**Instituto Politécnico do Cávado e do Ave**

**Escola Superior de Tecnologia**

**Licenciatura**

**em**

**Engenharia Informática Médica**

**Sistema de gestão de urgências de um hospital**

**Docente:** Luís Ferreira

Autor: Ana Margarida Vasco, nº26055

Daniel Macedo, nº26057

Pedro Pinto, nº26049

Novembro e dezembro de 2023

# Resumo

O propósito do nosso projeto prático na disciplina de Programação Orientada a Objetos (POO) consiste no desenvolvimento de um sistema de gestão de urgências hospitalares.

Para a implementação deste sistema, foram utilizadas as seguintes classes: Doente, Médico, Pessoa, SalaaEspera, Local, Consultório e Sistema. Cada uma dessas classes desempenha um papel específico no funcionamento global do projeto, contribuindo para uma gestão eficiente e organizada das situações de urgência.

Ao utilizar estas classes de forma integrada, o sistema visa otimizar a gestão de urgências hospitalares, proporcionando uma abordagem estruturada e eficaz para o tratamento de doentes, de acordo com as suas necessidades e prioridades.

Índice

[Resumo 1](#_Toc154934988)

[Índice de Figuras 3](#_Toc154934989)

[Introdução 5](#_Toc154934990)

[Revisão da Literatura Relacionada com o Trabalho 6](#_Toc154934991)

[Classes Utilizadas no Projeto 7](#_Toc154934992)

[**Hospital** 7](#_Toc154934993)

[**Consultório** 7](#_Toc154934994)

[**SalaaEspera** 11](#_Toc154934995)

[**Sistema** 14](#_Toc154934996)

[**Local** 18](#_Toc154934997)

[**Intervenientes** 19](#_Toc154934998)

[**Doente** 19](#_Toc154934999)

[**Medico** 23](#_Toc154935000)

[**Pessoa** 26](#_Toc154935001)

[Resultados 27](#_Toc154935002)

[Conclusão 28](#_Toc154935003)

# Índice de Figuras

[Figura 1: Atributos de Consultório 7](#_Toc154157587)

[Figura 2: Métodos de Consultório 7](#_Toc154157588)

[Figura 3: Construtor por Omissão 8](#_Toc154157589)

[Figura 4: Construtor Personalizado 8](#_Toc154157590)

[Figura 5: Construtor Estático 8](#_Toc154157591)

[Figura 7: Propriedade do Número 9](#_Toc154157592)

[Figura 6: Propriedade de Especialidade 9](https://d.docs.live.net/d535e3ac31600a3c/Ambiente%20de%20Trabalho/Relatório%20Trabalho%20POO.docx#_Toc154157593)

[Figura 8: Propriedade de Estado 9](#_Toc154157594)

[Figura 9: Propriedade Total de Consultórios 9](#_Toc154157595)

[Figura 10: Função que Insere Medico no Consultório 10](#_Toc154157596)

[Figura 11: Função que Verifica se Existe Medico no Consultório 10](#_Toc154157597)

[Figura 12: Função que Verifica se Existe Doente no Consultório 10](#_Toc154157598)

[Figura 13: Função que Adiciona Medico no Consultório 10](#_Toc154157599)

[Figura 14: Função que Adiciona Doente no Consultório 11](#_Toc154157600)

[Figura 15: Destrutor de classe do Consultório 11](#_Toc154157601)

[Figura 16: Atributos de SalaaEspera 11](#_Toc154157602)

[Figura 17: Métodos de SalaaEspera 12](#_Toc154157603)

[Figura 18: Construtor Instância da SalaaEspera 12](#_Toc154157604)

[Figura 19: Função que Adiciona Doente na Sala de Espera 12](#_Toc154157605)

[Figura 20: Função que Conta os Doentes 12](#_Toc154157606)

[Figura 21: Função que Mostra Lugares Ocupados 13](#_Toc154157607)

[Figura 22: Função que Verifica se Existe Doentes na Sala de Espera 13](#_Toc154157608)

[Figura 23: Função que Conta Quantos Doentes Existem na Sala de Espera 13](#_Toc154157609)

[Figura 24: Função que Retorna uma lista de Doentes ordenados por Prioridade 14](#_Toc154157610)

