,int nOcorrencia,

int identificadorTipoEmergencia, int coordenadas1, int coordenadas2, int nOperacionais, double areaArdida){}

* **Pessoa**

**Atributos**

private int nOperacional;

private string nome;

private int idade;

private int moradaPessoa;

**Construtores**

public Pessoa (int nOperacionalPar, string nomePar, int idadePar, int moradaPessoaPar) {}

* **Veículos**

**Atributos**

private string sigla;

private string categoria;

private int lugares;

**Construtores**

public Veiculos (string siglaPar, string categoriaPar, int lugaresPar) {}

* **Proteção civil**

**Atributos**

private string nome;

private string area;

Veiculos veiculo = new Veiculos();

**Construtores**

public ProtecaoCivil(string nome\_par, string area\_par){ }

# **Desafios Encontrados**

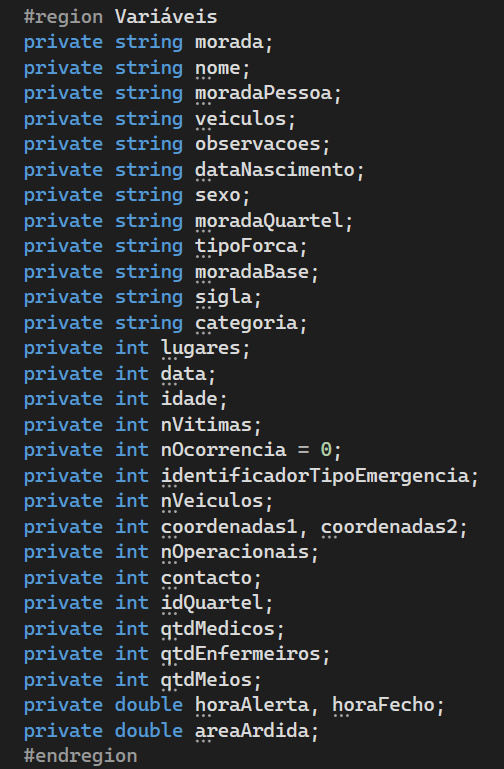
* Como armazenar as ocorrências, para posteriormente mostrar, ainda não entendemos o conceito total, mas achamos que estamos num bom caminho;

# **Desenvolvimento da Lógica das Classes**

* **Como foi mencionado anteriormente, foram criadas 7 classes que são** Bombeiros, Inem, ForcasArmadas, Ocorrência, Pessoa, ProtecaoCivil Veículos e tipo Ocorrência;
* Foi criada também uma classe abstrata chamada Veiculos;
* Por fim foi criada uma interface chamada IVeiculos.

**No Program:**

**Variáveis**

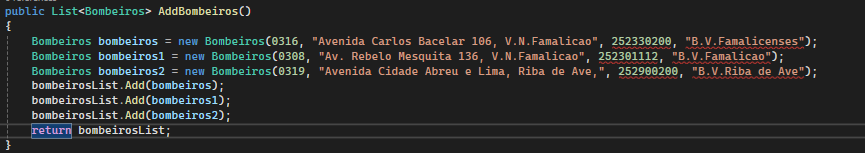
****

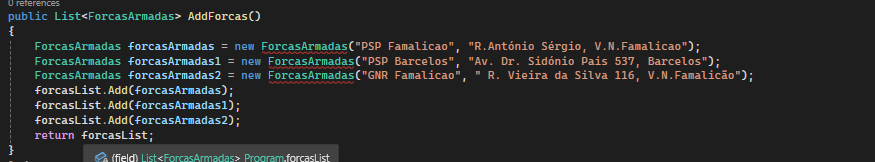
Criamos uma declaração de listas para as seguintes classes:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Adicionamos objetos do tipo “classe” a uma lista e retorna essa lista





Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Depois criamos um método para cada tipo de ocorrência, onde vai ser adicionado a ocorrenciaList o objeto que criamos a partir da classe “Ocorrencia” para o utilizador inserir os dados daquela respetiva urgência. No final o método dirige-nos para o menu caso o utilizador queira efetuar uma nova ocorrência.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, software

