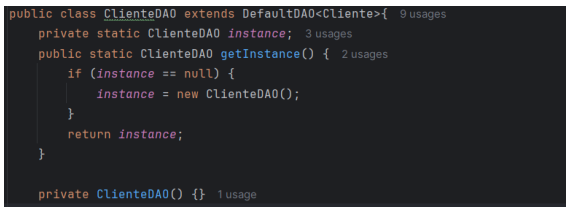
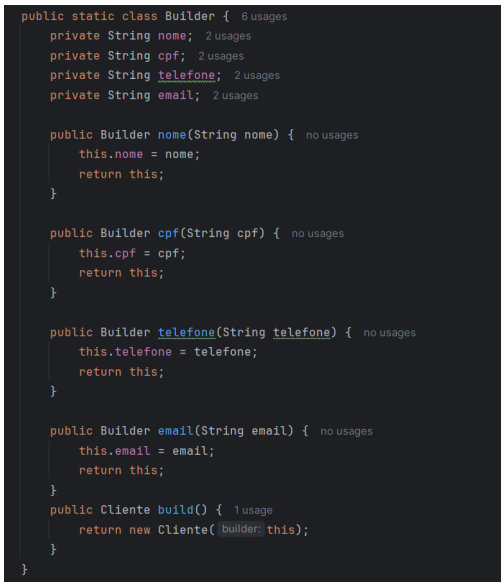
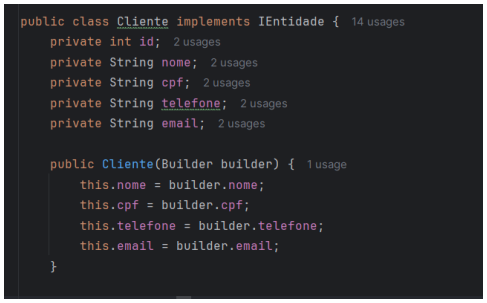
O tema implementado na aplicação foi pousada, ela possui clientes, quartos e reservas. Para a sua estruturação foi escolhido os padrões de projetos sigleton controlando a persistência dos dados, buider administrando a criação dos objetos, facade para criação de processos mais complexos e state controlando o status da reserva.

O padrão Singleton é utilizado para assegurar que um objeto tenha apenas uma instância na memória. Nesta aplicação, foi empregado nas classes responsáveis pela persistência dos dados, garantindo que o objeto não seja perdido e mantendo sua singularidade.



O padrão de projeto Builder é utilizado para facilitar a criação de objetos complexos, dividindo a inserção dos dados em vários métodos. Nesta aplicação, ele foi implementado nas classes models do sistema, simplificando a criação dos objetos e melhorando a legibilidade do código, pois fica claro para qual atributo cada dado está sendo direcionado.





O padrão Facade é utilizado para simplificar processos complexos, centralizando todas as suas etapas em uma classe facade, facilitando o uso futuro. Neste projeto, ele foi empregado para criação da reserva de um quarto. Optou-se por este padrão devido às várias etapas envolvidas nessa atividade, que necessitam a interação com diversas classes, como a verificação da disponibilidade do quarto em um período determinado.

```java
public class ReservaFacade {  2 usages
    ClienteDAO clienteDAO;  2 usages
    QuartoDAO quartoDAO;  3 usages
    ReservaDAO reservaDAO;  4 usages

    public ReservaFacade() {  1 usage
        this.clienteDAO = ClienteDAO.getInstance();
        this.quartoDAO = QuartoDAO.getInstance();
        this.reservaDAO = ReservaDAO.getInstance();
    }

    public Reserva fazerReserva(int idCliente, int idQuarto, int nrPessoas, LocalDate checkIn, LocalDat
        if(quartoDisponivel(idQuarto, checkIn, checkOut)) {
            Cliente cliente = clienteDAO.getById(idCliente);
            Quarto quarto = quartoDAO.getById(idQuarto);
            Reserva reserva =  new Reserva.Builder()
                    .cliente(cliente)
                    .quarto(quarto)
                    .numeroPessoas(nrPessoas)
                    .checkIn(checkIn)
                    .checkOut(checkOut)
                    .temporada(temporada)
                    .build();

            reservaDAO.save(reserva);
            return reserva;
        }
        else{
            throw new Exception("Qaurto não disponivel para está data");
        }
    }
```

O padrão State é utilizado para gerenciar atributos que possam ter vários estados, delegando o controle dos estados às próprias classes dos estados. Nesta aplicação, esse padrão foi empregado para controlar o status das reservas, proporcionando uma maneira clara e organizada de manejar esses diferentes estados.

```java
public abstract class Status {  5 inheritors  9 usages

    public Status(Reserva reserva) {  5 usages
        this.reserva = reserva;
    }

    protected Reserva reserva;  12 usages
    public String nome;  4 usages
    public abstract void proximo();  5 implementations  1 usage
    public abstract void cancela();  5 implementations  1 usage
}
```

```java
public class OcupandoStatus extends Status{  1 usage
    public OcupandoStatus(Reserva reserva) {  1 usage
        super(reserva);
        this.nome = "ocupado";
    }

    @Override  1 usage
    public void proximo() { reserva.setStatus(new FinalizadoStatus(reserva)); }

    @Override  1 usage
    public void cancela() {
        System.out.println("O cancelamento não pode ser feito depois da hospedagem já ter começado");
    }
}
```