[Figura 25: Destrutor de Classe da Sala de Espera 14](#_Toc154157611)

[Figura 26: Atributos de Sistema 14](#_Toc154157612)

[Figura 27: Listas de Doentes e Médicos no Sistema 14](#_Toc154157613)

[Figura 28: Função que Regista um Doente 15](#_Toc154157614)

[Figura 29: Função que Regista um Médico 15](#_Toc154157615)

[Figura 30: Função que Encaminha para a Triagem 15](#_Toc154157616)

[Figura 31: Função que Atualiza o Historico 16](#_Toc154157617)

[Figura 32: Função que Mostra Informação de um Doente 16](#_Toc154157618)

[Figura 33: Função que Atruibui um Medico a um Doente 16](#_Toc154157619)

[Figura 34: Função que Realiza uma Consulta 17](#_Toc154157620)

[Figura 35: Função que Faz a Triagem 17](#_Toc154157621)

[Figura 36: Função que Encaminha para a Sala de Espera 18](#_Toc154157622)

[Figura 37: Enum Estado 18](#_Toc154157623)

[Figura 38: Class Local (Herança) 18](#_Toc154157624)

[Figura 39: Atributos de Doente 19](#_Toc154157625)

[Figura 40: Métodos de Doente 19](#_Toc154157626)

[Figura 41: Construtor por omissão do Doente 20](#_Toc154157627)

[Figura 42: Construtor que permite personalizar os atributos do doente 20](#_Toc154157628)

[Figura 43: Construtor estático do atributo Total de Doentes 20](#_Toc154157629)

[Figura 45: Propriedade do id do Doente 21](#_Toc154157630)

[Figura 44: Propriedade do Nome 21](https://d.docs.live.net/d535e3ac31600a3c/Ambiente%20de%20Trabalho/Relatório%20Trabalho%20POO.docx#_Toc154157631)

[Figura 47: Propriedade do NIF 21](#_Toc154157632)

[Figura 46: Propriedade do Número de Utente 21](https://d.docs.live.net/d535e3ac31600a3c/Ambiente%20de%20Trabalho/Relatório%20Trabalho%20POO.docx#_Toc154157633)

[Figura 49: Propriedade de Pulseira 21](#_Toc154157634)

[Figura 48: Propriedade de Prioridade 21](https://d.docs.live.net/d535e3ac31600a3c/Ambiente%20de%20Trabalho/Relatório%20Trabalho%20POO.docx#_Toc154157635)

[Figura 50: Propriedade de Sintomas 22](https://d.docs.live.net/d535e3ac31600a3c/Ambiente%20de%20Trabalho/Relatório%20Trabalho%20POO.docx#_Toc154157636)

[Figura 51: Propriedade de Histórico 22](#_Toc154157637)

[Figura 52: Propriedade estática do atributo Total de Doentes 22](#_Toc154157638)

[Figura 53: Método ToString() da classe Doente 22](#_Toc154157639)

[Figura 54: Função que Atualiza Histórico 22](#_Toc154157640)

[Figura 55: Função que Mostra Informação do Doente 23](#_Toc154157641)

[Figura 56: Destrutor da Class Doente 23](#_Toc154157642)

[Figura 57: Atributos de Médico 23](#_Toc154157643)

[Figura 58: Métodos de Médico 23](#_Toc154157644)

[Figura 59: Construtor por omissão de Médico 24](#_Toc154157645)

[Figura 60: Construtor que permite a personalização dos atributos do Médico 24](#_Toc154157646)

[Figura 61: Construtor estático do atributo Total de Médicos 24](#_Toc154157647)

[Figura 63: Propriedade do Nome 25](#_Toc154157648)

[Figura 62: Propriedade de Especialidade 25](https://d.docs.live.net/d535e3ac31600a3c/Ambiente%20de%20Trabalho/Relatório%20Trabalho%20POO.docx#_Toc154157649)

[Figura 64: Propriedade do Número de Identificação 25](#_Toc154157650)

[Figura 65: Propriedade estática do atributo Total de Médicos 25](#_Toc154157651)

[Figura 66: Método ToString() na classe do Médico 26](#_Toc154157652)

[Figura 67: Destrutor de class de Médico 26](#_Toc154157653)

[Figura 68: Atributo de Pessoa (Herança) 26](#_Toc154157654)

# Introdução

## 

A gestão de um sistema de urgências num hospital tem um papel fundamental na prestação de cuidados de saúde de qualidade. Atualmente, os hospitais enfrentam diversos desafios na organização e priorização de urgências, resultando em tempos de espera prolongados e possível subutilização de recursos médicos e na disponibilidade de recursos eficientes.

