**Gestão de Alojamentos Turísticos**

EST-IPCA Barcelos

LESIPL

Programação Orientada a Objetos

Gilberto Viana Claro

Nº30537

15 de Novembro de 2024

Índice

[**Introdução** 3](#_Toc182588331)

[**Objetivos do projeto** 3](#_Toc182588332)

[**Estrutura do Código** 3](#_Toc182588333)

[**Program.cs** 5](#_Toc182588334)

[**Análise dos conceitos de Programação Orientada a Objetos** 5](#_Toc182588335)

[**Conclusão** 7](#_Toc182588336)

# **Introdução**

O objetivo deste trabalho é desenvolver um projeto para a gestão de alojamentos turísticos que permite a gestão de clientes, proprietários e reservas. Este projeto foi desenvolvido em C#, utilizando conceitos basilares do Paradigma Orientado a Objetos. O projeto foi estruturado em várias classes que representam as entidades principais (Clientes, Alojamentos, Reservas e Proprietários) e uma classe de serviço que permite manipular essas entidades.

# **Objetivos do projeto**

Os principais objetivos deste projeto foram:

* Criar um projeto em C# para a gestão de alojamentos turísticos.
* Aplicar conceitos fundamentais da Programação Orientada a Objetos (POO): abstração, herança, polimorfismo e encapsulamento (4 pilares)
* Permitir criar e gerir clientes e reservas através de uma interface.

# **Estrutura do Código**

**Classes principais**

Foram criadas várias classes:

* **Alojamento** - Representa um alojamento disponível para reserva, com propriedades como Id, nome, endereço, capacidade e preço por noite.
* **Reserva** - Representa uma reserva de um alojamento feita por um cliente, com informações como data de início, data de fim, valor total, e uma referência ao cliente e ao alojamento associados.
* **Cliente** - Representa um cliente que pode realizar reservas, com propriedades como Id, nome, email e telefone.
* **Proprietário** - Representa o proprietário de um alojamento, com as mesmas propriedades do cliente e uma adicional morada.
* **Pagamento** - Representa o pagamento de uma reserva, com propriedades como Id, valor, data de pagamento e uma referência à reserva.

**Classe Abstrata**

Para aplicar o conceito de abstração e herança, foi criada uma classe abstrata “Pessoa”, que define propriedades comuns a clientes e proprietários, como Id, Nome, Email e Telefone. A classe pessoa também inclui um método abstrato “MostrarDetalhes”, que é implementado de forma personalizada nas classes cliente e proprietário.

**Classe de serviço**

A classe “GestaoAlojamentoService” é responsável pela gestão de clientes e reservas. Nela estão incluídos métodos para:

* Adicionar um novo cliente (AdicionarCliente), que cria um cliente e o adiciona a uma lista privada de clientes.
* Adicionar uma nova reserva (AdicionarReserva), que cria uma reserva com base nos dados fornecidos e calcula automaticamente o valor total com base na duração da estadia e no preço por noite.

# **Program.cs**

A execução do programa ocorre no ficheiro program.cs onde é criado um cliente e um proprietário, e as suas informações são mostradas com o método “MostrarDetalhes”

É criado também um alojamento e uma reserva que são mostrados detalhes acerca dos mesmos na consola.

# **Análise dos conceitos de Programação Orientada a Objetos**

O projeto utiliza os quatro pilares de POO da seguinte forma:

**Herança**

A herança é aplicada na classe Pessoa, que é a classe base para Cliente e Proprietário. Com isso, o cliente e proprietário compartilham as propriedades Id, nome, email e telefone, sem a necessidade de redefini-las, aumentando a reutilização e a organização do código.

**Encapsulamento**

O encapsulamento é implementado pela definição de modificadores de acesso, onde as listas de clientes e reservas na classe “GestaoAlojamentoService” são privadas. Isso restringe o acesso direto a esses dados, permitindo manipulação apenas através dos métodos públicos da classe, como AdicionarCliente e AdicionarReserva.

**Abstração**

A classe abstrata Pessoa fornece uma representação genérica de uma "pessoa" no sistema, definindo propriedades e métodos comuns, mas deixando a implementação específica do método “MostrarDetalhes” para as classes derivadas (cliente e proprietário).

**Polimorfismo**

O polimorfismo é aplicado através do método abstrato “MostrarDetalhes” na classe Pessoa. Este método é implementado de forma distinta em Cliente e Proprietário, permitindo que cada tipo de pessoa mostre suas informações específicas ao chamar “MostrarDetalhes”.

**Melhorias futuras**

* **Adicionar Validação de Dados**: Garantir que os dados, como data de início e data do fim, sejam válidos antes de criar uma reserva.
* **Adicionar Funcionalidades**: Incluir funcionalidades para calcular descontos.

# **Conclusão**

Este projeto de gestão de alojamentos turísticos permitiu aplicar os principais conceitos da Programação Orientada a Objetos, como herança, encapsulamento, abstração e polimorfismo. Através de uma estrutura modular em C#, foi possível criar um projeto organizado e escalável, com classes bem definidas para clientes, proprietários, alojamentos e reservas. A implementação demonstra como a POO facilita a manutenção e extensão do código, fornecendo uma base sólida para futuras melhorias e funcionalidades adicionais.