Relatório do trabalho da disciplina de Programação Orientada a Objetos

Gestão de alojamentos turísticos

Tiago Afonso – 20452

Licenciatura em Engenharia de Sistemas Informáticos

Docente: Luís G. Ferreira

Novembro de 2023

# Resumo

O projeto propõe o desenvolvimento de um Sistema de Gestão de Alojamentos Turísticos utilizando Programação Orientada a Objetos em C# (C Sharp). O projeto pretende oferecer uma plataforma abrangente para a administração eficiente de propriedades turísticas, focando em registos, consultas, reservas, check-in, gestão de clientes e alojamentos.

**Índice**

[Resumo I](#_Toc150984926)

[Introdução 5](#_Toc150984927)

[Objetivos do Projeto: 5](#_Toc150984928)

[Arquitetura do projeto 6](#_Toc150984929)

[Alojamentos DDL 7](#_Toc150984930)

[Classe Propriedade 7](#_Toc150984931)

[Classe Propriedades 9](#_Toc150984932)

[Pessoas DLL 13](#_Toc150984933)

[Gestão Turística DLL 14](#_Toc150984934)

[Classe Reserva 14](#_Toc150984935)

**Lista de Figuras**

[Figura 1 - Divisão do projeto 6](#_Toc150984819)

[Figura 2 - Diagrama de Classes - Propriedade 7](file:///C:\Users\tiafo\Desktop\LESI\A2_S1\POO\TP_20452_FASE1\Relatorio_POO_20452.docx#_Toc150984820)

[Figura 3 - Atributos de uma Propriedade 7](#_Toc150984821)

[Figura 4 - Construtor de uma Propriedade 8](#_Toc150984822)

[Figura 5 - Propriedades de uma Propriedade 8](#_Toc150984823)

[Figura 6 - Override Propriedade 9](file:///C:\Users\tiafo\Desktop\LESI\A2_S1\POO\TP_20452_FASE1\Relatorio_POO_20452.docx#_Toc150984824)

[Figura 7 - Diagrama de Classes - Propriedades 9](#_Toc150984825)

[Figura 8 - Atributos Propriedades (Array) 10](#_Toc150984826)

[Figura 9 - Construtor Array Propriedades 10](#_Toc150984827)

[Figura 10 - Propriedades da classe Propriedades 10](#_Toc150984828)

[Figura 11 - OUTROS METODOS - > ShowPropriedades 11](#_Toc150984829)

[Figura 12 - OUTROS METODOS -> ExistePropriedade 11](#_Toc150984830)

[Figura 13 - OUTROS METODOS -> InserirPropriedade 12](#_Toc150984831)

[Figura 14 - OUTROS METODOS -> Verificadisponibilidade 12](#_Toc150984832)

[Figura 15 - Diagrama de Classes Pessoas 13](#_Toc150984833)

[Figura 16 - Diagrama de Classes de uma Reserva 14](#_Toc150984834)

[Figura 17 - Atributos da Classe Reserva 14](#_Toc150984835)

[Figura 18 - Construtor de uma Reserva 15](#_Toc150984836)

# Introdução

Na era dinâmica e cada vez mais conectada do turismo, a eficaz gestão de alojamentos turísticos é essencial para proporcionar experiências positivas aos viajantes. O projeto em questão visa desenvolver um Sistema de Gestão de Alojamentos Turísticos, utilizando linguagem C# (C Sharp).

A indústria hoteleira enfrenta desafios constantes na busca por uma gestão eficiente, que abrange desde o registo preciso de informações até a facilitação do processo de reservas e check-in. Este projeto emerge como resposta a essa necessidade, propondo uma solução para a administração de alojamentos turísticos.

O Sistema de Gestão proposto não apenas simplificará tarefas operacionais, como também proporcionará uma experiência aprimorada tanto para os gestores quanto para os hóspedes.

Para é proposto alcançar alguns objetivos.

## Objetivos do Projeto:

* **Eficiência Operacional:** Desenvolver um sistema que otimize a gestão diária de alojamentos, reduzindo redundâncias e automatizando processos.
* **Experiência do Cliente:** Aprimorar a experiência dos clientes, permitindo reservas intuitivas, check-ins rápidos e acesso facilitado a informações relevantes.
* **Registro Detalhado:** Garantir um registo detalhado e seguro de dados, abrangendo informações dos clientes, propriedade e condições dos alojamentos.
* **Versatilidade:** Criar um sistema adaptável, capaz de gerenciar diferentes tipos de alojamentos e se ajustar às necessidades específicas de diversos estabelecimentos.