Assim, com o aumento constante dos doentes, que procuram as urgências, tende a gerar pressão nos recursos hospitalares o que origina desafios na triagem, no atendimento prioritário e na gestão eficaz das salas de espera.

A eficiência da gestão de urgências é fundamental para salvar vidas e garantir que os doentes recebam tratamento adequado quando necessário. Uma abordagem sistemática pode reduzir os tempos de espera, melhorar a disponibilidade de recursos e aumentar a satisfação do doente.

Este sistema visa preencher falhas na gestão de emergências hospitalares, fornecendo uma solução abrangente e eficiente. De forma a atender às necessidades específicas do ambiente de urgências hospitalares, pretendemos criar uma ferramenta que integre as classes consultório, sala de espera, local, sistema, médico, doente, pessoa. Deste modo, permite uma triagem mais rápida e precisa, disponibilizando doentes com base nas suas condições, prioridades e histórico, criando listas de espera dinâmicas, garantindo que doentes mais urgentes recebam atendimento imediato.

# Revisão da Literatura Relacionada com o Trabalho

A gestão eficaz de um sistema de urgências é fundamental num hospital para melhorar a eficiência e a qualidade do atendimento. Nesta gestão é crucial a integração de classes específicas para consultório, médico, doente e salas de espera e permite simplificar a gestão global do sistema.

Num atendimento competente o maior desafio encontra-se no Sistema, na classificação das prioridades e na gestão da lista de espera. Ao longo do tempo, pesquisas têm demonstrado que uma boa gestão das listas de espera é crucial para facilitar o fluxo de doentes. A disponibilidade dinâmica de recursos pode ser facilitada por sistemas informáticos que melhoram a distribuição da carga de trabalho e reduzem os tempos de espera. Através da linguagem C# consegue-se criar comunicação com o utilizador e conter diferentes classes de um sistema hospitalar.

# Classes Utilizadas no Projeto

Neste projeto utilizamos as seguintes classes: SalaaEspera, Local, Consultório, Sistema na biblioteca Hospital e Doente, Médico, Pessoa na biblioteca Intervenientes. Utilizamos também uma class Program para o main.

## **Hospital**

## **Consultório**

A classe consultório contém diversos atributos:

* Uma constante para defenir o tamanho máximo do consultório;
* Número do consultório, do tipo inteiro;
* Especialidade do médico, do tipo especialidade;
* Uma variável do tipo Medico chamada medicoResponsavel;
* Uma variável do tipo Doente chamada doenteConsultorio;
* O número total de médicos, do tipo inteiro;
* O número total de consultórios, do tipo inteiro.

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

Descrição gerada automaticamente

Figura : Atributos de Consultório

A seguir aos atributos temos os Métodos:

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

Descrição gerada automaticamente

Figura : Métodos de Consultório

No partido CONST temos 3 funções:

* **Construtor por omissão:** Inicializa propriedades padrão, como número do consultório, especialidade (padrão para pediatria), estado (padrão para ocupado) e incrementa uma variável estática totCon.
* **Construtor personalizado:** Permite a personalização de propriedades como número, especialidade, estado, e associação a um médico e doente específicos. Também incrementa totCon.
* **Construtor estático:** Inicializa uma variável estática totCon. Além disso, há comentários que explicam a função de cada construtor. A região #region CONST é usada para agrupar esses construtores visualmente em ambientes de desenvolvimento integrado (IDE).

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

Descrição gerada automaticamente

Figura : Construtor por Omissão

Uma imagem com texto, software, Software de multimédia, captura de ecrã

Descrição gerada automaticamente

Figura : Construtor Personalizado

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

Descrição gerada automaticamente

Figura : Construtor Estático

Seguidamente, as propriedades, que estão divididas em propriedades de instância e propriedades de classe, as primeiras são utilizadas para obter ou definir os diversos atributos do consultório, já as últimas permitem obter o número total de consultórios, e é um valor controlado apenas internamente.

