Alunos: Leandro João Marqueti - 202010005789

Antony Rairon da Silva Moreira - 202010005390

Eduardo Antônio Raimundo de Souza - 202010005786

O Domínio escolhido pela equipe foi o Pousada para desenvolver a aplicação envolvendo os padrões de projetos, e os Padrões de projeto escolhidos para a aplicação foram os Singleton, Builder e Factory.

**Singleton -** É um padrão de projeto que garante que uma classe tenha apenas uma instância e forneça um jeito de acessá-la globalmente. O objetivo é limitar a criação de objetos de uma classe específica a um único objeto e disponibilizar um meio de acessá-lo de qualquer lugar.

Para implementar o Singleton, geralmente criamos uma classe com um construtor privado, impedindo a instância direta dela de fora da própria classe. Em vez disso, a classe oferece um método estático, normalmente chamado de "getInstance()", que retorna a única instância da classe.

Este método getInstance() verifica se a instância já foi criada. Se sim, retorna a instância existente e caso contrário cria uma nova instância e a retorna. Dessa forma, sempre que o método getInstance() é chamado, ele retorna a mesma instância, garantindo que exista apenas uma instância da classe durante a execução do programa.

O Singleton é usado com grande frequência quando queremos ter apenas uma instância de uma classe que controla recursos compartilhados, como conexões com bancos de dados, gerenciadores de logs ou caches. Ele também pode ser muito útil quando precisamos coordenar ações em um sistema distribuído ou limitar o número de instâncias de uma classe por questões de desempenho ou consistência de dados.

Em nosso projeto foi aplicado na classe responsável pelo gerenciamento da pousada, garantindo que haja apenas uma instância dessa classe em toda a aplicação, utilizado para notificar os hóspedes sobre a disponibilidade de quartos e atualizações nas reservas.

**Builder -** É um padrão de projeto que tem como objetivo separar a construção de um objeto complexo de sua representação, permitindo que o mesmo processo de construção possa criar diferentes representações.

O padrão Builder é bem útil quando se deseja construir objetos complexos com diferentes configurações possíveis. Ele oferece uma maneira flexível de criar objetos, isolando o código de construção do objeto final do código que utiliza o objeto construído.

A principal ideia por trás do padrão Builder é permitir a construção de objetos complexos de forma incremental, em etapas, usando um conjunto de métodos específicos para cada parte do objeto. Isso oferece um alto nível de flexibilidade, permitindo a criação de objetos com diferentes combinações de partes.

É frequentemente usado em situações em que a criação direta de um objeto usando um construtor tradicional se torna complicada ou não escalável. Permitindo a criação de objetos com uma sintaxe mais clara e modular, além de facilitar a inclusão de novas funcionalidades no processo de construção.

Em nosso projeto foi aplicado na construção de objetos complexos, como os dados de um hóspede, permitindo a criação passo a passo com diferentes opções configuráveis.

**Factory -** É um padrão de projeto que ajuda na criação de objetos. Ele oferece uma maneira de criar objetos sem ter que especificar a classe concreta exata no código.

Em vez de criar diretamente um objeto, você usa uma classe especial chamada Factory. Essa classe tem um método chamado factory method, que é responsável por criar e retornar instâncias de objetos. Você chama esse método da classe Factory para obter o objeto desejado.

A vantagem do Factory Method é que ele permite adicionar novos tipos de objetos sem precisar modificar o código existente. Você pode criar subclasses da classe Factory e implementar o factory method de acordo com suas necessidades específicas.

Com o Factory Method, você trabalha com a classe Factory por meio de uma interface comum, sem se preocupar com os detalhes de criação do objeto real. Isso torna o código mais flexível e fácil de manter.

Em nosso projeto foi utilizado na criação dos objetos de estrutura de dados, como a classe Pousada e Quarto.

Conclusão:

Em nosso projeto essa combinação de padrões garante que as classes mantenham um estado consistente e evita problemas de concorrência quando várias partes do sistema tentam acessar e modificar esses dados (Singleton), facilita a criação de objetos complexos passo a passo tornando o processo mais legível e flexível (Builder) e abstrai a criação de objetos para uma classe dedicada, permitindo criar diferentes tipos de quartos com base em critérios específicos (Factory). Cada um dos padrões são aplicados em diferentes partes do projeto para melhorar a organização, a eficiência, a robustez, a flexibilidade e a manutenção do código do sistema da pousada.