Instituto Politécnico do Cávado e do Ave

Licenciatura em Engenharia de Sistemas Informáticos

Programação Orientada a Objetos

1º Trabalho Prático

Students :
Francisco Arantes - 23504

Teacher :
Luís G. Ferreira

15 de novembro de 2023

Conteúdo

Lista de Siglas

UC Unidade Curricular

POO Programação Orientada a Objetos

NIF Número de Identificação Fiscal

1 Introdução

Este trabalho da *(UC)* de *(POO)* foca a análise de problemas reais simples e a aplicação do Paradigma Orientado a Objetos na implementação de possíveis soluções.

O tema escolhido parte da criação de uma loja *online* com classes que representam produtos, clientes e pedidos. Posteriormente serão implementados métodos especiais como por exemplo: adicionar certo produto a um carrinho, calcular o preço total e processar pedidos de forma a gerir a loja *online*.

2 Objetivos do Projeto

Esta cadeira tem como objetivo consolidar os conceitos fundamentais do Paradigma Orientado a Objetos. A abordagem prática é enfatizada através da análise de problemas reais.

É dada uma ênfase particular ao desenvolvimento de habilidades de programação em *C#*, uma linguagem de programação muito utilizada no contexto do Paradigma Orientado a Objetos.

Este projeto tem como objetivo final fornecer as competências necessárias para aplicar os princípios do Paradigma Orientado a Objetos de maneira eficaz na resolução de problemas práticos e no desenvolvimento de *software* de qualidade.

3 Primeira Fase

Durante a Fase 1 do projeto, concentramo-nos na estrutura inicial, identificação e implementação essencial das classes para o sistema em desenvolvimento. Foram identificadas as classes que desempenharão funções fundamentais dentro do paradigma orientado a objetos. Foi utilizado o *Visual Paradigm* para alguns rascunhos de como seria o potencial diagrama de classes do sistema. 3, posteriormente foi usado o *Visual Studio* para a construção do diagrama de classes. 5

A implementação inicial destas classes foi realizada com sucesso, escolheu-se a estrutura de dados *array*, alinhando-se com as exigências específicas do projeto.

O relatório atual documenta o progresso até à data. Destacam-se as classes, detalhes sobre a implementação inicial e as estrutura de dado adotada. Adicionalmente, são apresentados desafios enfrentados durante esta fase, juntamente com as soluções propostas para superá-las.

Um dos principais focos desta fase foi o cumprimento dos prazos estabelecidos. A gestão eficiente do tempo foi aplicada de forma a garantir que as atividades planeadas fossem concluídas dentro do prazo definido.

4 Arquitetura do Projeto

4.1 Biblioteca de classes *User*

A figura 1 mostra parte do diagrama de classes do sistema de gestão da loja *online*, esta parte representa as classes presentes na biblioteca de classes dos utilizadores.

A *class User* realçada pelo retângulo vermelho representa um utilizador do sistema, as classe *Admin* e *Costumer* destacadas por o retângulo a verde, herdam as propriedades *email* e *password* da classe *User*.

Para além das classes destacadas foram criadas outras 3 classes, *Users, Admins* e *Costumers*, responsáveis por gerir os respetivos utilizadores com o uso da estrutura de dados escolhida (*array*).

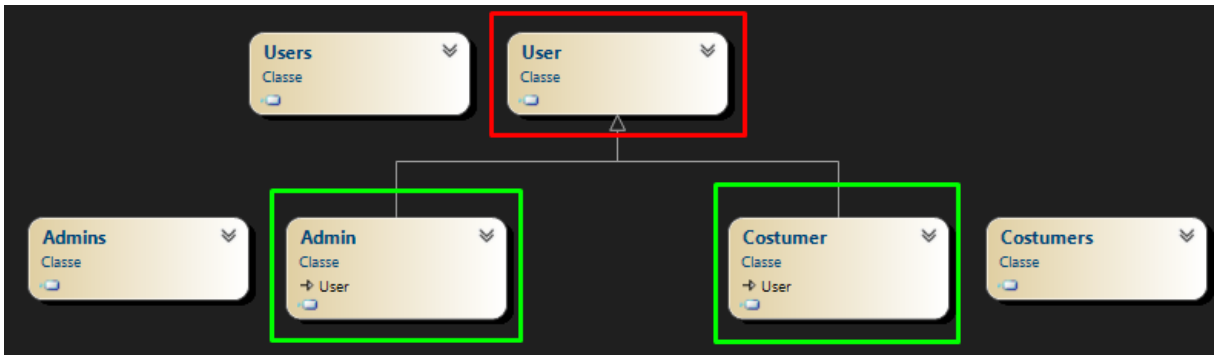

Figura 1: Relação entre *User* e *Costumer/Admin*

4.1.1 Classe *User* e *Users*

A figura 3, mostra as propriedades e métodos implementados para cada classe do tipo *User*.

Preferencialmente a começar pela classe *Users* realçada a vermelho, foi implementada a propriedade *AllUsers* que consiste em um *array* com todos os *users*, o limite de *users* é definido no inicio da class. De seguida, foram implementados métodos para adicionar e remover *users*. Foi também criado um construtor para inicializar uma nova instância da classe*Users* do tipo *array*.