# Arquitetura do projeto

Este projeto está planeado para fazer uma divisão por Bibliotecas onde serão inseridas as classes fundamentais para a gestão da aplicação.

Estas divisões abrangem classes fundamentais como:

* Cliente
* Gestor
* Propriedades
* Reserva
* Check-in

Para além destas classes fez-se também uma classe para inserir todos os objetos criados na sua classe por exemplo:

* Clientes
* Propriedades
* Reservas

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

Descrição gerada automaticamente

Figura - Divisão do projeto

Na **Figura 1** observa-se que nesta primeira fase foi criada uma pasta “DLL” para armazenar todas as bibliotecas criadas onde ser guardaram as classes, e tem uma aplicação consola para gerir e incrementar os métodos.

Esta divisão foi feita em 3 bibliotecas:

1. **Alojamentos:** contém a classe **Propriedade** e **Propriedades.**
2. **Gestão turística:** contém a classe **Reserva**, **Reservas** e **Check-in** (por implementar).
3. **Pessoas:** contém a classe **Cliente**, **Clientes** e **Gestor**.

## Alojamentos DDL

DLL criada para alocar as classes de Propriedade e Propriedades.

### Classe Propriedade

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

Descrição gerada automaticamenteSobre a **Figura 2** podesse observar que os atributos **privados** dados a uma Propriedade estão inseridos na Tabela **Campos**, sendo estes:

* Agregado – número de pessoas que podem utilizar a propriedade.
* Codpropriedade – ID da propriedade.
* Disponibilidade – Disponivel ou não disponível.
* Localização
* Nome
* Preconoite – Preço por noite
* Tipo – Casa Rural, Bangalô, etc…

A tabela **Propriedades** é responsável por alocar os métodos publicos para se possa ler e atribuir valores às variáveis privadas antes expostas.

A tabela **Métodos** reune funções de Construtor de um Objeto do tipo Propriedade, e o método “To String” que permite o retorno de uma string formatada com todas as informações da Propriedade.  
Seguem-se imagens do código produzido nestas secções.

Figura - Diagrama de Classes - Propriedade

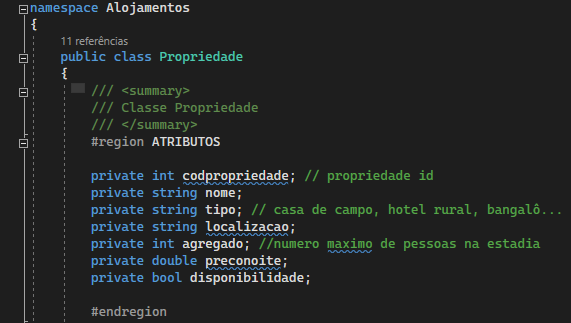


Figura 3 - Atributos de uma Propriedade

Na Figura 3, observa-se que os atributos estão todos privados e com identificação se é inteiro, double, string ou bool.

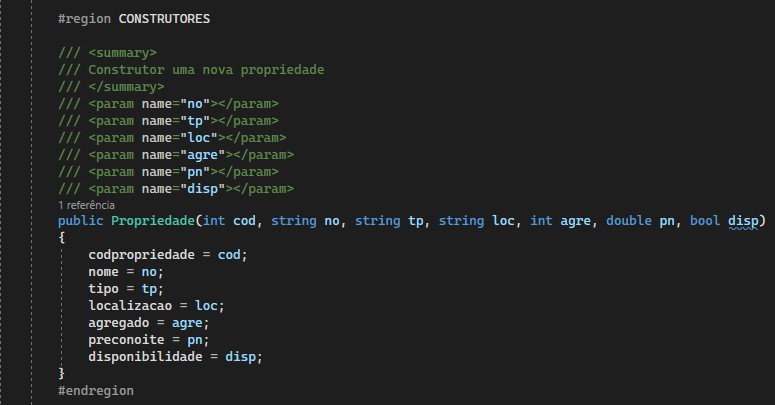


Figura 4 - Construtor de uma Propriedade

Na **Figura 4** existe o método **public Propriedade**, que permite ao utilizador criar uma nova Propriedade baseada nos valores inseridos.

