Sistema Integrado de Gestão Hoteleira

Objetivo & Contexto: Sistema de Hotelaria

Este projeto apresenta um sistema de gestão hoteleira robusto e modular, focado na aplicação dos princípios de arquitetura de software e o desenvolvimento baseado em componentes.

O objetivo central deste projeto foi criar um sistema modularizado, onde cada componente fosse empacotado como um arquivo .jar.

CSU02 | Alugar Quarto

Processo de reserva e ocupação de quartos.

CSU03 | Manter Hóspede

Criação e consulta de informações de hóspedes.

CSU04 | Lançar Serviço/Produtos

Registro de serviços/produtos adicionais consumidos pelo hóspede.

CSU07 | Encerrar Estadia

Processo de checkout e liberação do quarto.

CSU08 | Avaliar Hospedagem

Feedback sobre a experiência do hóspede.

Caso de Uso | Sistema Completo

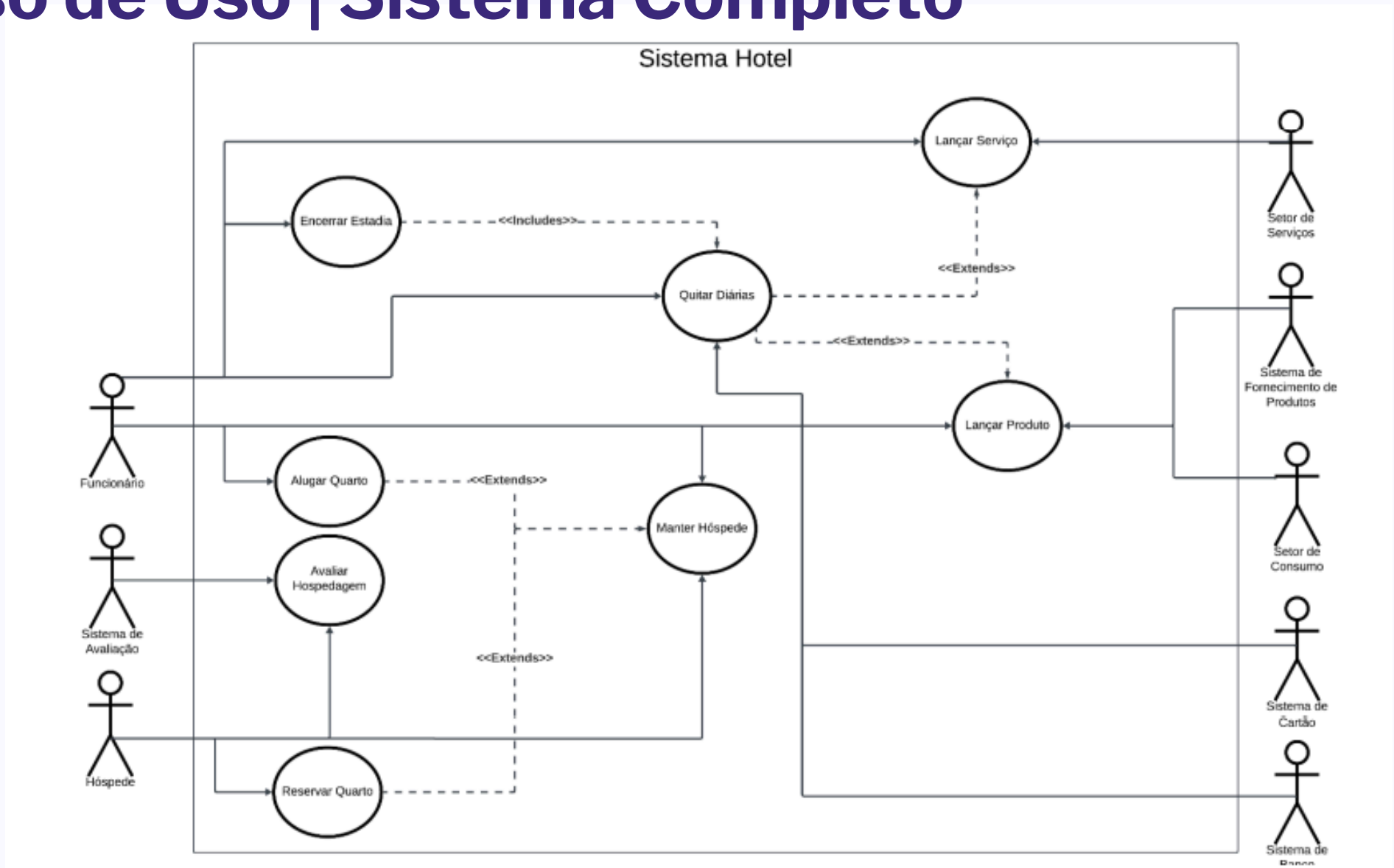
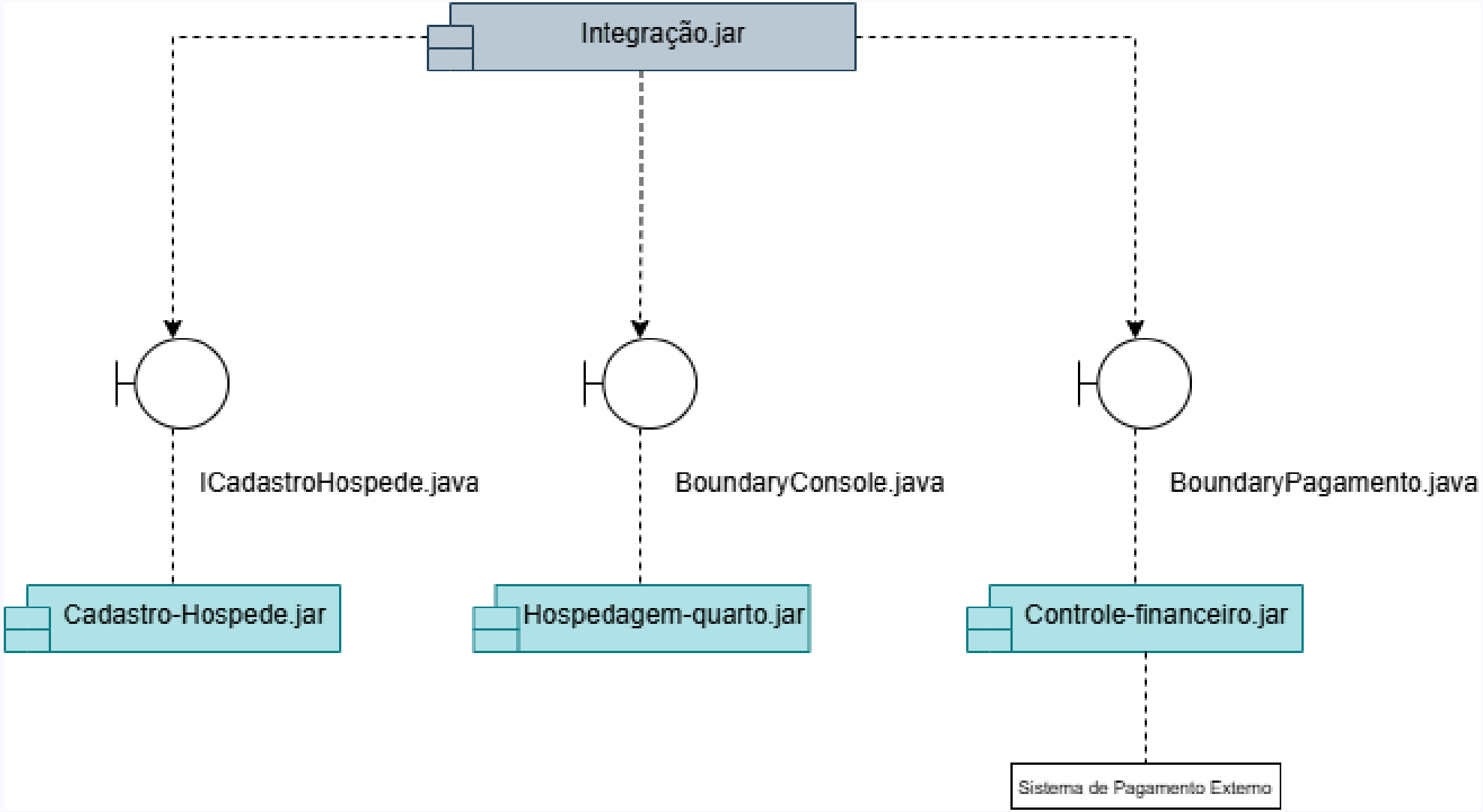
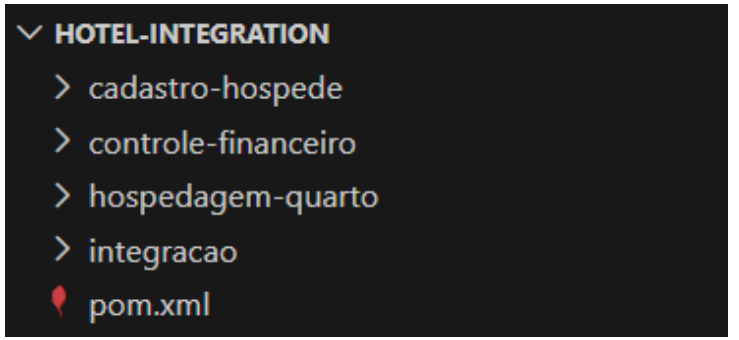


