Instituto Politécnico do Cávado e do Ave

Escola Superior de Tecnologia

Curso Licenciatura em

Engenharia Informática Médica

Relatório de Trabalho Pratico

Ana Margarida Maia Pinto nº23458

Letícia Carvalho Azevedo nº 23554

Diogo Mário Sá Fernandes nº 24017

Dezembro de 2024

Esta página foi deixada em branco propositadamente.

Instituto Politécnico do Cávado e do Ave

Escola Superior de Tecnologia

Curso Licenciatura

em

Engenharia Informática Médica

Relatório de Trabalho Pratico

Unidade Curricular

Programação Orientada a Objetos

Nome dos Alunos

Ana Margarida Maia Pinto

Letícia Carvalho Azevedo

Diogo Mário Sá Fernandes

Nome do Docente:

Tiago Daniel Torres Castro

Novembro de 2024

Esta página foi deixada em branco propositadamente.

Resumo

O presente relatório pretende dar a conhecer, o trabalho prático realizado na unidade curricular de Programação Orientada a Objetos, com o objetivo de projetar uma aplicação em C# para a gestão de um Centro de Saúde. Este trabalho foi dividido em duas partes para uma mais fácil implementação.

Neste documento apresentamos uma explicação sobre como foi estruturado o programa.

Palavras-Chaves: Programação Orientada a Objetos, Saúde, C#, Algoritmos, Classes, Objetos, Atributos, Métodos, Herança.

Abstract

This report aims to present the practical work carried out on the Object Oriented Programming course, with the objective of designing an application in C# for the management of a Health Center. This work was divided into two parts for easier implementation.

In this document we present an explanation of how the program was structured.

Keywords: Object-Oriented Programming, Health, C#, Algorithms, Classes, Objects, Attributes, Methods, Inheritance.

Índice

Lista de Acrónimos e Notação

UML Unified Modeling Language

1. Introdução

Este projeto enquadra-se na Unidade Curricular de Programação Orientada a Objetos, lecionada pelo docente Tiago Castro, do curso de Engenharia Informática Médica do Instituto Politécnico do Cávado e do Ave. O objetivo deste será que sejam desenvolvidas soluções em C#.

1.1 Objetivos

Neste tópico, apresentamos os objetivos do trabalho proposto. Entre as metas estão a análise de problemas reais, o aprimoramento da experiência e das habilidades no desenvolvimento de software, e o fortalecimento da proficiência em C#. Também procuramos estruturar uma solução utilizando uma biblioteca de classes, garantir a produção de código de alta qualidade, bem documentado e realizar testes adequados no código desenvolvido.

1.2 Estrutura documento

Quanto à organização do documento, este será dividido em dois tópicos principais: Implementação Teórica e Implementação Prática. No tópico da implementação teórica iremos detalhar o planeamento realizado para abordar a resolução do projeto. Já no tópico da implementação prática será apresentada a execução prática do problema, mostrando como as soluções foram implementadas.

2. Enunciado

Neste trabalho é-nos proposto nove temas para implementar na linguagem C#, sendo estes os seguintes:

- Gestão de Atividades de Socorro;
- Gestão de Obra de Construção Civil;
- Gestão de Condomínios;
- Gestão de rendas/imóveis;
- Gestão de alojamentos turísticos;
- Helpdesk;

- Comércio eletrónico;
- Gestão de jardim zoológico;
- Gestão de Centro de Saúde;

Para este projeto, pretendemos realizar um sistema que permita gerir um Centro de Saúde.

3. Implementação Teórica

Neste capítulo estará descrito todo o planeamento feito para a resolução deste projeto. Este está dividido em vários tópicos que depois de definidos foram implementados na prática que iremos ver no capítulo a seguir.

3.1. Estruturação do Projeto em Camadas

A solução para este trabalho será dividida em vários projetos para uma maior organização do código.