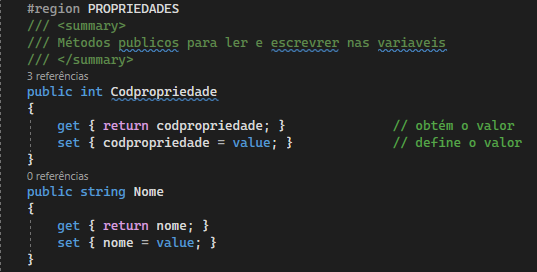


Figura 5 - Propriedades de uma Propriedade

Na **Figura 5** estão representados os métodos públicos para obter e definir os valores das variáveis, realçando que as restantes variáveis não apresentadas utilizam o mesmo método.

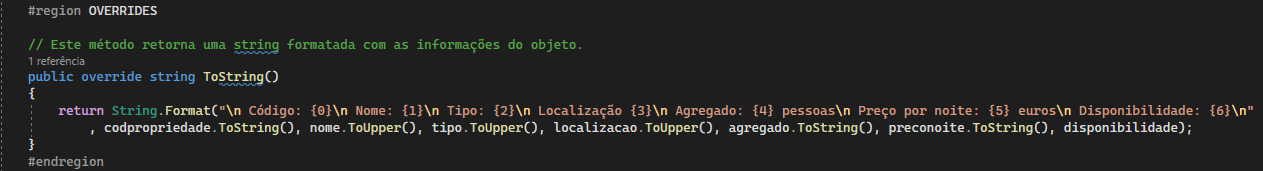
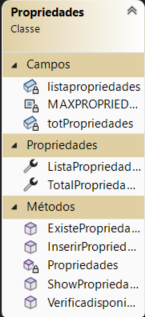
Na Região **OVERRIDES (Figura 6)** foi construído um método para apresentar uma string formatada das informações referentes à Propriedade, mostrando por linhas separadas a variável e o seu respetivo valor.

Figura - Override Propriedade

### Classe Propriedades

Esta classe tem como desafio alocar todas as Propriedades criadas na classe anterior num array de objetos.

Na **Figura 7** observa-se que como atributos esta classe tem:

* **listadepropriedades** – array de Propriedades(objeto)
* **MAXPROPRIEDADES** – uma constante que define um número máximo de 100 objetos que podem ser guardados.
* **totPropriedades** – uma varável que será utilizada pela classe.

Na **REGIÃO PROPRIEDADES,** existem dois métodos para tornar públicos os valores do array lista de propriedades (clone) e totPropriedades.

Na região Métodos verificam-se as funções:

* ExistePropriedade
* InserirPropriedade
* Propriedades
* ShowPropriedades
* VerificaDisponibilidade

Figura 7 - Diagrama de Classes - Propriedades

Na **REGIÃO ATRIBUTOS** estão inseridos os atributos como static da própria classe.

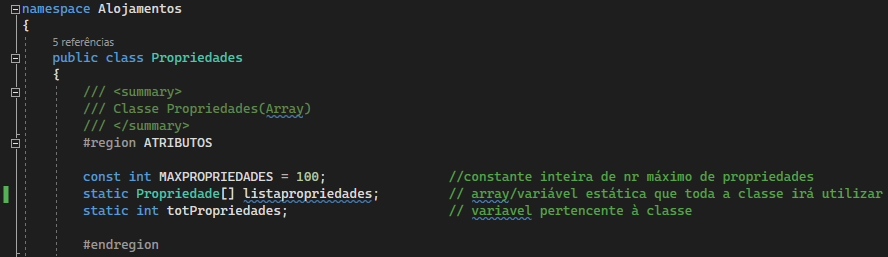


Figura - Atributos Propriedades (Array)

Na **Figura 8** verifica-se as variáveis const para não poder ser alterada, e static para permanecerem na classe.

Na **REGIÃO CONSTRUTORES** está inserido o método de criação de um array.