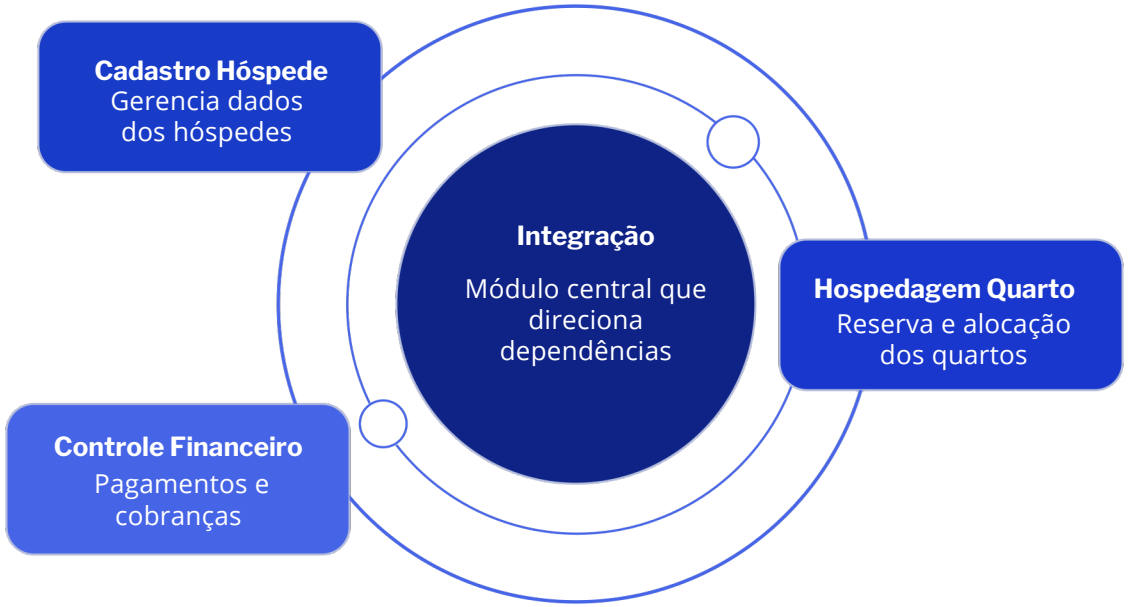
Diagrama de Componentes



Arquitetura em Componentes:

O projeto foi dividido em três componentes independentes, sendo eles:



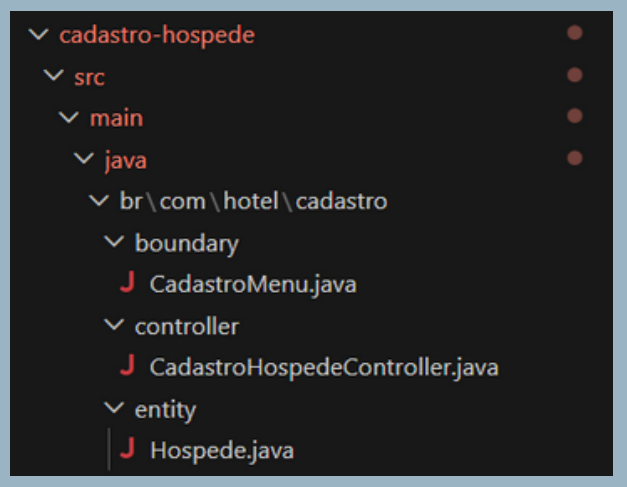


Cada componente é empacotado como .jar e publicado localmente no repositório Maven, sendo o módulo integração referencia os outros três como suas dependências.