- **CentrodeSaudeProject.App** – onde o método main está contido, sendo este o projeto de inicialização;
- **CentrodeSaudeProject.Class** – onde estão as classes do projeto, com os respetivos atributos e métodos essenciais para cada classe;
- **CentrodeSaudeProject.Enums** – onde se encontram os enumerados;

Esta estruturação em camadas ajuda na manutenção, legibilidade e escalabilidade do código, permitindo a modificação de uma camada sem afetar as outras. Por fim, isto facilita a compreensão do sistema como um todo.

3.2. Definição das Estruturas de Dados a utilizar

Neste projeto, foi essencial definir uma estrutura de dados adequada para atender às necessidades do trabalho prático. Isso envolveu a criação de classes, listas e outros tipos de dados personalizados.

3.3. Classes

Para a realização deste projeto, foi necessário definir algumas classes que serviriam como modelo para a criação dos objetos posteriormente. Essas classes possuem propriedades e métodos que, em alguns casos, podem ser herdados de outras classes.

Neste subcapítulo, apresentaremos todas as classes necessárias. Após análise e discussão, identificamos como essenciais a criação de seis classes, que serão detalhadas a seguir:

- **Paciente**: Representa um paciente do centro de saúde.
- **Exame**: Representa um exame médico solicitado para um paciente
- **Cama**: Representa uma cama disponível ou ocupada no centro de saúde
- **Consulta**: Representa uma consulta médica agendada para um paciente
- **CentroSaude**: Representa o centro de saúde em si e centraliza os dados e operações do sistema de gestão
- **Quarto**: Representa os quartos do centro de saúde, onde os pacientes podem ser internados
- **Médico**: Representam os médicos que realizam consultas e exames no centro de saúde
- **Enfermeiro**: Representam os enfermeiros responsáveis pelo cuidado básico dos pacientes
- **Pessoa**: Representa uma abstração básica para qualquer individuo no sistema

3.4. Desenvolvimento da lógica de Classes

Os ficheiros fornecem a base de um sistema de gestão para um centro de saúde.

Classe Pessoa:

É uma classe abstrata que encapsula atributos básicos de uma pessoa, como nome, idade, número do cartão de Cidadão, NIF e sexo. Tem um método ToString () para apresentar uma pessoa como string. Serve como base para outras classes mais específicas, isto é, a classe Paciente, Médico e Enfermeiro vão herdar a classe Pessoa.

Classe Paciente:

É uma classe que representa os pacientes no sistema, conseguimos aceder às consultas associadas, cama ocupada e estado do paciente.

Tem 4 métodos essenciais:

- **AtribuirCama(Cama)**: liga o paciente a uma cama e atualiza o estado para internado;
- **LiberarCama():** Remove a ligação entre paciente e cama e atualiza o estado para alta;
- **AdicionarConsulta(Consulta) e RemoverConsulta(Consulta):** gerência as consultas do paciente;

Esta classe está diretamente relacionada com camas, consultas e quartos.

Classe Médico:

É uma classe que representa os médicos e herda da classe Pessoa. Tem um método simples ToString () para exibir o ID e os dados do médico.

Classe Enfermeiro:

É uma classe que representa os enfermeiros e herda da classe Pessoa. Tem um método semelhante ao médico, apenas inclui um método para exibir os dados do enfermeiro.

Classe Consulta:

É uma classe que gere as consultas realizadas pelos pacientes. Tem como atributos a data, custo, diagnóstico, exames associados e o médico responsável. Tem como métodos principais **AdicionarExame (Exame) e RemoverExame (Exame)** e gerem os exames ligados à consulta.

Os construtores da classe inicializam o médico e data como informações obrigatórias.

Classe Exame:

É uma classe que define os exames realizados no centro. Tem como atributos a data, resultado, tipo e médico responsável. Apenas tem dois métodos, o **AssociarMedico (Medico) e o ToString ()**, que nomeadamente atribui o exame a um médico e exibe os detalhes do exame.

Classe Cama:

É uma classe que representa as camas nos quartos predefinidos. Tem como atributos o número, disponibilidade e enfermeiro responsável.