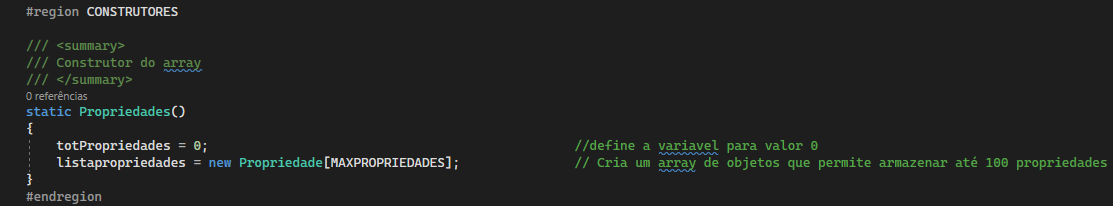


Figura - Construtor Array Propriedades

Na **Figura 9** observa-se que o método **Propriedades()** define o valor da variável totPropriedades para iniciar em 0, e cria um array de propriedades com o valor máximo de 100 apresentado anteriormente.

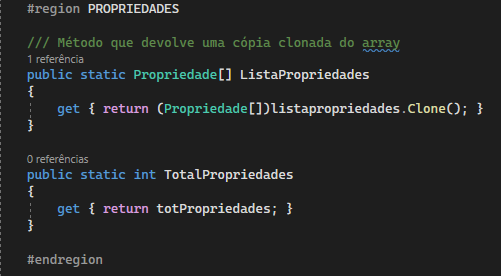


Figura 10 - Propriedades da classe Propriedades

Na **REGIÃO PROPRIEDADES (FIGURA 10)** tem o método que obtém uma cópia do array e também a variável totPropriedades.

Por fim na **REGIÃO OUTROS MÉTODOS**, implementou-se as funções anteriormente apresentadas.

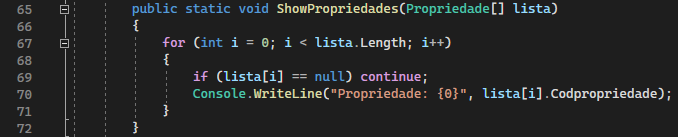


Figura 11 - OUTROS METODOS - > ShowPropriedades

Na **Figura 11** a função inicia um Loop sobre todos os elementos do array **(linha 67)**, verifica se o elemento do array é nulo, caso seja avança para o próximo **(linha 69)** e imprime na consola os códigos de propriedades existentes **(linha 70)**.

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

Descrição gerada automaticamente

Figura 12 - OUTROS METODOS -> ExistePropriedade

Na **Figura 12** baseado no código da propriedade fornecido **(linha 75)**, procura um objeto no array (lista propriedades) **(linha77)**, caso o codigo da propriedade não seja nulo e esteja no array então devolve "true" **(linhas 79 e 81)**, caso contrário devolve “false”.

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

Descrição gerada automaticamente

Figura - OUTROS METODOS -> InserirPropriedade

O método acima apresentado **(Figura 13)**, utiliza o método anteriormente criado “ExistePropriedade” para verificar se o codigo da nova propriedade já existe, se já existe ele devolve o valor “false” **(linhas 91 e 93),** caso não exista ele adiciona a propriedade ao array e incrementa o contador de propriedades, dando o valor de “true” **(linhas 97 a 99)**.

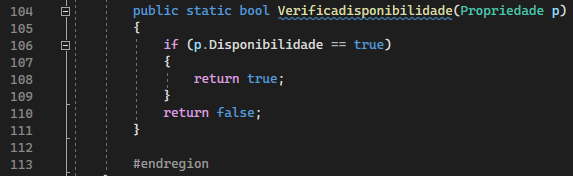


Figura 14 - OUTROS METODOS -> Verificadisponibilidade

Este método **(Figura 14)** verifica se uma propriedade está disponível ou não verificando se na sua variável “disponibilidade” está atribuído o valor true (disponível) ou false (indisponível).

## Pessoas DLL

Biblioteca criada para armazenar todas as classes referentes a pessoas tais como, Cliente, Clientes (array), Gestor.

No próximo Diagrama de Classes verifica-se as 3 classes desta Biblioteca.