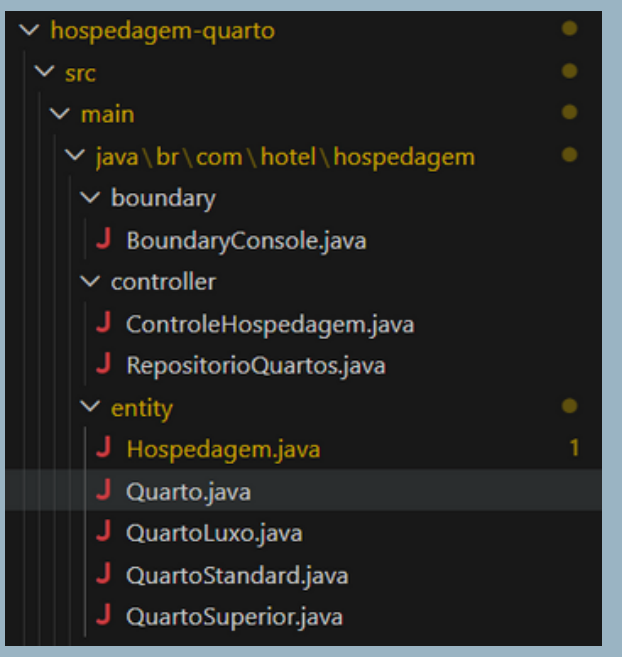
Divisão BCE / VCP

O padrão Boundary-Control-Entity (BCE) foi aplicado em cada módulo para garantir uma separação clara de responsabilidades e promover a modularidade, para assim facilitar na integração no módulo principal.

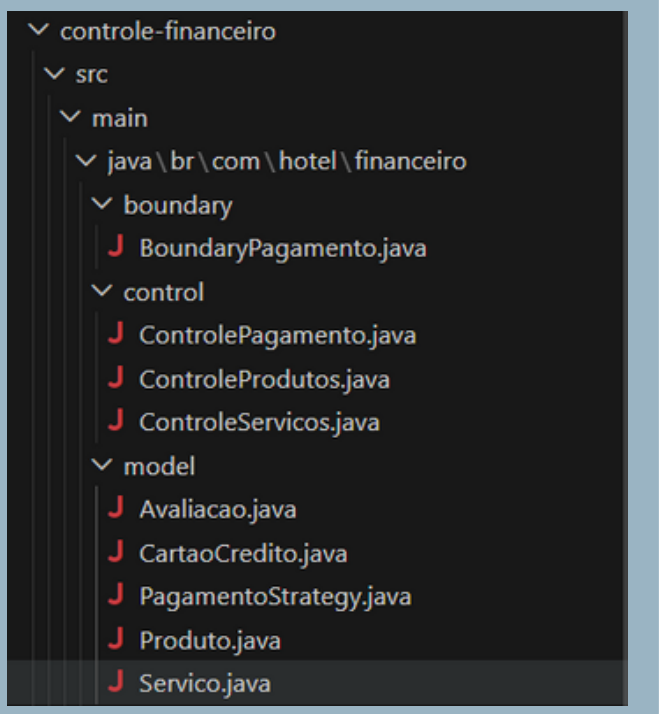
Cadastro-Hospede



Hospedagem-Quarto



Financeiro



Princípios SOLID | Cadastro-Hospede

SRP (Princípio da Responsabilidade Única):

- Validador.java é o responsável apenas pelas validações.

```java
package br.com.hotel.cadastro.utils;

public class Validador {

    public static boolean validarCPF(String cpf) {
        return cpf != null && cpf.matches(regex: "\\d{11}");
    }

    public static boolean validarEmail(String email) {
        return email != null && email.contains(s: "@");
    }
}
```