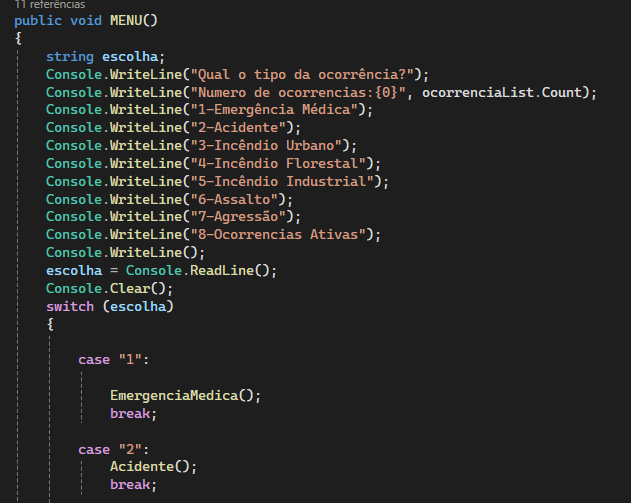
Descrição gerada automaticamente

De seguida apresentamos um método para mostrar as ocorrências ativas, isto através de um foreach que vai mostrar a ocorrência da Lista de ocorrências “ocorrenciaList”, e os respetivos atributos/características da ocorrência.

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

Posteriormente temos o método do Menu, onde é a tela inicial do usuário e que vai escolher qual ocorrência necessita ou outras opções, isso através de um switch case. E consequentemente temos a Main onde recebe o método feito para adicionar ocorrências, onde retorna a ocorrenciaList.



Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, Tipo de letra, captura de ecrã, número

Descrição gerada automaticamente

# Métodos Classe Veículos

# Uma imagem com texto, captura de ecrã, software Descrição gerada automaticamente

**Exemplos de algumas classes:**

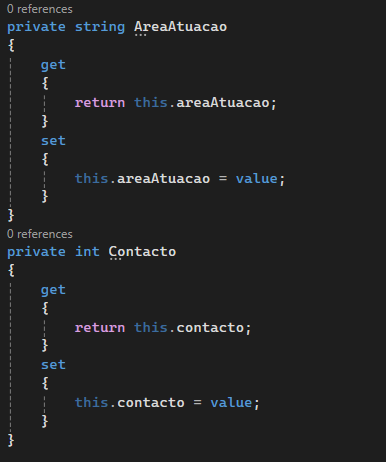
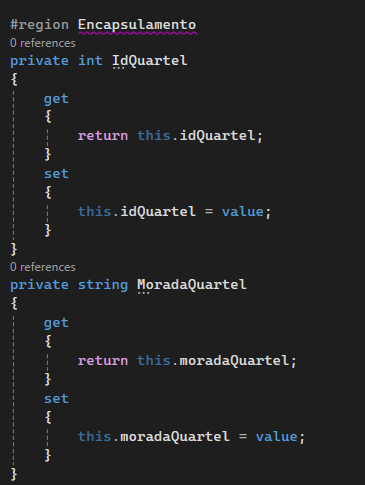
**Proteção civil: exemplo de encapsulamento**

Uma imagem com texto, captura de ecrã, software

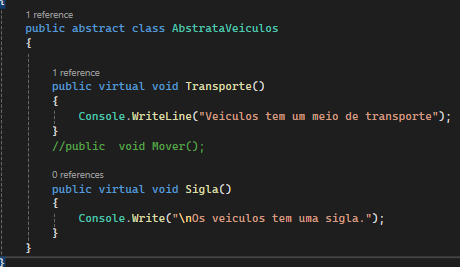
Descrição gerada automaticamente Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

**Bombeiros:Encapsulameto**



**Classe abstrata: Veículos**

****

**Interface: Veículos**

**Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente**

Desafios encontrados

* Implementação da classe abstrata;
* Implementar todas as ideias que tivemos;
* Por vezes tivemos dificuldades na implementação de código;

Conclusão

* Em conclusão, a programação orientada a objetos (POO) é uma abordagem muito útil para desenvolver software. Ao longo deste trabalho, aprendemos sobre os conceitos principais de POO, como encapsulamento, herança e polimorfismo, que nos ajudaram a organizar melhor o nosso código.
* No futuro, à medida que continuamos a aprender e a praticar, é importante explorar novas técnicas e ferramentas que nos ajudem a melhorar ainda mais as nossas habilidades em programação.