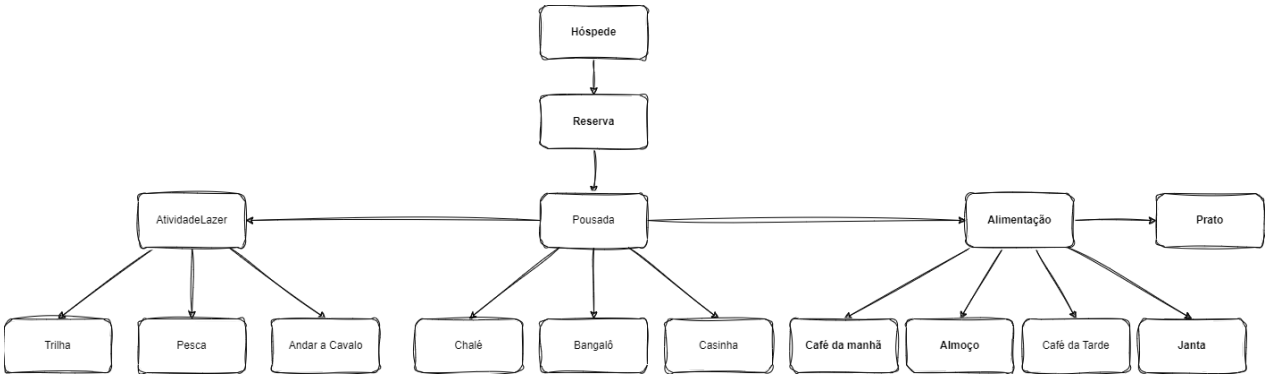
Alunos: (202020901641) - Igor Semphoski de Assis
(202020902751) - Isadora Cristina Marques Firmo

Padrões de Projeto: Singleton, Builder e Factory

Descrição do projeto:

O domínio selecionado foi o de Pousada. Imaginamos os seguintes componentes no nosso sistema:



Partimos do pressuposto de que haveria apenas uma única pousada disponível para ser reservada. Portanto, a pousada só poderia estar ocupada por um hóspede por vez. Além disso, a pousada irá oferecer serviços de alimentação para o hóspede que estiver hospedado, podendo contemplar desde o café da manhã até a janta. A alimentação é cobrada de acordo com um cardápio, que pode ser definido diariamente. Por fim, a pousada ainda contará com variadas atividades de lazer para entreter o hóspede. Cada atividade terá um tempo de duração distinto.

Singleton:

O padrão de projeto Singleton é usado quando desejamos garantir que uma classe tenha apenas uma única instância em todo o sistema. Além disso, ele fornece um ponto de acesso global para essa instância. Aqui estão algumas situações em que o padrão Singleton pode ser aplicado:

- Gerenciamento de conexões de banco de dados: quando você precisa de uma única conexão compartilhada em várias partes do código, o Singleton pode garantir que apenas uma instância da conexão seja criada e compartilhada.
- Registros de log: usar um Singleton para gerenciar registros de log permite que você acesse e grave logs de qualquer lugar da aplicação, mantendo uma única instância do logger.

Utilizamos o padrão Singleton na entidade Pousada, pois, de acordo com a nossa regra de negócio, teríamos uma única pousada registrada no sistema e gostaríamos de controlar a realização de reservas, de modo que apenas um hóspede pudesse realizar uma reserva por vez.

Builder:

O padrão de projeto Builder é usado quando queremos criar objetos complexos passo a passo. Ele separa a construção de um objeto complexo de sua representação, permitindo que o mesmo processo de construção seja usado para criar diferentes representações do objeto. Aqui estão algumas situações em que o padrão Builder pode ser aplicado:

- Criação de objetos com muitos parâmetros: quando uma classe tem um grande número de parâmetros em seu construtor, o uso do padrão Builder pode tornar a criação do objeto mais legível e fácil de entender, permitindo a configuração seletiva de cada parâmetro.
- Criação de objetos imutáveis: o Builder é útil quando você deseja criar objetos imutáveis, onde todos os campos são definidos no momento da construção. Isso pode ser útil em cenários onde a consistência e a segurança dos objetos são importantes.
- Construção de objetos complexos: se você está lidando com a criação de objetos complexos com várias etapas ou partes opcionais, o Builder pode ajudar a simplificar o processo de construção, permitindo que você defina os passos de construção de forma clara e organizada.

Desenvolvemos uma classe builder para cuidar da construção da parte de alimentação do sistema de Pousada. Fizemos esta escolha, pois, como a pousada poderá ter períodos de refeições diferentes e com cardápios variados, o processo de criação de cardápios seria alcançado de uma forma mais simples se comparado com a instanciação manual dos objetos.

Factory:

O padrão de projeto Factory é usado quando queremos criar objetos de um conjunto de subclasses relacionadas, mas queremos que a lógica de criação seja isolada do código cliente. Ele fornece um método de fábrica que encapsula a criação dos objetos, permitindo que o cliente trabalhe com a interface comum das classes. Aqui estão algumas situações em que o padrão Factory pode ser aplicado:

- Encapsulamento de lógica de criação: se a lógica de criação de objetos envolve etapas complexas ou dependências externas, a Factory pode encapsular essa lógica e fornecer uma interface simples para o cliente.
- Flexibilidade na criação de objetos: o uso do padrão Factory permite que você adicione ou altere facilmente a lógica de criação de objetos sem afetar o código cliente, tornando o sistema mais flexível e fácil de manter.

Utilizamos o padrão factory para encapsular a lógica utilizada para criar as atividades de lazer existentes no nosso sistema, para permitir que cada atividade de lazer possa ter a possibilidade de implementar a sua própria regra de negócio de forma desacoplada e independente.