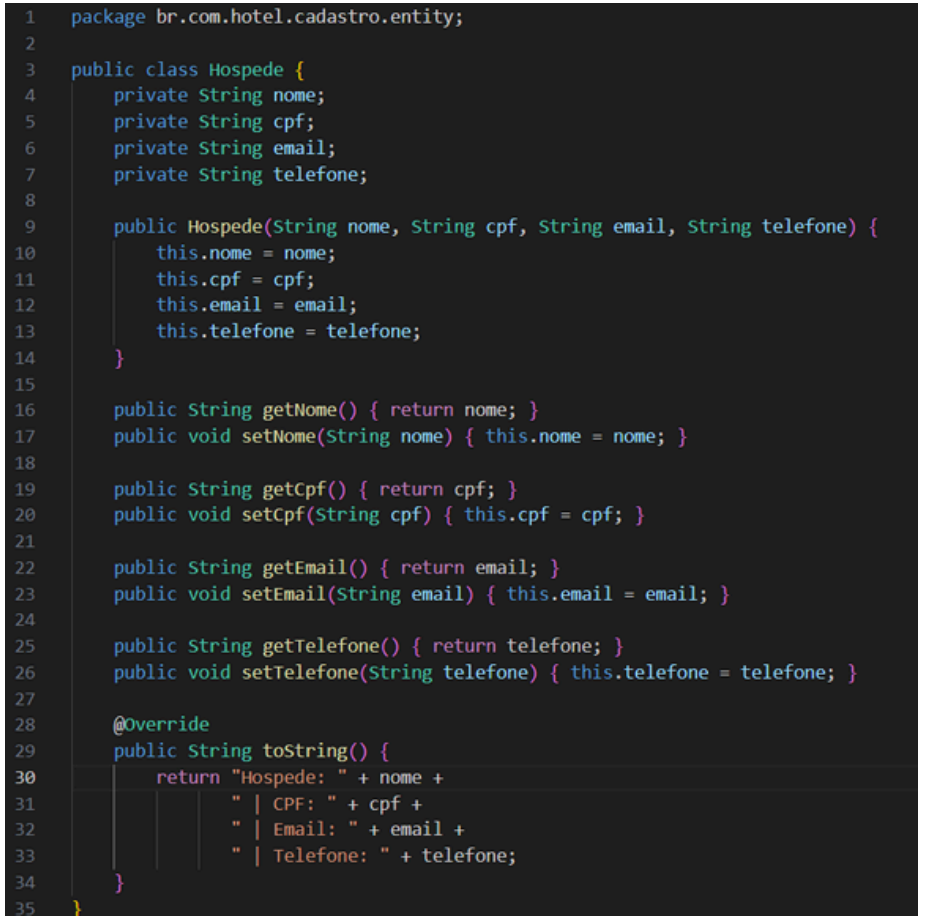
- HospedeRepository armazena hóspedes.

```java
package br.com.hotel.cadastro.repository;

import java.util.ArrayList;
import java.util.List;

import br.com.hotel.cadastro.entity.Hospede;

public class HospedeRepository {

    private List<Hospede> hospedes = new ArrayList<>();

    public Hospede salvar(Hospede hospede) {
        hospedes.add(hospede);
        return hospede;
    }

    public Hospede buscar(String cpf) {
        return hospedes.stream()
                .filter(h -> h.getCpf().equals(cpf))
                .findFirst()
                .orElse(other: null);
    }

    public List<Hospede> listar() {
        return hospedes;
    }
}
```

Princípios SOLID | Cadastro-Hospede

SRP (Princípio da Responsabilidade Única):

- Hospede.java contém apenas atributos do hóspede.

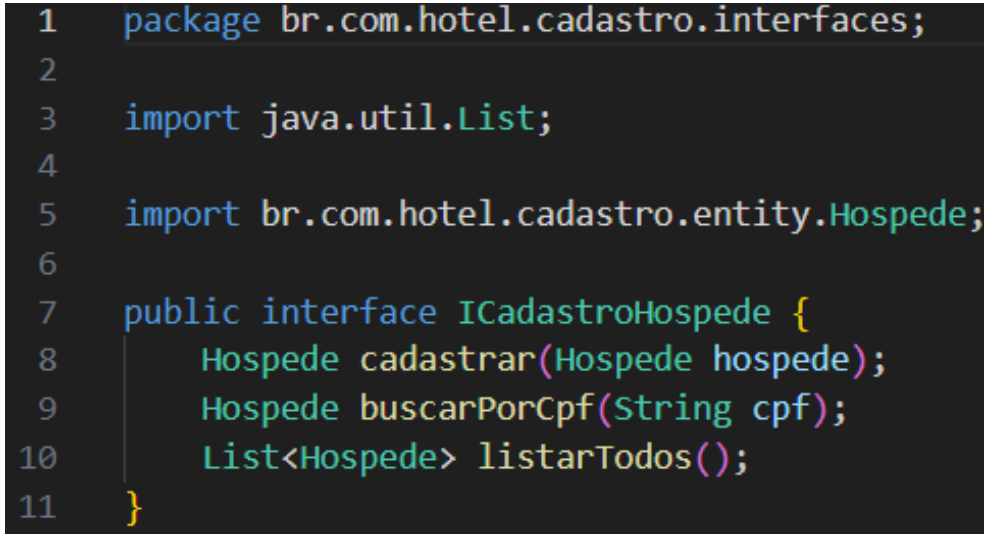


```java
package br.com.hotel.cadastro.entity;

public class Hospede {
    private String nome;
    private String cpf;
    private String email;
    private String telefone;

    public Hospede(String nome, String cpf, String email, String telefone) {
        this.nome = nome;
        this.cpf = cpf;
        this.email = email;
        this.telefone = telefone;
    }

    public String getNome() { return nome; }
    public void setNome(String nome) { this.nome = nome; }

    public String getCpf() { return cpf; }
    public void setCpf(String cpf) { this.cpf = cpf; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    public String getTelefone() { return telefone; }
    public void setTelefone(String telefone) { this.telefone = telefone; }

    @Override
    public String toString() {
        return "Hospede: " + nome +
               " | CPF: " + cpf +
               " | Email: " + email +
               " | Telefone: " + telefone;
    }
}
```

Princípios SOLID | Cadastro-Hospede

ISP (Princípio de Segregação de Interface)

- ICadastroHospede define apenas as operações necessárias (cadastrar, buscar, listar).