Propriedades Instância:

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

Figura : Propriedade do Número

Figura : Propriedade de Especialidade

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

Descrição gerada automaticamente

Figura : Propriedade de Estado

Propriedades classe:

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

Descrição gerada automaticamente

Figura : Propriedade Total de Consultórios

No parâmetro OUTROS temos algumas funções:

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

Descrição gerada automaticamente

Figura : Função que Insere Medico no Consultório

A função InsereMedicoConsultorio faz o seguinte:

* Adiciona um médico ao consultório, verificando se o médico já existe no consultório.
* Retorna true se a operação for bem-sucedida, false caso contrário.

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

Descrição gerada automaticamente

Figura : Função que Verifica se Existe Medico no Consultório

A função ExisteMedicoConsultorio faz o seguinte:

* Verifica se um médico já está associado ao consultório.
* Retorna true se o médico existir, false caso contrário.

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

Descrição gerada automaticamente

Figura : Função que Verifica se Existe Doente no Consultório

A função ExisteDoenteConsultorio faz o seguinte:

* Verifica se um doente está associado ao consultório.
* Uma imagem com texto, captura de ecrã, Tipo de letra, software

  Descrição gerada automaticamenteRetorna true se houver um doente, false caso contrário.

Figura : Função que Adiciona Medico no Consultório

A função AdicionarMedicoConsultorio faz o seguinte:

* Adiciona um médico ao consultório, exibindo mensagens adequadas.
* Atribui o médico ao consultório se não houver um médico associado.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura : Função que Adiciona Doente no Consultório

A função AdicionarDoenteConsultorio faz o seguinte:

* Adiciona um doente ao consultório, exibindo mensagens adequadas.
* Verifica se o médico associado ao doente é o mesmo responsável pelo consultório.
* Atribui o doente ao consultório se as condições forem atendidas.

Por último, o destrutor da classe, utilizado para eliminar a classe construída, para limpar a memória.

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

Descrição gerada automaticamente

Figura : Destrutor de classe do Consultório

## **SalaaEspera**

A classe SalaaEspera contém diversos atributos:

* Uma lista de doentes;
* Capacidade máxima da sala de espera, do tipo inteiro;
* Uma lista de recursos disponíveis.

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

Descrição gerada automaticamente

Figura : Atributos de SalaaEspera

A seguir aos atributos temos os Métodos:

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

Descrição gerada automaticamente

Figura : Métodos de SalaaEspera

Nos construtores temos este construtor instância:

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

Descrição gerada automaticamente

Figura : Construtor Instância da SalaaEspera

Este construtor é utilizado para criar uma instância da classe SalaEspera com capacidade máxima definida (capacidadeMaxima), estado inicial vazio (Estado.VAZIO), e listas vazias de doentes e recursos disponíveis.

No parâmetro outros temos várias funções:

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

Descrição gerada automaticamente

Figura : Função que Adiciona Doente na Sala de Espera

A função AdicionarDoenteSalaEspera faz o seguinte:

* Adiciona um objeto Doente à lista de doentes na sala de espera.

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

Descrição gerada automaticamente

Figura : Função que Conta os Doentes

A função ContarDoentes faz o seguinte:

* Retorna o número de doentes presentes na sala de espera.

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

Descrição gerada automaticamente

Figura : Função que Mostra Lugares Ocupados

A função MostrarLugaresOcupados faz o seguinte:

* Exibe na console os nomes dos doentes presentes na sala de espera, ou informa que não há doentes.

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

Descrição gerada automaticamente

Figura : Função que Verifica se Existe Doentes na Sala de Espera

A função ExisteDoenteSalaEspera faz o seguinte:

* Verifica se há pelo menos um doente na sala de espera.
* Retorna true se há doentes, false se não há.

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

Descrição gerada automaticamente

Figura : Função que Conta Quantos Doentes Existem na Sala de Espera

A função QuantosDoentesExistemSalaEspera faz o seguinte:

* Conta quantos doentes na sala de espera possuem uma pulseira específica.
* Retorna o número de doentes com a pulseira especificada.