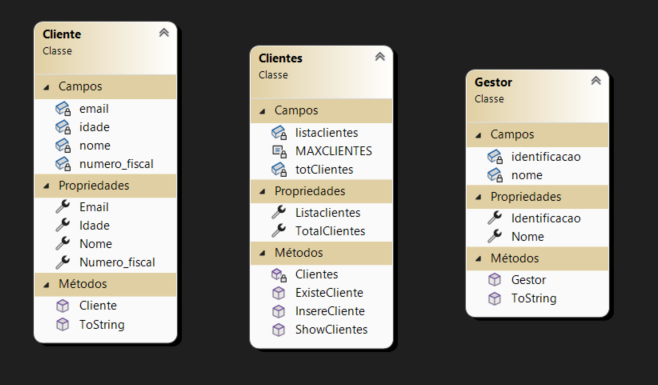


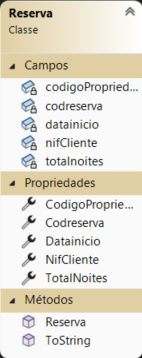
Figura 15 - Diagrama de Classes Pessoas

Na **Figura 15** observa-se que a estrutura utilizada para construir cada classe é semelhante à anteriormente apresentada (Propriedade), mudando apenas os nomes e tipos de variáveis, mas utilizando os mesmos métodos anteriormente referidos.

## Gestão Turística DLL

Nesta Biblioteca estão alojadas as classes de Reserva e Reservas.

### Classe Reserva

Na **Figura 16** observa-se que a classe Reserva tem como seus atributos os as seguintes variáveis:

* codigoPropriedade – referente ao codigo de uma propriedade registada.
* codreserva – codigo que ficará associado à reserva.
* datainicio – data para o aluguer da propriedade.
* nifCliente – Número Fiscal do Cliente que pretende alugar a Propriedade.
* Totalnoites – número de noites que o Cliente pretende ficar.

Estão presentes também as suas Propriedades para obter os valores públicos, e os Métodos para criar uma reserva e representar a mesma.

Figura 16 - Diagrama de Classes de uma Reserva

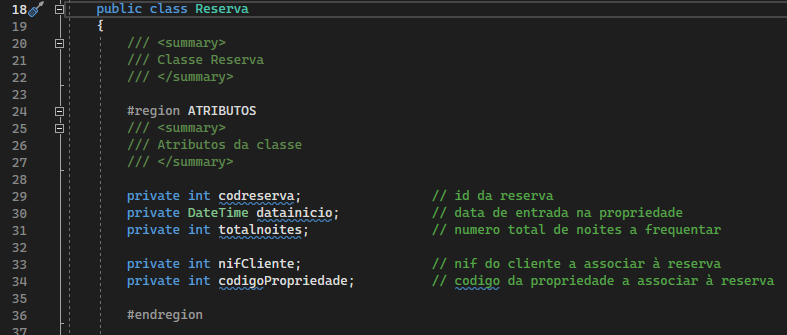


Figura 17 - Atributos da Classe Reserva

Na **Figura 17** observa-se as variáveis criadas para atribuir a uma reserva bem como as variáveis que serão referentes ao cliente e propriedade pertencentes a essa reserva.

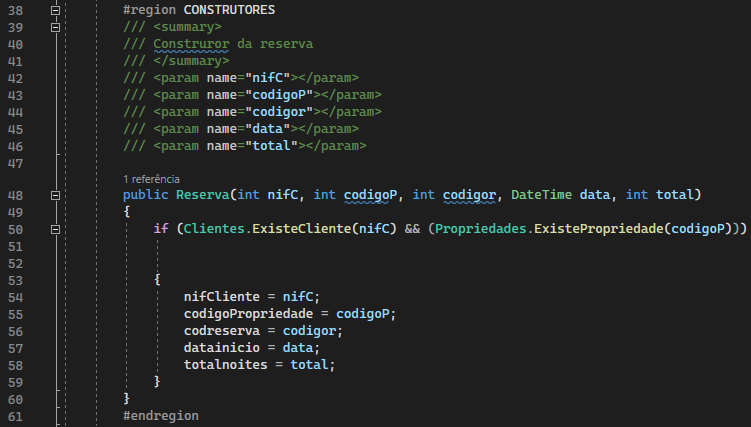


Figura 18 - Construtor de uma Reserva

Na Figura 18 observa-se que a **função Reserva()**, para além das varáveis fornecidas, esta verifica se a variável **nifC** (correspondente ao Número Fiscal do Cliente) e a variável **codigoP** (correspondente ao codigo de uma propriedade) estão presentes nos arrays dessas mesmas classes. **(linha 50).**

Caso essa verificação seja bem sucedida ele adiciona todos os valores fornecidos à respetiva RESERVA.