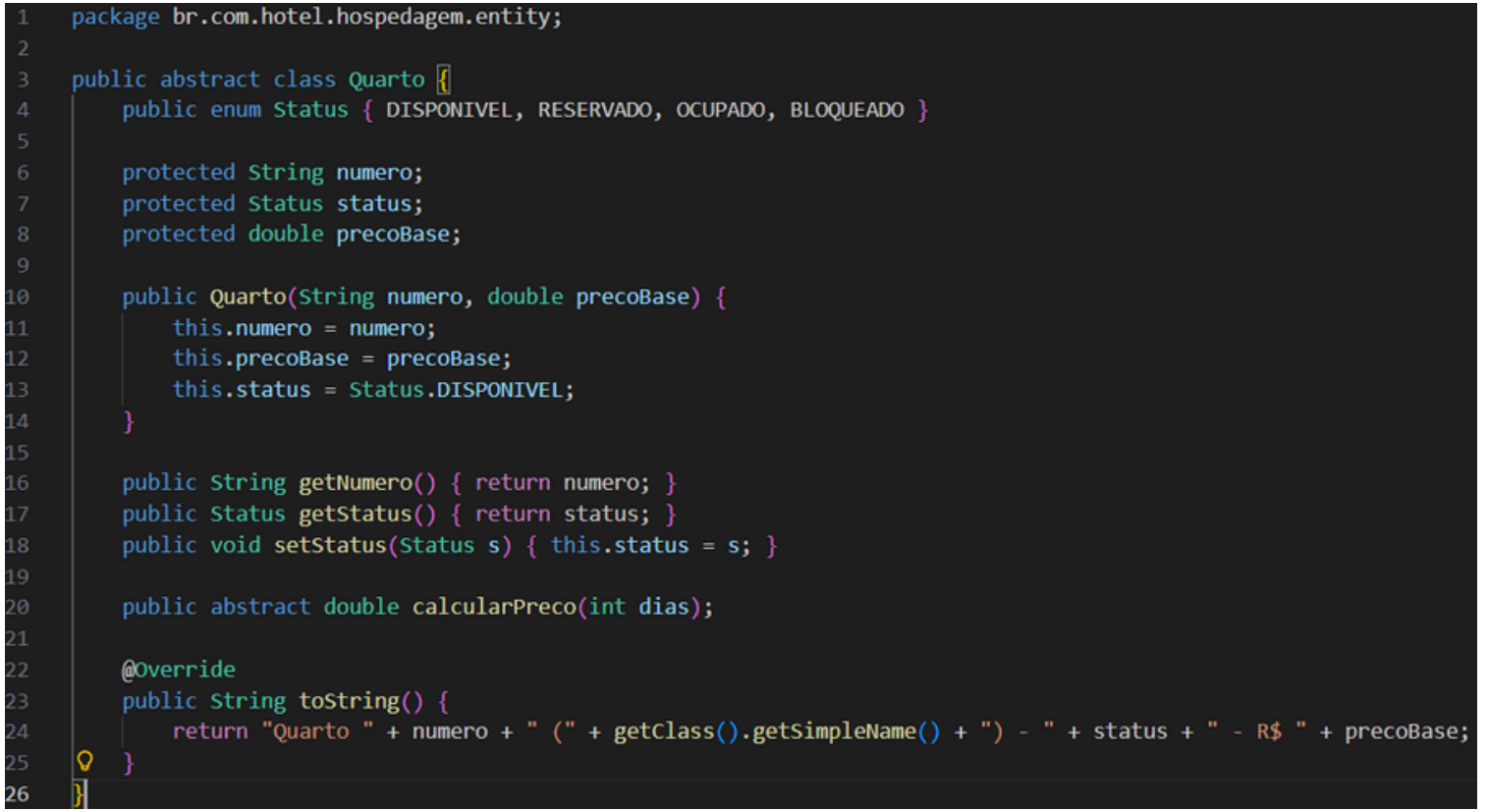


```java
package br.com.hotel.cadastro.interfaces;

import java.util.List;

import br.com.hotel.cadastro.entity.Hospede;

public interface ICadastroHospede {
    Hospede cadastrar(Hospede hospede);
    Hospede buscarPorCpf(String cpf);
    List<Hospede> listarTodos();
}
```

