Programação Orientada a Objetos

Relatório de Trabalho Prático

Danilo Castro - 25457

Engenharia de Sistemas Informáticos

Novembro 2024

ESCOLA SUPERIOR DE TECNOLOGIA IPCA

Índice

Conteúdo

Lista de Figuras

Introdução

Este trabalho aborda o desenvolvimento de um sistema em C# dedicado à gestão de um alojamento turístico. A fase inicial do projeto foca na identificação e implementação das classes essenciais, além da definição das estruturas de dados fundamentais.

Este projeto tem como objetivo desenvolver uma aplicação que permita a administração eficiente de um alojamento turístico, englobando funcionalidades essenciais como a gestão de clientes, reservas, check-ins e check-outs, alojamentos e avaliações.

Através de uma interface intuitiva desenvolvida em Windows Forms, o sistema é projetado para proporcionar aos utilizadores (administradores e funcionários do alojamento) uma plataforma centralizada e organizada para realizar operações diárias, facilitando o acompanhamento das reservas, o atendimento aos clientes e a análise de feedback.

O projeto será desenvolvido utilizando princípios de programação orientada a objetos, com classes específicas para cada funcionalidade (tais como Client, Accommodation, Reservation, entre outras). Além disso, foi implementada uma estrutura modular, que inclui uma DLL personalizada, permitindo o cálculo de custos e a verificação de disponibilidade, além de métodos auxiliares para otimizar a gestão de informações de clientes e reservas.

A documentação gerada não só descreve as classes e estruturas de dados, mas também explicaa lógica subjacente a cada componente.

Estrutura do Código

O código fonte da aplicação Tourism_Accommodation_System.csproj contém os seguintes ficheiros:

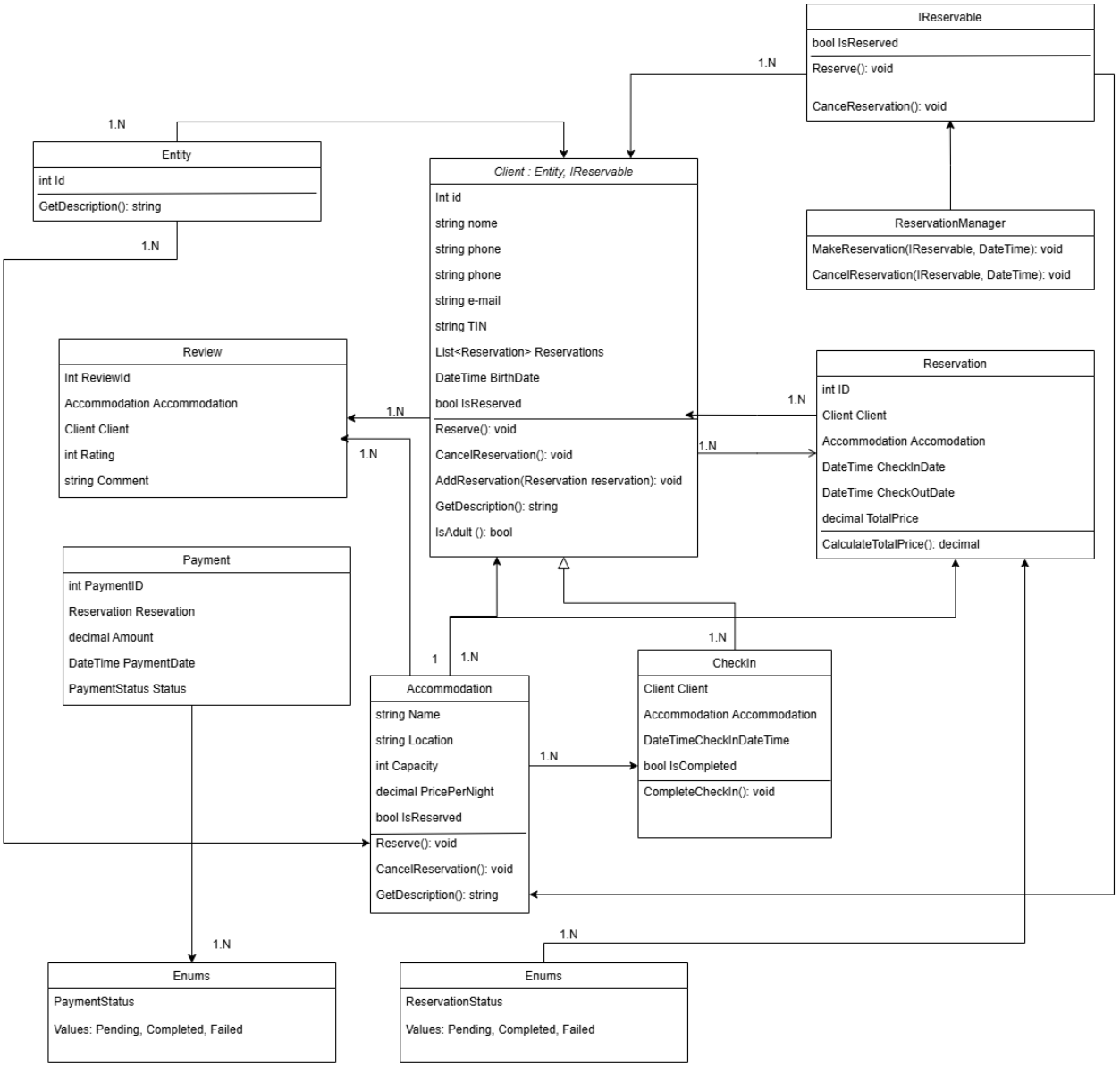
- Models
  - Program.cs
  - Client.cs
  - Accommodation.cs
  - Reservation.cs
  - Review.cs
  - Entity,cs
  - Check-in.cs
  - Payment.cs
  - Enums.cs
  - IReservable.cs