Para a classe *User* realçada a verde, foi definido o *email* e a *password* e 2 construtores, que aceitam nenhum e 2 argumentos(*email* e *password*) respetivamente.

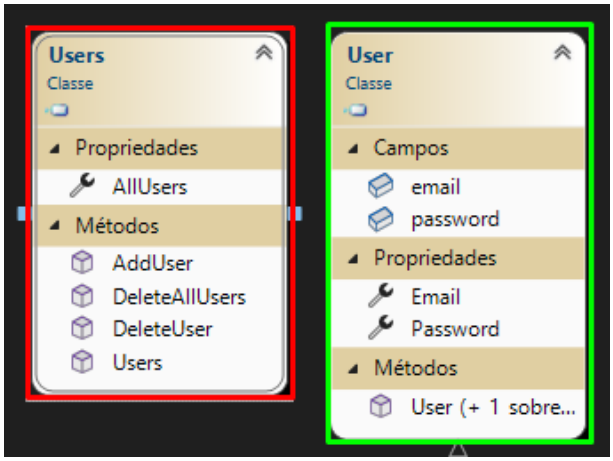

Figura 2: Propriedades e Métodos da classe *User* e *Users*

4.1.2 Classe *Costumer* e *Costumers*

A figura 2, mostra as propriedades e métodos implementados para cada classe do tipo *Costumer*.

Preferencialmente a começar pela classe *Costumer* realçada a verde, foram implementadas as propriedades, *Address*, NIF,*PhoneNumber*, *userName* e *zipCode* baseadas no registo do site da PcDiga 4.

Posteriormente foram implementados métodos para a classe *Costumer*, 2 construtores para inicializar membros estáticos e uma nova instância da class *Costumer* com valores padrão e mais outros 2 construtores que recebem parâmetros como, nome, email, password mais direcionado para um registo simples, e morada, código-zip, contacto telefónico e NIF mais direcionado para um registo ou personalização mais detalhada do perfil do cliente.

Para a classe *Costumers* realçada a vermelho, foi implementado novamente métodos para adicionar e remover clientes de uma estrutura de dados.

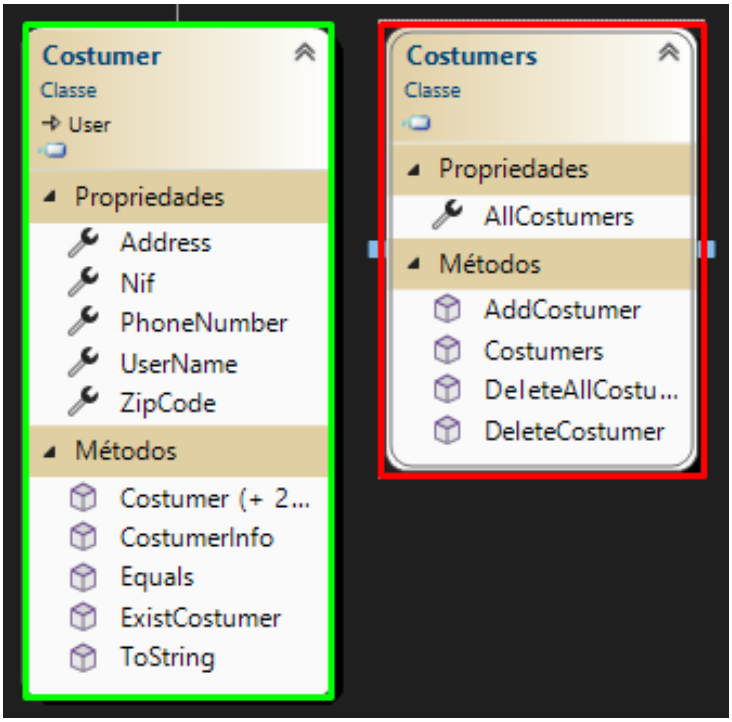

Figura 3: Propriedades e Métodos da classe *Costumer* e *Costumers*

4.1.3 Classe *Admin* e *Admins*

A figura 4, mostra as propriedades e métodos implementados para cada classe do tipo *Costumer*.

Preferencialmente a começar pela classe *Admin* realçada a verde, foram implementadas as propriedades, *AdminKey* responsável por atribuir um fator extra de segurança/privacidade ao *Admin* e *AdminUsername* que será o nome com que o administrador se irá destacar.

Posteriormente foram implementados métodos, 2 construtores para inicializar membros estáticos e uma nova instância da class *Admin* com valores padrão e mais outros 2 construtores que recebem parâmetros como, email, password e chave mais direcionado para um *login in*, e nome, email, password e chave mais direcionado para um registo do administrador.

Para a classe *Admins* realçada a vermelho, foi implementado novamente métodos para adicionar e remover administradores de uma estrutura de dados.