Princípios SOLID | Hospedagem-Quarto

OCP (Princípio do aberto/fechado):

- A hierarquia Quarto permite adicionar novos tipos de quarto sem alterar clientes que usam Quarto.



```java
package br.com.hotel.hospedagem.entity;

public abstract class Quarto {
    public enum Status { DISPONIVEL, RESERVADO, OCUPADO, BLOQUEADO }

    protected String numero;
    protected Status status;
    protected double precoBase;

    public Quarto(String numero, double precoBase) {
        this.numero = numero;
        this.precoBase = precoBase;
        this.status = Status.DISPONIVEL;
    }

    public String getNumero() { return numero; }
    public Status getStatus() { return status; }
    public void setStatus(Status s) { this.status = s; }

    public abstract double calcularPreco(int dias);

    @Override
    public String toString() {
        return "Quarto " + numero + " (" + getClass().getSimpleName() + ") - " + status + " - R$ " + precoBase;
    }
}
```

Princípios SOLID | Hospedagem-Quarto

LSP (Princípio da substituição de Liskov):

- As instâncias de QuartoLuxo, QuartoSuperior substituem Quarto sem alterar a lógica de hospedagem.

```java
package br.com.hotel.hospedagem.entity;

public class QuartoLuxo extends Quarto {
    public QuartoLuxo(String numero, double precoBase) { super(numero, precoBase); }
    @Override public double calcularPreco(int dias) { return (precoBase * 1.5) * dias; }
}
```

```java
package br.com.hotel.hospedagem.entity;

public class QuartoSuperior extends Quarto {
    public QuartoSuperior(String numero, double precoBase) { super(numero, precoBase); }
    @Override public double calcularPreco(int dias) { return (precoBase * 1.2) * dias; }
}
```

```java
package br.com.hotel.hospedagem.entity;

public class QuartoStandard extends Quarto {
    public QuartoStandard(String numero, double precoBase) { super(numero, precoBase); }
    @Override public double calcularPreco(int dias) { return precoBase * dias; }
}
```

Princípios SOLID | Controle-Financeiro

OCP (Princípio do aberto/fechado) e DIP (Princípio da Inversão de Dependência):

- *PagamentoStrategy* permite adicionar novos meios de pagamento sem alterar *ControlePagamento* ou *BoundaryPagamento.*

```java
package br.com.hotel.financeiro.model;

public interface PagamentoStrategy {

    double calcularDesconto(double valor);

    String getNomeMetodo();
}
```

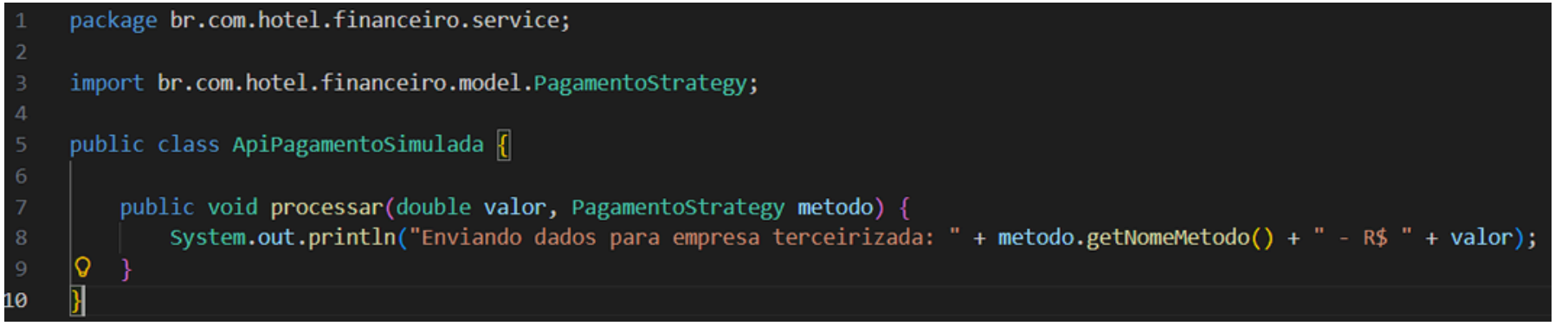
```java
package br.com.hotel.financeiro.control;

import br.com.hotel.financeiro.model.PagamentoStrategy;

public class ControlePagamento {

    public double processarPagamento(double valorTotal, PagamentoStrategy strategy) {
        return strategy.calcularDesconto(valorTotal);
    }
}
```

```java
package br.com.hotel.financeiro.boundary;

import br.com.hotel.financeiro.model.PagamentoStrategy;
import br.com.hotel.financeiro.service.ApiPagamentoSimulada;

public class BoundaryPagamento {

    private ApiPagamentoSimulada api;

    public BoundaryPagamento(ApiPagamentoSimulada api) {
        this.api = api;
    }

    public void realizarPagamento(double valor, PagamentoStrategy metodo) {
        double valorFinal = metodo.calcularDesconto(valor);
        api.processar(valorFinal, metodo);
        System.out.println("Pagamento realizado: R$ " + valorFinal);
    }
}