- Services
  - ReservationManager.cs
  - Payment.cs

- Forms
  - MainWindow.cs
  - ClientManagemtForm.cs
  - Accommodation.cs
  - CheckInForm.cs
  - ReviewForm.cs

- UserControls
  - UserControlClient.cs
  - UserControlAccommodation.cs
  - UserControlReservation.cs
  - UserControlCheckin.cs
  - UserControlReview.cs
  - UserControlClientPayment.cs

- Utilities
  - TourismUtilities.dll


IReservable
bool IsReserved
Reserve(): void
CanceReservation(): void
1.N
1.N
Entity
int Id
GetDescription(): string
1.N
Client : Entity, IReservable
Int id
string nome
string phone
string phone
string e-mail
string TIN
List<Reservation> Reservations
DateTime BirthDate
bool IsReserved
Reserve(): void
CancelReservation(): void
AddReservation(Reservation reservation): void
GetDescription(): string
IsAdult (): bool
ReservationManager
MakeReservation(IReservable, DateTime): void
CancelReservation(IReservable, DateTime): void
Review
Int ReviewId
Accommodation Accommodation
Client Client
int Rating
string Comment
1.N
1.N
Reservation
int ID
Client Client
Accommodation Accomodation
DateTime CheckInDate
DateTime CheckOutDate
decimal TotalPrice
CalculateTotalPrice(): decimal
1.N
1.N
Payment
int PaymentID
Reservation Resevation
decimal Amount
DateTime PaymentDate
PaymentStatus Status
1
1.N
1.N
CheckIn
Client Client
Accommodation Accommodation
DateTimeCheckInDateTime
bool IsCompleted
CompleteCheckIn(): void
Accommodation
string Name
string Location
int Capacity
decimal PricePerNight
bool IsReserved
Reserve(): void
CancelReservation(): void
GetDescription(): string
1.N
1.N
1.N
Enums
PaymentStatus
Values: Pending, Completed, Failed
1.N
Enums
ReservationStatus
Values: Pending, Completed, Failed

Classes e Subclasses

1. Program.cs

A classe Program contém o método Main, que serve como ponto de entrada para a execução do programa. O código dentro do método realiza as seguintes operações:

1. ApplicationConfiguration.Initialize(); é usado para definir configurações da aplicação, como DPI e fontes padrão.
2. Application.Run(new MainWindow()); abre o MainWindow, que é o formulário principal e a interface inicial para o utilizador.

2. Entity.cs

A classe entity é uma classe base abstrata no sistema de gestão de alojamentos turísticos. Ela fornece um identificador único (Id) para cada entidade que herda dela, garantindo consistência na identificação de objetos no sistema. Além disso, Entity define um método abstrato GetDescription() que deve ser implementado pelas subclasses para fornecer uma descrição básica da entidade. Esta estrutura facilita a organização e permite que outras classes, como Client e Accommodation, herdem características comuns.