Tem três métodos principais:

- **Ocupar ():** Marca a cama como ocupada;
- **Libertar ():** Marca a cama como disponível;
- **ToString ():** Exibe informações da cama;

Classe Quarto:

É uma classe que contém camas e gere a lotação. Tem como atributos o número do quarto, a lotação máxima e a respetiva lista de camas.

Tem três métodos principais:

- **AdicionarCama (Cama):** Adiciona uma cama ao quarto;
- **AtribuirCama ():** Retorna a primeira cama disponível e marca como ocupada;
- **AtribuirEnfermeiro (Enfermeiro):** Liga um enfermeiro responsável ao quarto;

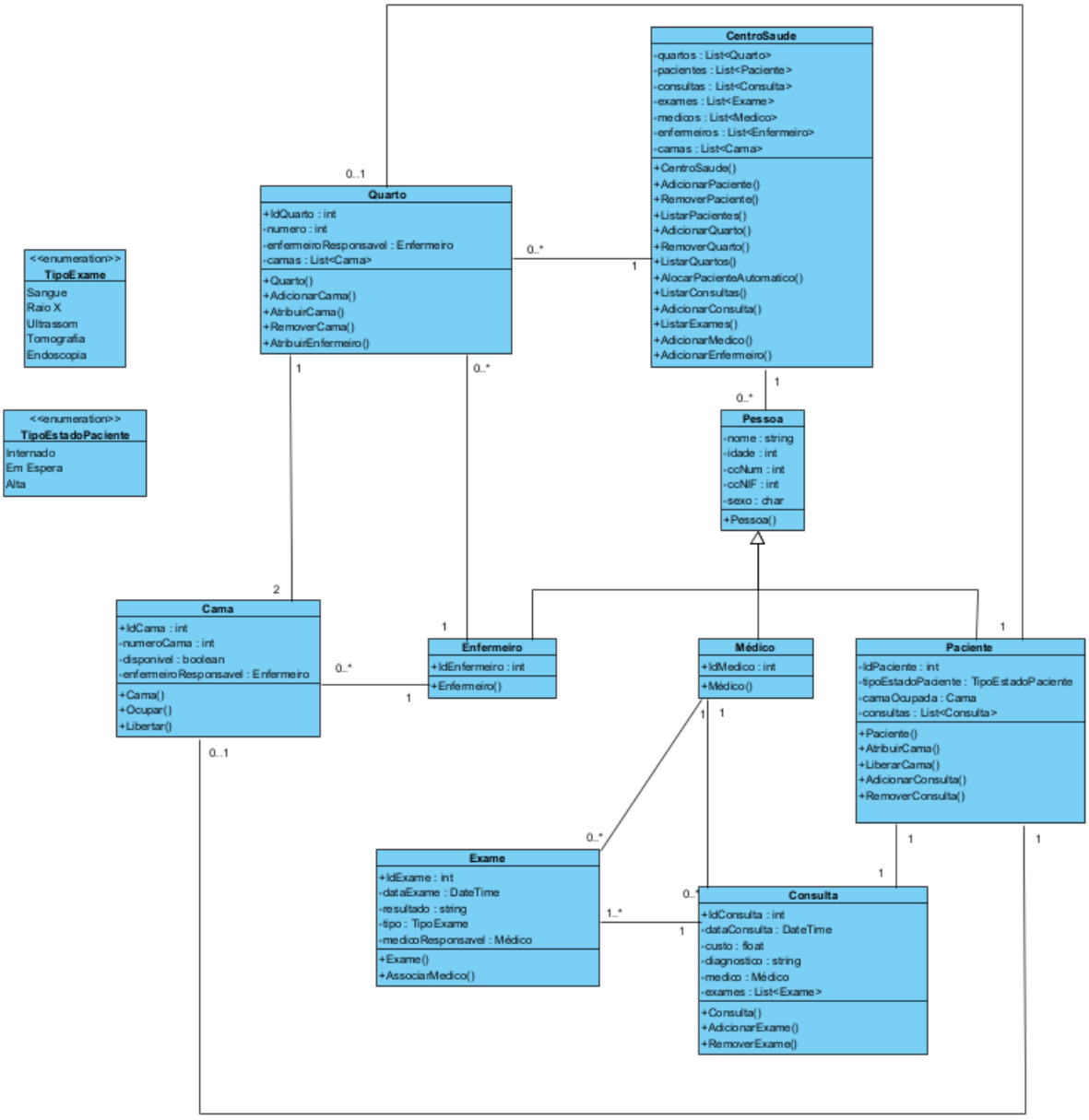
Classe CentroSaúde:

É uma classe que centraliza todas as operações, tais como a gestão de pacientes, médicos, enfermeiros, consultas, exames, quartos e camas. Implementa lógica de alto nível, tais como a alocação automática de camas para pacientes e lista de informações.

3.5. Diagrama de Classes

Um diagrama de classes é uma ferramenta de modelação utilizada para representar a estrutura de um sistema orientado a objetos. Este descreve as classes, anteriormente referidas no tópico anterior, juntamente com os seus atributos, métodos e as relações entre elas. Desta forma, o diagrama proporciona uma visão clara da organização do software e das interações entre os seus componentes.

Na imagem seguinte, é apresentado o diagrama de classes:



4. Implementação Prática

Neste capítulo, será apresentada a implementação prática de todo o projeto. Os subcapítulos abordarão cada projeto individualmente, com exceção do projeto. .App, que foi utilizado exclusivamente para testar os métodos implementados e, portanto, não contém conteúdo relevante para documentação detalhada.

4.1. CentrodeSaudeProject.Enums

Conforme mencionado anteriormente, este projeto teve como objetivo criar um ficheiro .cs que reúne todos os enumerados necessários para o desenvolvimento do trabalho. Os enumeradores definidos são os seguintes:

- **TipoEstadodoPaciente**: Define os possíveis estados em que um paciente pode estar no fluxo do centro de saúde.
- **TipoExame**: Define os tipos de exames disponíveis no centro de saúde, como Exame de Sangue, Raios-X, etc.

4.2. Exemplos de Utilização (4 Pilares Fundamentais)

Herança: A herança permite que uma classe herde atributos e métodos de outra.

A classe Pessoa é a classe base que fornece atributos comuns como Nome, Idade, CC e NIF.

As classes Paciente, Medico e Enfermeiro herdam esses atributos e comportamentos básicos:

- Paciente adiciona o estado clínico (TipoEstadoPaciente) e a cama ocupada.
- Medico tem um ID exclusivo para identificação.
- Enfermeiro também possui um identificador próprio.

Estas classes herdam os atributos da Classe Pessoa

Abstração: A abstração oculta os detalhes da implementação, apresenta apenas funcionalidades relevantes.

A classe Pessoa serve apenas como base para as especializações.

A classe Pessoa é declarada como abstrata. Isso significa que ela não pode ser instanciada diretamente:

- É um conceito genérico que define o que é comum para todas as pessoas.
- Apenas classes derivadas (Paciente, Medico, Enfermeiro) podem ser instanciadas.

Polimorfismo: O polimorfismo permite que os métodos em diferentes classes derivadas tenham comportamentos distintos.

O método ToString está sobrescrito em várias classes derivadas de Pessoa Classe Paciente, Classe Médico, Classe Enfermeiro:

- Na classe Paciente, ele exibe informações como o estado do paciente e a cama ocupada.
- Na classe. Médico, exibe o ID e o nome do médico.
- Na classe Enfermeiro, exibe o ID e o nome do enfermeiro.

Encapsulamento: O encapsulamento protege os dados, permitindo que sejam acedidos e modificados apenas por métodos.

- A classe Cama controla o atributo _disponivel usando a propriedade Disponivel Classe Cama.
- Métodos como Ocupar e Libertar garantem que o estado da cama seja alterado de maneira segura.

4.3. CentrodeSaudeProject.Class

Conforme abordado na teoria, nesta secção serão apresentadas as classes propriamente ditas, contendo apenas os seus atributos, métodos e propriedades, organizados de forma clara e objetiva.