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

Descrição gerada automaticamente

Figura : Função que Retorna uma lista de Doentes ordenados por Prioridade

A função ObterDoentesPorPrioridade faz o seguinte:

* Retorna uma lista de doentes na sala de espera ordenados por prioridade, do maior para o menor.

Por último, o destrutor da classe, utilizado para eliminar a classe construída, para limpar a memória.



Figura : Destrutor de Classe da Sala de Espera

## **Sistema**

A classe Sistema contém diversos atributos:

* Uma lista de lista de Espera;
* Uma lista de lista de médicos;
* Uma lista de consultórios.

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

Descrição gerada automaticamente

Figura : Atributos de Sistema

A seguir temos várias funções:

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

Descrição gerada automaticamente

Figura : Listas de Doentes e Médicos no Sistema

public Sistema faz o seguinte:

* Inicializa duas listas, uma para doentes (listaDeEspera) e outra para médicos (listaDeMedicos), quando um novo sistema é criado.

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

Descrição gerada automaticamente

Figura : Função que Regista um Doente

A função RegistarDoente faz o seguinte:

* Regista um novo doente na lista de espera, criando uma instância da classe Doente com os parâmetros fornecidos.
* Adiciona o doente à lista de espera.
* Exibe uma mensagem informando o registo e encaminha o doente para triagem.

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

Descrição gerada automaticamente

Figura : Função que Regista um Médico

A função RegistarMedico faz o seguinte:

* Regista um novo médico na lista de médicos, criando uma instância da classe Medico com os parâmetros fornecidos.
* Adiciona o médico à lista de médicos.
* Exibe uma mensagem informando o registo.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura : Função que Encaminha para a Triagem

A função EncaminharParaTriagem faz o seguinte:

* Função privada que encaminha um doente para triagem com base na sua pulseira.
* Exibe mensagens específicas dependendo da pulseira do doente.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura : Função que Atualiza o Historico

A função AtualizarHistorico faz o seguinte:

* Atualiza o histórico médico de um doente na lista de espera com base no ID fornecido.
* Exibe mensagens informativas.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura : Função que Mostra Informação de um Doente

A função ObterInformação faz o seguinte:

* Retorna informações de um doente na lista de espera com base no ID fornecido.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura : Função que Atruibui um Medico a um Doente

A função AtribuirMedico faz o seguinte:

* Atribui um médico a um doente com base nos IDs fornecidos.
* Exibe mensagens informativas.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura : Função que Realiza uma Consulta

A função RealizarConsulta faz o seguinte:

* Realiza uma consulta para um doente com base no ID fornecido e em um médico.
* Verifica consultórios disponíveis e simula a realização da consulta.
* Exibe mensagens informativas.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura : Função que Faz a Triagem

A função RealizarTriagem faz o seguinte:

* Realiza a triagem de um doente, atribuindo prioridade com base na pulseira.
* Encaminha o doente para a sala de espera após a triagem.
* Exibe mensagens simuladas de triagem.

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

Descrição gerada automaticamente

Figura : Função que Encaminha para a Sala de Espera

A função EncaminharParaSalaEspera faz o seguinte:

* Encaminha um doente para a sala de espera após realizar a triagem.
* Adiciona o doente à lista de espera e à sala de espera.
* Exibe mensagens informativas.

## **Local**

Esta classe é usada como herança das classes acima referidas (Consultório, SalaaEspera e Sistema). Nela inserimos algumas linhas de código que são as seguintes:

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

Descrição gerada automaticamente

Figura : Enum Estado

Este excerto de código significa que:

* Define um conjunto de valores possíveis para o estado, como VAZIO e OCUPADO.

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

Descrição gerada automaticamente

Figura : Class Local (Herança)

Esta classe mostra que:

* Possui uma propriedade chamada estado que pode assumir os valores definidos no enum Estado.
* A relação é de composição, indicando que a classe Local tem um membro do tipo Estado.

## **Intervenientes**

## **Doente**

A classe Doente contém diversos atributos:

* Tem o id do doente, do tipo inteiro;
* NIF do doente, do tipo inteiro;
* Número do utente, do tipo inteiro;
* Pulseira do doente, do tipo pulseira;
* Prioridade, do tipo prioridade;
* Historico medico do doente, do tipo string;
* Sintomas do doente, do tipo string;
* O número total de doentes, do tipo inteiro.