Propriedades:

1. int Id: Identificador único para cada entidade no sistema. Esta propriedade é crucial para diferenciar cada instância das classes que herdam de Entity.

Métodos:

- GetDescription() : string (Método Abstrato):
  - Método abstrato que será implementado por todas as subclasses de Entity. Este método retorna uma descrição básica da entidade com o objetivo de permitir que cada classe filha forneça a sua própria descrição, de acordo com as suas características específicas. Por exemplo, Client pode retornar uma descrição com o nome e o e-mail do cliente, enquanto Accommodation pode descrever o nome e a localização do alojamento.

3. Client.cs

A classe Client representa um cliente no sistema de gestão de alojamentos turísticos. Cada cliente possui informações pessoais e dados de contato, além de uma lista de reservas associadas. Esta classe herda de Entity (classe abstrata) para garantir um identificador único (Id) e implementa a interface IReservable, permitindo que o cliente seja marcado como reservado ou não.

Propriedades:

1. int Id: Identificador único herdado de Entity, que permite distinguir cada cliente.
2. string Name: Nome completo do cliente.
3. string Email: Endereço de e-mail para contato.
4. string Phone: Número de telefone do cliente.
5. string TIN: Número de identificação fiscal (NIF) do cliente.
6. DateTime BirthDate: Data de nascimento do cliente, usada para validação de idade.
7. List<Reservation> Reservations: Lista de reservas associadas ao cliente.
8. bool IsReserved: Indica se o cliente possui uma reserva ativa, conforme a implementação de IReservable.

```
using Tourist_Accommodation_System.Interfaces;
using TourismUtilities;

namespace Tourist_Accommodation_System.Models
{
    7 referências
    public class Client : Entity
    {
        1 referência
        public int Id { get; set; }
        2 referências
        public string Name { get; set; }
        2 referências
        public string Email { get; set; }
        1 referência
        public string Phone { get; set; }
        1 referência
        public string TIN { get; set; } //nif
        1 referência
        public DateTime BirthDate { get; set; }
        2 referências
        public List<Reservation> Reservations { get; set; }

        // Construtor dos objetos do cliente
        0 referências
        public Client(int id, string name, string email, string phone, string tin, DateTime birthDate)
        {
            Id = id;
            Name = name;
            Email = email;
            Phone = phone;
            TIN = tin;
            BirthDate = birthDate;
            Reservations = new List<Reservation>();
        }
```

Métodos:

- GetDescription() : string:
    - Retorna uma breve descrição do cliente, incluindo o nome e o e-mail facilitando a obtenção de uma visão geral do cliente, útil para exibição em interfaces de lista ou consultas.
- AddReservation(Reservation reservation):
    - Adiciona uma nova reserva à lista de reservas do cliente e permite associar reservas ao cliente, facilitando o histórico e a gestão de múltiplas reservas.
- IsAdult() : bool:
    - Verifica se o cliente é maior de idade (18 anos ou mais), com base na sua data de nascimento e assegura que apenas clientes com idade suficiente possam fazer reservas, de acordo com as políticas do sistema.
- Reserve():
    - Implementação do método da interface IReservable, que marca o cliente como tendo uma reserva ativa (IsReserved = true). Facilita a atualização do estado de reserva do cliente, integrando-o com a lógica de reservas no sistema.
- CancelReservation():
    - Implementação do método CancelReservation de IReservable, que cancela a reserva ativa do cliente (IsReserved = false), permitindo cancelar o estado de reserva do cliente, essencial para a gestão de cancelamentos.

```
#region
1 referência
public override string GetDescription()
{
    return $"Client: {Name}, Email: {Email}";
}

//metodo para adicionar reserva
0 referências
public void AddReservation(Reservation reservation)
{
    Reservations.Add(reservation);
}
#endregion

#region
2 referências
public bool IsReserved { get; private set; }

0 referências
public void Reserve()
{
    // Lógica para reservar para o cliente
    IsReserved = true;
    Console.WriteLine("Client reservation made.");
}

0 referências
public void CancelReservation()
{
    // Lógica para cancelar a reserva do cliente
    IsReserved = false;
    Console.WriteLine("Client reservation canceled.");
}
#endregion
```

4. Accommodation.cs

A classe Accommodation representa um alojamento no sistema de gestão de alojamentos turísticos. Cada instância da classe caracteriza-se por atributos como o nome, localização, capacidade e preço por noite, que permitem definir as especificações de cada alojamento disponível. Assim como a classe Client herda de Entity, garantindo um identificador único (Id), e implementa a interface IReservable, possibilitando a reserva e o cancelamento de reservas do alojamento.