Classes	Atributos	ToString ()
Paciente	idPaciente consultas camaOcupada tipoEstadoPaciente Construtor	Sim
Médico	IdMedico Construtor	Sim
Exame	IdExame dataExame resultado Tipo medicoResponsavel Construtor	Sim
Cama	IdCama Número Disponível Construtor	Sim
Consulta	IdConsulta Medico Data Custo Diagnostico Exames Construtor	Sim
CentroSaude	Pacientes Quartos Medicos Enfermeiros Consultas	Não

	Exames Camas	
Quarto	IdQuarto Número Camas EnfermeiroResponsavel	Sim
Pessoa	Nome Idade CCNum NIF Sexo	Sim

4.4. Menus

Na arquitetura do sistema, a classe CentroDeSaúde serve como base para o projeto. Essa classe é representada no ficheiro Program.cs, que atua como o núcleo onde toda a lógica inicial é controlada.

Os exames e as consultas são elementos diretamente relacionados a outras classes, o que significa que a funcionalidade das mesmas está encapsulada nas classes médico e paciente. Assim, a eficiência em criar menus independentes para estas funcionalidades no contexto principal é muito baixa.

Os menus, como o do paciente, focam ações essenciais e específicas, como visualizar as informações de um paciente e adicionar novos pacientes ao sistema, entre outras. Manter essas funcionalidades evita a sobrecarga do utilizador e aperfeiçoa a interface do sistema.

A criação de menus melhora a estrutura do sistema e facilita a utilização do mesmo.

5. Desafios Encontrados

Durante o desenvolvimento do projeto, tivemos várias dificuldades, a primeira foi realizar a lógica do programa em si, de como iriamos criar um programa para gerir uma Unidade de Saúde. Outra grande dificuldade foi mesmo o uso da matéria lecionada na UC, mas agora podemos dizer que esta já está mais consolidada.

Outro desafio que tivemos foi tentar fazer algo grande, acabando por nos perder dentro do código em si, tendo até que refazer algumas partes, para simplificar e tornar mais legível e simples.

O maior desafio foi a implementação dos métodos e da lógica dos mesmos, pois a teoria de classes e a estrutura com menus foi algo que ficou consolidado a partir da Unidade Curricular, o problema foi mesmo a linguagem que é bastante confusa em comparação a outras linguagens.

6. Futuras Implementações

Após a realização do trabalho ficaram por fazer algumas coisas importantíssimas, que serão as implementações futuras, começando com o uso de interfaces, agora que o trabalho foi finalizado, e vendo a extensão e alguma confusão no código, percebemos que o uso de interfaces é essencial, a principal razão pela qual não foram usadas neste programa, foi que apenas tivemos essa perceção mesmo no fim da realização do trabalho, e como o trabalho já estava estruturado e o tempo também não permitia a mudança então fica para o futuro.

Outra implementação imprescindível é a salvaguarda da informação, ou seja, algo que guarde a informação das listas do programa, ainda fizemos a pesquisa de como o iriamos fazer e chegamos a duas hipóteses, ou o uso de ficheiros .txt ou então do uso de JSON, começamos a implementar a opção dos ficheiros .txt, mas não conseguimos realizá-la e, como o prazo estava próximo, acabamos por não o realizar voltando à versão anterior.

7. Conclusão

A realização deste trabalho possibilitou o aprofundamento dos conhecimentos adquiridos, ao longo do semestre, desta presente cadeira.

O projeto mostrou-se desafiante, mas com o devido planeamento e execução, os obstáculos foram superados e os objetivos estabelecidos foram alcançados. A comunicação entre os membros da equipa foi fundamental para a sua finalidade.

Desta forma, o trabalho não só correspondeu à exigência da unidade curricular, como também se revelou uma oportunidade enriquecedora para aplicar os conhecimentos teóricos na prática, contribuindo para o crescimento profissional.

Bibliografia

[1] Moodle IPCA

https://elearning.ipca.pt/2425/my/

[2] Visual Paradigm

[3] Github

https://github.com/Pura51/Gestao-de-Centro-de-Saude