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

Descrição gerada automaticamente

Figura : Atributos de Doente

A seguir dos atributos temos os METODOS:

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

Descrição gerada automaticamente

Figura : Métodos de Doente

No parâmetro CONST temos:

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

Descrição gerada automaticamente

Figura : Construtor por omissão do Doente

Este excerto de código significa que:

* Cria um objeto Doente com valores padrão.
* Inicializa os atributos com valores específicos, como idDoente, nome, nif, etc.
* Incrementa a variável estática totDoe, que pode estar a contar o número total de instâncias da classe.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura : Construtor que permite personalizar os atributos do doente

Este excerto de código significa que:

* Cria um objeto Doente com valores personalizados, permitindo a inicialização específica dos atributos.
* Aceita parâmetros para Id, nome, nif, nutente, pul (pulseira), pri (prioridade), hist (histórico) e sint (sintomas).
* Incrementa a variável estática totDoe.

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

Descrição gerada automaticamente

Figura : Construtor estático do atributo Total de Doentes

Seguidamente, as propriedades, que estão divididas em propriedades de instância e propriedades de classe, as primeiras são utilizadas para obter ou definir os diversos atributos do consultório, já as últimas permitem obter o número total de consultórios, e é um valor controlado apenas internamente.

Propriedades Instância:

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

Figura : Propriedade do id do Doente

Figura : Propriedade do Nome

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

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

Descrição gerada automaticamente

Figura : Propriedade do NIF

Figura : Propriedade do Número de Utente

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

Figura : Propriedade de Pulseira

Figura : Propriedade de Prioridade

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

Figura : Propriedade de Sintomas

Figura : Propriedade de Histórico

Propriedades classe:

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

Descrição gerada automaticamente

Figura : Propriedade estática do atributo Total de Doentes

Nos overrides temos uma função tostring:

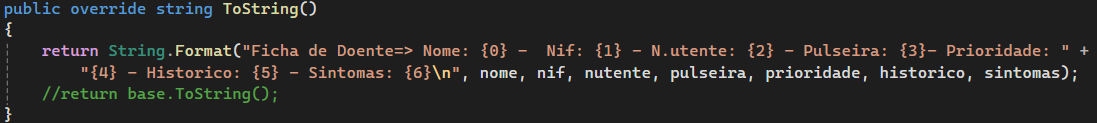


Figura : Método ToString() da classe Doente

Este excerto de código significa que:

* Sobrescreve o método ToString() da classe base Object.
* Retorna uma string formatada contendo informações relevantes sobre o objeto Doente, como nome, NIF, número de utente, pulseira, prioridade, histórico e sintomas.
* A formatação é realizada usando a função String.Format().

No parâmetro Outros, temos varias funções:

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

Descrição gerada automaticamente

Figura : Função que Atualiza Histórico

A função AtualizarHistorico faz o seguinte:

* Atualiza o histórico médico do doente, acrescentando informações ao histórico existente.

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

Descrição gerada automaticamente

Figura : Função que Mostra Informação do Doente

A função ObterInformacao faz o seguinte:

* Retorna uma string formatada contendo informações essenciais sobre o doente, incluindo ID, nome, pulseira, prioridade e histórico médico.

Por último, o destrutor da classe, utilizado para eliminar a classe construída, para limpar a memória.

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

Descrição gerada automaticamente

Figura : Destrutor da Class Doente

## **Medico**

A classe Medico contém diversos atributos:

* Especialidade do medico, do tipo especialidade;
* Tem o número de identificação do medico, do tipo inteiro;
* Total de médicos, do tipo inteiro.

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

Descrição gerada automaticamente

Figura : Atributos de Médico

A seguir dos atributos temos os METODOS:

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

Descrição gerada automaticamente

Figura : Métodos de Médico

No parâmetro CONT temos:

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

Descrição gerada automaticamente

Figura : Construtor por omissão de Médico

Este excerto de linhas de código significa que:

* Inicializa os atributos de um médico com valores específicos, como nome, especialidade e número de identificação.
* Incrementa a variável estática totMed, que pode estar a contar o número total de instâncias da classe.

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

Descrição gerada automaticamente

Figura : Construtor que permite a personalização dos atributos do Médico

Este excerto de linhas de código significa que:

* Aceita parâmetros para nome, esp (especialidade) e nidentificacao (número de identificação).
* Incrementa a variável estática totMed.