Propriedades:

1. int Id: Identificador único do alojamento, herdado de Entity.
2. string Name: Nome do alojamento (ex.: "Apartamento Vista Mar").
3. string Location: Localização geográfica do alojamento.
4. int Capacity: Capacidade máxima de hóspedes que o alojamento pode receber.
5. decimal PricePerNight: Preço por noite de estadia no alojamento.
6. bool IsReserved: Indicador de reserva ativa ou não do alojamento, conforme a interface IReservable.

Métodos:

- GetDescription() : string:

- Retorna uma breve descrição do alojamento, incluindo o nome, a localização e o preço por noite facilitando a exibição do alojamento em listas ou consultas, permitindo uma visualização rápida das informações essenciais.

➢ Reserve():

- Implementação do método da interface IReservable, que marca o alojamento como reservado (IsReserved = true) atualizando o estado de reserva do alojamento, integrando-o com a lógica de reservas e garantindo a gestão correta de disponibilidade.

➢ CancelReservation():

- Implementação do método CancelReservation da interface IReservable, que cancela o estado de reserva ativa do alojamento (IsReserved = false) permitindo assim cancelar a reserva do alojamento, facilitando a gestão de reservas e cancelamentos.

```
using Tourist_Accommodation_System.Interfaces;
using TourismUtilities;

namespace Tourist_Accommodation_System.Models
{
    7 referências
    public class Accommodation : Entity
    {
        1 referência
        public int Id { get; set; }  // Identificador único do alojamento
        2 referências
        public string Name { get; set; }  // Nome do alojamento
        2 referências
        public string Location { get; set; }  // Localização do alojamento
        1 referência
        public int Capacity { get; set; }  // Capacidade máxima de pessoas
        3 referências
        public decimal PricePerNight { get; set; }  // Preço por noite

        #region
        // Construtor para inicializar um alojamento
        0 referências
        public Accommodation(int id, string name, string location, int capacity, decimal pricePerNight)
        {
            Id = id;
            Name = name;
            Location = location;
            Capacity = capacity;
            PricePerNight = pricePerNight;
        }
        1 referência
        public override string GetDescription()
        {
            return $"Accommodation: {Name} in {Location}, Price per Night: {PricePerNight}";
        }
        #endregion

        2 referências
        public bool IsReserved { get; private set; }
        0 referências
        public void Reserve() { IsReserved = true; }
        0 referências
        public void CancelReservation() { IsReserved = false; }
```

5. Reservation

A classe Reservation representa uma reserva no sistema de gestão de alojamentos turísticos. Cada reserva está associada a um cliente e a um alojamento específico, e inclui informações sobre as datas de check-in e check-out, bem como o custo total da estadia. A classe Reservation encapsula a lógica para calcular o preço total com base na duração da estadia e no preço por noite do alojamento.

Propriedades:

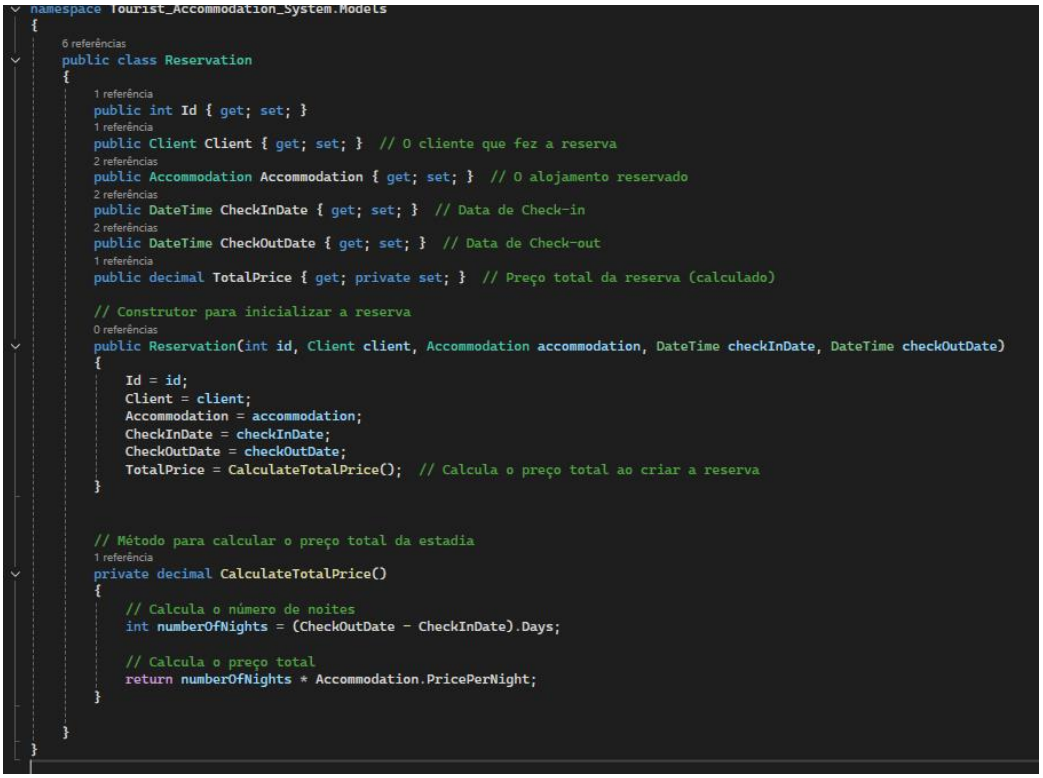
1. int Id: Identificador único da reserva.
2. Client Client: Cliente que efetuou a reserva, permitindo associar a reserva a um cliente específico.
3. Accommodation Accommodation: Alojamento reservado, ligando a reserva a um alojamento específico.
4. DateTime CheckInDate: Data de check-in, indicando o início da estadia.
5. DateTime CheckOutDate: Data de check-out, indicando o final da estadia.
6. decimal TotalPrice: Preço total da reserva, calculado automaticamente com base nas noites e no preço por noite do alojamento.

Métodos:

- CalculateTotalPrice() : decimal:
    - Método privado que calcula o custo total da estadia multiplicando o número de noites pelo preço por noite do alojamento. Automatiza assim o cálculo do custo total de uma reserva, essencial para gerir os valores a pagar pelo cliente e garantir a precisão no cálculo do preço.

Construtor

- Reservation(int id, Client client, Accommodation accommodation, DateTime checkInDate, DateTime checkOutDate):
    - Inicializa uma nova reserva, associando-a a um cliente e um alojamento, e definindo as datas de check-in e check-out. Durante a criação, o TotalPrice é calculado automaticamente chamando CalculateTotalPrice(). Facilita a criação de reservas com dados completos e calcula automaticamente o preço total ao inicializar a reserva.



```
namespace Tourist_Accommodation_System.Models
{
    6 referências
    public class Reservation
    {
        1 referência
        public int Id { get; set; }
        1 referência
        public Client Client { get; set; }  // O cliente que fez a reserva
        2 referências
        public Accommodation Accommodation { get; set; }  // O alojamento reservado
        2 referências
        public DateTime CheckInDate { get; set; }  // Data de Check-in
        2 referências
        public DateTime CheckOutDate { get; set; }  // Data de Check-out
        1 referência
        public decimal TotalPrice { get; private set; }  // Preço total da reserva (calculado)

        // Construtor para inicializar a reserva
        0 referências
        public Reservation(int id, Client client, Accommodation accommodation, DateTime checkInDate, DateTime checkOutDate)
        {
            Id = id;
            Client = client;
            Accommodation = accommodation;
            CheckInDate = checkInDate;
            CheckOutDate = checkOutDate;
            TotalPrice = CalculateTotalPrice();  // Calcula o preço total ao criar a reserva
        }


        // Método para calcular o preço total da estadia
        1 referência
        private decimal CalculateTotalPrice()
        {
            // Calcula o número de noites
            int numberOfNights = (CheckOutDate - CheckInDate).Days;

            // Calcula o preço total
            return numberOfNights * Accommodation.PricePerNight;
        }

    }
}
```

6. Review.cs

A classe Review representa uma avaliação realizada por um cliente sobre um alojamento no sistema de gestão de alojamentos turísticos. Cada avaliação inclui informações sobre o cliente que fez a avaliação, o alojamento avaliado, uma classificação numérica (rating) e um comentário. A classe Review permite registar o feedback dos clientes, ajudando a monitorizar a qualidade dos alojamentos e melhorar a experiência dos futuros hóspedes.

