IPCA INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE

Programação Orientada a Objetos (POO)

1º Fase

Daniel Filipe Alves Vilmin

Nº28003

Índice

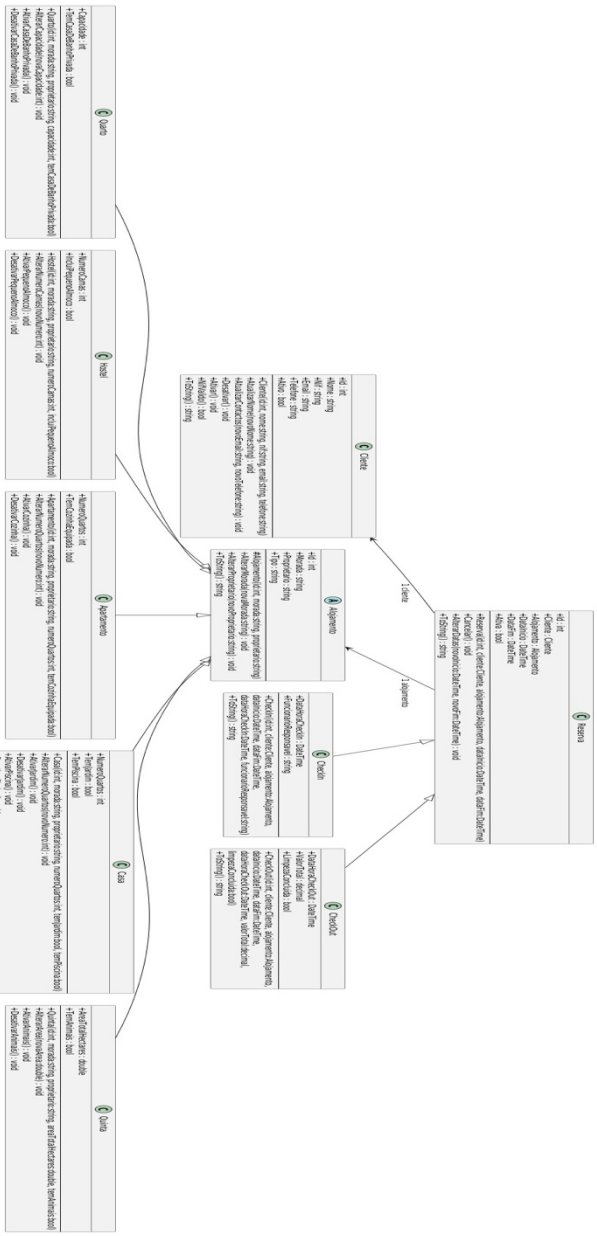
Introdução

A presente Fase 1 do trabalho de Programação Orientada a Objetos tem como objetivo desenvolver a estrutura fundamental de um sistema de gestão de alojamentos turísticos.

Este projeto procura simular um cenário realista, onde é necessário gerir clientes, alojamentos de diferentes tipologias e todo o ciclo de vida de reservas, incluindo operações de check-in e check-out.

Nesta primeira fase, são identificadas e implementadas as classes essenciais do domínio da aplicação, definindo a hierarquia de alojamentos, as entidades principais (Cliente, Reserva) e as suas relações. É também desenvolvida a arquitetura base do sistema, garantindo encapsulamento, abstração, herança e polimorfismo.

Diagrama de Estrutura de Classes



SuperClasses:

Neste projeto, conto com duas superclasses:

1. Alojamento: é a classe base que serve como modelo de todos os tipos de alojamentos, ela contêm um indentificado (ID) único, morada, proprietário e o seu respetivo tipo (quarto, casa, apartamento, etc)

2. Reserva: é a superclasse da hierarquia das reservas e dos estados delas, de check-in, e check-out.

SubClasses:

Derivadas da classe mãe Alojamento, temos as seguintes subclasses:

- **Casa:** Tem como campos próprios número de quartos, se tem jardim e se tem piscina.
  Possui métodos como AlterarNumeroQuartos, que permite modificar o número de quartos, e ainda métodos para ativar/desativar jardim e ativar/desativar piscina.
- **Apartamento:** Tem como campos próprios o número de quartos e se tem cozinha equipada.
  Inclui métodos como AlterarNumeroQuartos, que permite atualizar a quantidade de quartos, e métodos para ativar ou desativar a cozinha equipada conforme necessário.
- **Quinta:** Possui como campos específicos a área total em hectares e a indicação se tem animais.
  Disponibiliza métodos como AlterarArea, responsável por atualizar a área total da quinta, bem como métodos para AtivarAnimais e DesativarAnimais, permitindo gerir esta característica particular desta tipologia de alojamento.
- **Hostel:** Os seus campos próprios incluem o número total de camas e se inclui pequeno-almoço.
  Conta com métodos como AlterarNumeroCamas, que permite ajustar a capacidade do hostel, e métodos para ativar ou desativar a opção de pequeno-almoço incluído.
- **Quarto:** Tem como campos específicos a capacidade máxima e a indicação se tem casa de banho privada.
  Dispõe de métodos como AlterarCapacidade, que permite definir uma nova capacidade, e ainda métodos para AtivarCasaDeBanhoPrivada e DesativarCasaDeBanhoPrivada, controlando este atributo característico deste tipo de alojamento.

Derivadas da classe mãe Reserva, temos as seguintes subclasses:

- **Check-in:** Representa o momento em que o cliente dá entrada no alojamento. Possui como campos próprios a data e hora do check-in e o funcionário responsável pela operação.
  O método ToString() é sobrescrito para apresentar informação adicional específica do check-in, mantendo o comportamento polimórfico relativamente à classe base Reserva.
- **Check-Out:** Representa o momento em que o cliente abandona o alojamento. Inclui como campos próprios a data e hora do check-out, o valor total a pagar e a indicação se a limpeza foi concluída após a saída.
  No seu construtor, define automaticamente a reserva como inativa, refletindo o fim da estadia.
  Tal como no Check-In, o método ToString() é sobrescrito para acrescentar informação específica ao texto da reserva.

Por último, temos uma classe isolada no sistema, a classe **Cliente**, uma vez que não herda de nenhuma outra classe nem possui subclasses associadas.

Esta classe representa diretamente a entidade cliente do sistema, armazenando toda a informação relevante necessária para a sua identificação e gestão.

A classe contém os campos fundamentais como o Id, Nome, NIF, Email, Telefone e o estado Ativo, que indica se o cliente está disponível para realizar reservas. Para além dos atributos, a classe disponibiliza um conjunto de métodos essenciais para gerir e atualizar os dados do cliente:

- AtualizarNome() – permite alterar o nome do cliente, assegurando que o novo nome é válido.
- AtualizarContactos() – possibilita atualizar simultaneamente o email e o número de telefone, facilitando a manutenção dos contactos.
- Desativar() – muda o estado do cliente para inativo, o que funciona como uma forma de “remoção lógica”, protegendo o histórico do sistema.
- Ativar() – reativa o cliente, permitindo novamente a realização de reservas.
- NifValido() – executa uma validação simples ao NIF, verificando se contém exatamente nove dígitos numéricos.
- ToString() – devolve uma representação textual completa do cliente, útil para listagens e apresentação de dados no sistema.

Melhorias Futuras

Para melhorias futuras, seria interessante o repensamento da minha classe reserva e acrescentar outras subclasses e remover ou editar as existentes. Implementação gráfica, utilizando tencologias como o Windows forms, tornando o sistema mais intuitivo e acessível.

Acrescentar mecanismos para servir de base de dados, como Json, ou a implentaçao em si de uma base de dados em SQL.

Conclusão

A realização desta primeira fase permitiu estabelecer a base estrutural do sistema de gestão de alojamentos turísticos, definindo as principais entidades e relações que sustentam o funcionamento da aplicação.

Foram identificadas e desenvolvidas as classes centrais do domínio, bem como as subclasses especializadas, garantindo uma estrutura clara e preparada para evoluções futuras. Além disso, foi assegurada a organização lógica do sistema e a validação básica dos dados, criando alicerces sólidos para funcionalidades mais avançadas.

Embora esta fase se tenha focado essencialmente na modelação e implementação das classes fundamentais, o trabalho realizado abre caminho para futuras melhorias, como a implementação de mecanismos de persistência, a criação de uma interface gráfica e o refinamento da hierarquia de reservas. Assim, a Fase 1 representa um passo essencial na construção do sistema, preparando todas as bases necessárias para o desenvolvimento das próximas etapas.

Anexos

Git: https://github.com/DanielVilm/AlojamentoLocal/tree/main/Alojamento