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

Descrição gerada automaticamente

Figura : Construtor estático do atributo Total de Médicos

Este excerto de linhas de código significa que:

* Um construtor estático é chamado automaticamente antes de qualquer instância da classe ser criada ou qualquer membro estático ser acessado.

Seguidamente, as propriedades, que estão divididas em propriedades de instância e propriedades de classe, as primeiras são utilizadas para obter ou definir os diversos atributos do consultório, já as últimas permitem obter o número total de consultórios, e é um valor controlado apenas internamente.

Propriedades instância:

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

Figura : Propriedade do Nome

Figura : Propriedade de Especialidade

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

Descrição gerada automaticamente

Figura : Propriedade do Número de Identificação

Propriedades classe:

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

Descrição gerada automaticamente

Figura : Propriedade estática do atributo Total de Médicos

Nos overrides temos:

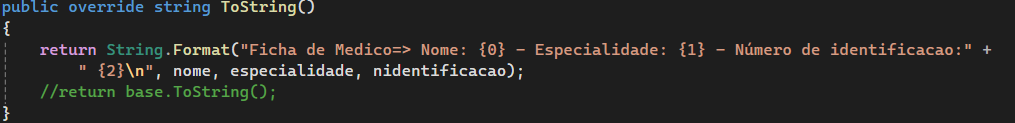


Figura : Método ToString() na classe do Médico

Este excerto de código ToString significa que:

* Sobrescreve o método ToString() da classe base Object.
* Retorna uma string formatada contendo informações relevantes sobre o objeto Medico, como nome, especialidade e número de identificação.
* A formatação é realizada usando a função String.Format().

Por último, o destrutor da classe, utilizado para eliminar a classe construída, para limpar a memória.

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

Descrição gerada automaticamente

Figura : Destrutor de class de Médico

## **Pessoa**

Esta classe é usada como herança das classes acima referidas (Doente e Medico). Nela inserimos algumas linhas de código que são as seguintes:



Figura : Atributo de Pessoa (Herança)

Atributo de Doente e Medico.

# Resultados

O desenvolvimento do sistema de gestão de urgências hospitalares baseado em Programação Orientada a Objetos (POO) resultou na criação de classes como Doente, Médico, Pessoa, Local, Sala de Espera, Consultório e Sistema. Utilizando as propriedades e métodos destas classes, foi possível modelar eficazmente a interação entre doentes, médicos, salas de espera e outros elementos fundamentais para a gestão de urgências hospitalares. A implementação dos enums Pulseira, Prioridade, Estado e Especialidade ofereceu uma abordagem flexível para classificar e priorizar casos médicos, contribuindo para a eficiência do sistema.

O sistema permitiu a criação, registo e atribuição de médicos e doentes, simulação de consultas, triagem e encaminhamento para salas de espera, demonstrando uma capacidade abrangente de gerir eventos urgentes num ambiente hospitalar. A integração das classes, construtores e métodos refletiu uma estrutura coesa e modular, facilitando a manutenção e expansão do sistema no futuro.

# Conclusão

Através da implementação do sistema de gestão de urgências hospitalares em C# utilizando POO, conseguimos desenvolver uma solução robusta e flexível para lidar com emergências médicas. A modelagem orientada a objetos proporcionou uma representação clara e estruturada dos elementos do sistema, criando assim uma base sólida para operações hospitalares em tempo real.

Os construtores e métodos personalizados nas classes Doente e Médico possibilitaram a criação de instâncias com valores padrão ou personalizados, proporcionando adaptabilidade ao contexto específico do ambiente hospitalar. A utilização de enums como Pulseira e Prioridade trouxe uma semântica clara e consistente na categorização e priorização de casos médicos.

Os resultados evidenciam uma implementação bem-sucedida do sistema, destacando a eficiência na gestão de doentes, a atribuição de médicos e a realização de consultas. A estrutura modular do código promove a extensibilidade e manutenção, assegurando a capacidade de adaptação às necessidades futuras do hospital. Este projeto representa não apenas uma solução funcional, mas também um exemplo de boas práticas de programação orientada a objetos na construção de sistemas complexos.