Propriedades:

1. int ReviewId: Identificador único da avaliação.
2. Client Client: Cliente que fez a avaliação, associando a avaliação a um cliente específico.
3. Accommodation Accommodation: Alojamento avaliado, associando a avaliação a um alojamento específico.
4. int Rating: Classificação atribuída ao alojamento (geralmente entre 1 e 5), que representa a satisfação do cliente.
5. string Comment: Comentário adicional do cliente, detalhando a sua experiência no alojamento.

Construtor:

- Review(Client client, Accommodation accommodation, int rating, string comment):
    - Inicializa uma nova avaliação, associando-a ao cliente e ao alojamento, e definindo o rating e o comentário. Auxilia a criação de avaliações completas com informações detalhadas sobre o cliente e o alojamento, bem como a satisfação do cliente.

```
namespace Tourist_Accommodation_System.Models
{
    1 referência
    public class Review
    {
        0 referências
        public int ReviewId { get; set; }
        1 referência
        public Client Client { get; set; }
        1 referência
        public Accommodation Accommodation { get; set; }
        1 referência
        public int Rating { get; set; }
        1 referência
        public string Comment { get; set; }

        0 referências
        public Review(Client client, Accommodation accommodation, int rating, string comment)
        {
            Client = client;
            Accommodation = accommodation;
            Rating = rating;
            Comment = comment;
        }
    }
}
```

7. Check-in.cs

A classe CheckIn representa o processo de check-in de um cliente num alojamento no sistema de gestão de alojamentos turísticos. Esta classe inclui informações sobre o cliente, o alojamento e a data/hora do check-in. Além disso, fornece uma funcionalidade para marcar o check-in como concluído, mantendo o estado atualizado do processo de entrada dos hóspedes.

Propriedades:

1. Client Client: Representa o cliente que realiza o check-in, associando o processo a uma pessoa específica.
2. Accommodation Accommodation: Indica o alojamento onde o cliente está a realizar o check-in.
3. DateTime CheckInDateTime: Registra a data e hora em que o check-in foi realizado, permitindo o acompanhamento cronológico do processo.

4. bool IsCompleted: Indica se o check-in foi concluído ou não. Por padrão, o valor é definido como false e atualizado para true quando o check-in é finalizado.

Métodos:

1. CompleteCheckIn():
    - **Descrição**: Método que marca o check-in como concluído, definindo a propriedade IsCompleted como true.
    - **Objetivo**: Atualizar o estado do check-in para indicar que o cliente finalizou o processo de entrada no alojamento.

Construtor:

- CheckIn(Client client, Accommodation accommodation, DateTime checkInDateTime):
    - Inicializa uma nova instância de CheckIn, associando o cliente e o alojamento, além de definir a data e hora do check-in. O campo IsCompleted é definido como false por padrão. Ajuda assim na criação de um processo de check-in completo, com as informações necessárias para registrar o evento.

```
1 referência
public class CheckIn
{
    // Properties
    1 referência
    public Client Client { get; set; }
    1 referência
    public Accommodation Accommodation { get; set; }
    1 referência
    public DateTime CheckInDateTime { get; set; }
    2 referências
    public bool IsCompleted { get; set; }

    // Constructor
    0 referências
    public CheckIn(Client client, Accommodation accommodation, DateTime checkInDateTime)
    {
        Client = client;
        Accommodation = accommodation;
        CheckInDateTime = checkInDateTime;
        IsCompleted = false; // Por padrão, o check-in é inicialmente não concluído
    }

    // Método para concluir o check-in
    0 referências
    public void CompleteCheckIn()
    {
        IsCompleted = true;
    }
}
```

8. Payment.cs

A classe Payment representa um pagamento realizado no sistema de gestão de alojamentos turísticos, associando o pagamento a uma reserva específica e mantendo o

registo do montante, data e estado do pagamento. Esta classe é útil para gerir transações financeiras associadas às reservas.