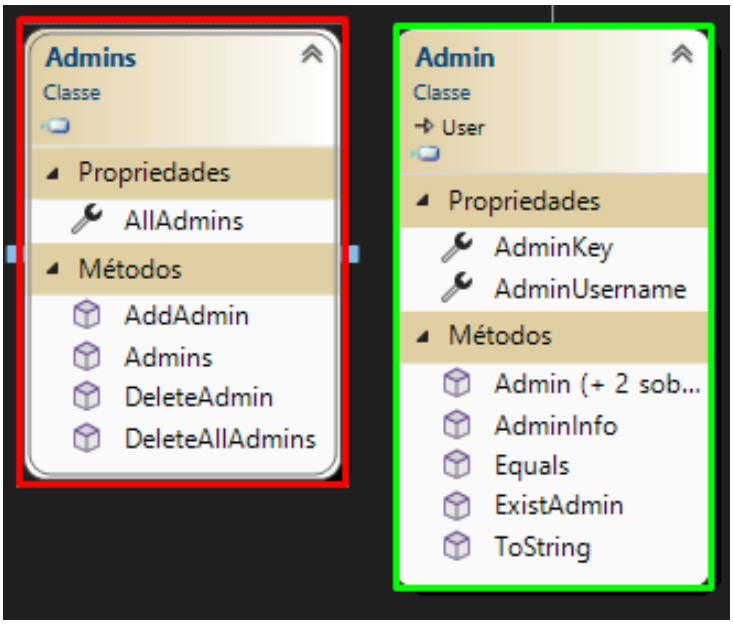

Figura 4: Propriedades e Métodos da classe *Admin* e *Admins*

4.2 Biblioteca de classes *Order*

A figura 5 mostra a segunda parte do diagrama de classes, mais especificamente as classes presentes na biblioteca de classes *Order*.

A *class Order* realçada pelo retângulo vermelho representa um pedido efetuado por um utilizador do sistema de uma certa quantidade de produtos, as classe *Payment* e *ShoppingCart* destacadas por o retângulo a verde, herdam as propriedades *orderId* e *totalAmmount* da classe *Order*.

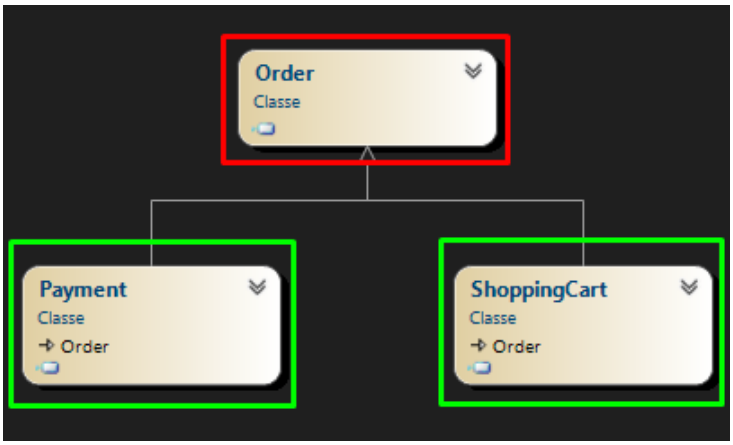

Figura 5: Relação entre *Order* e *Payment/ShoppingCart*

4.3 Biblioteca de classes *Product*

A *class Product* mostrada na figura 6, representa um produto selecionado por um utilizador/criado por um administrador do sistema.

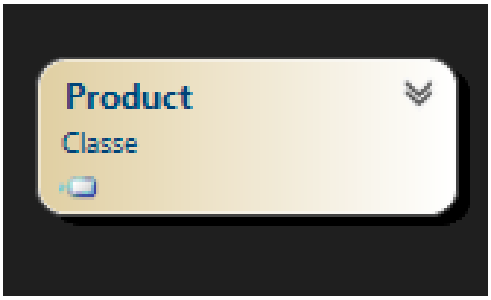

Figura 6: Classe implementada na biblioteca *Product*

5 Problemas encontrados

Para esta primeira fase, a maior dificuldade encontrada foi a estruturação e compreensão do negócio, ou seja, a capacidade de transpor as lojas *online* que estão bastante presentes no nosso dia a dia, em um projeto orientado a objetos. Pensar nas relações entre as classes que iria precisar de implementar, juntamente com as propriedades e métodos associados.

6 Conclusão

Em suma, nesta primeira fase explorou-se e analisou-se o tema de uma Loja *Online*, as classes foram identificadas e implementadas essencialmente e o prazo estipulado foi cumprido.

Houve a tentativa de implementar mais métodos para as classes *Product, Payment, ShoppingCart* e *Shipping* de modo a, enriquecer o trabalho e a consolidação dos conceitos básicos de POO.

Apesar disto, foi demonstrado como pretendido os conceitos básicos de POO.

7 Bibliografia

1. GitHub do Professor Luís G. Ferreira: LESI-POO-2023-2024;
2. GitHub pessoal com conceitos básicos de C#: CS-Programming-Language;
3. Visual Paradigm: Instalação
4. PcDiga: Website
5. Visual Studio: Instalação
6. Sebenta de C sharp do professor;
7. Sebenta de C sharp de Joe Mayo;