Propriedades:

1. int PaymentId: Identificador único do pagamento.
2. Reservation Reservation: Reserva associada ao pagamento, ligando o pagamento a uma transação específica.
3. decimal Amount: Valor do pagamento realizado.
4. DateTime PaymentDate: Data do pagamento, indicando quando a transação foi efetuada.
5. PaymentStatus Status: Estado do pagamento, definido por uma enumeração (PaymentStatus) com valores como Pending, Completed e Failed.

Construtor:

- Payment(int paymentId, Reservation reservation, decimal amount, DateTime paymentDate, PaymentStatus status):
    - Inicializa uma nova instância de Payment, associando-a a uma reserva e definindo o valor, data e estado do pagamento. Facilita a criação de pagamentos completos com todas as informações necessárias para registar e gerir transações.

```
namespace Tourist_Accommodation_System.Models
{
    1 referência
    public class Payment
    {
        1 referência
        public int PaymentId { get; set; }  // Identificador único do pagamento
        1 referência
        public Reservation Reservation { get; set; }  // Reserva associada ao pagamento
        1 referência
        public decimal Amount { get; set; }  // Quantidade paga
        1 referência
        public DateTime PaymentDate { get; set; }  // Data do pagamento
        1 referência
        public PaymentStatus Status { get; set; }  // Estado do pagamento

        // Construtor para inicializar o pagamento
        0 referências
        public Payment(int paymentId, Reservation reservation, decimal amount, DateTime paymentDate, PaymentStatus status)
        {
            PaymentId = paymentId;
            Reservation = reservation;
            Amount = amount;
            PaymentDate = paymentDate;
            Status = status;
        }
    }
}
```

9. Enums.cs

As enumerações são usadas no sistema para representar estados e tipos, facilitando a organização e a consistência dos dados. Abaixo estão as enums utilizadas no projeto.

A enumeração PaymentStatus representa os possíveis estados de um pagamento no sistema, facilitando o acompanhamento e a gestão do processo de pagamento.

Valores:

- Pending: Indica que o pagamento está pendente.
- Completed: Indica que o pagamento foi concluído com sucesso.
- Failed: Indica que o pagamento falhou.

Esta enumeração pode ser usada para categorizar tipos de alojamentos no sistema, permitindo que diferentes tipos de alojamentos sejam identificados e geridos facilmente.

Valores:

- Room: Representa um quarto.
- Apartment: Representa um apartamento.
- House: Representa uma casa.

```
namespace Tourist_Accommodation_System.Models
{
    2 referências
    public enum PaymentStatus
    {
        Pending,
        Completed,
        Failed
    }

    0 referências
    public enum ReservationStatus
    {
        Pending,
        Confirmed,
        Cancelled
    }

    0 referências
    public enum AccommodationType
    {
        Room,
        Apartment,
        House
    }
}
```

10. IReservable.cs

A interface IReservable define um conjunto de métodos e propriedades que qualquer classe reservável (como Client e Accommodation) deve implementar. Ao padronizar esses métodos e propriedades, a interface IReservable permite que o sistema trate de forma consistente qualquer entidade que possa ser reservada ou cuja reserva possa ser cancelada.

Propriedades:

1. bool IsReserved: Indica se a entidade (ex.: cliente ou alojamento) possui uma reserva ativa ou não.

Métodos:

- Reserve():
  - Método que marca a entidade como reservada com o objetivo de permitir que qualquer classe que implemente IReservable possa ser marcada como reservada, alterando o estado de IsReserved para true.
- CancelReservation():
  - Método que cancela a reserva, marcando a entidade como não reservada permitindo assim que a reserva seja cancelada, alterando o estado de IsReserved para false.

A interface IReservable padroniza o comportamento de reserva e cancelamento em diferentes classes no sistema de gestão de alojamentos turísticos, facilitando a manipulação de reservas e permitindo a expansão futura do sistema com novas entidades reserváveis, caso necessário.

```
namespace Tourist_Accommodation_System.Interfaces
{
    public interface IReservable
    {
        void Reserve(DateTime date); // Método que realiza uma reserva para uma data específica
        void CancelReservation(); // Método que cancela a reserva
        bool IsReserved { get; }
        DateTime ReservationDate { get; set; }
        bool IsAvailable(DateTime date); // Método que verifica disponibilidade
    }
}
```

Github

O código-fonte e a documentação completos do sistema de gestão de um alojamento turístico desenvolvido como parte deste trabalho acadêmico estão disponíveis no seguinte repositório do GitHub: https://github.com/DaniloTeixeiraCastro/TP_POO_24257