```

Princípios SOLID | Controle-Financeiro

SRP (Princípio da Responsabilidade Única):

- ApiPagamentoSimulada é apenas responsável por simular envio ao serviço externo.



```java
package br.com.hotel.financeiro.service;

import br.com.hotel.financeiro.model.PagamentoStrategy;

public class ApiPagamentoSimulada {

    public void processar(double valor, PagamentoStrategy metodo) {
        System.out.println("Enviando dados para empresa terceirizada: " + metodo.getNomeMetodo() + " - R$ " + valor);
    }
}
```

Coesão e Acoplamento

Coesão (REP/CRP/CCP):

- Cada componente foi projetado para executar uma função específica do domínio, resultando em **alta coesão**;

- O **princípio da Reutilização/Liberação Equivalente (REP)** é evidenciado pela possibilidade de reuso de cada .jar;

- O **Princípio do Reuso Comum (CRP)** e o **Princípio do Fechamento Comum (CCP**) garantem que classes que mudam juntas estão no mesmo componente e que classes reutilizadas juntas estão no mesmo componente.

Acoplamento (ADP/SDP/SAP):

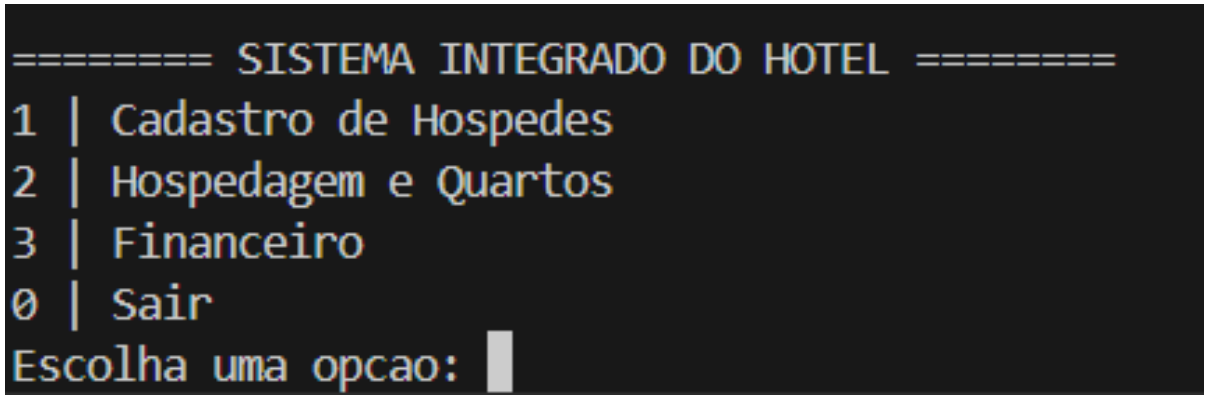
- A integração entre os módulos é feita via dependências **Maven** e chamadas diretas, mantendo um **acoplamento gerenciável**;

- O **Princípio de Equivalência Acíclica (ADP)** é mantido pelas dependências unidirecionais;

- O **Princípio das Dependências Estáveis (SDP)** e o **Princípio das Abstrações Estáveis (SAP)** são considerações para futuras melhorias.

Sistema em Execução

Sistema Integrado - Menu Principal



Sistema em Execução

Sistema Integrado - Cadastro-Hospede

```
--- MENU CADASTRO DE HOSPEDES ---
1 | Cadastrar Hospede
2 | Buscar por CPF
3 | Listar Todos
0 | Sair
Escolha uma opcao:
```

```
--- MENU CADASTRO DE HOSPEDES ---
1 | Cadastrar Hospede
2 | Buscar por CPF
3 | Listar Todos
0 | Sair
Escolha uma opcao: 1
Nome: Luiza Nanni
CPF (11 digitos): 42587596255
Email: luiza@teste.com.br
Telefone: 11985621548
Hospede cadastrado com sucesso!
```

```
--- MENU CADASTRO DE HOSPEDES ---
1 | Cadastrar Hospede
2 | Buscar por CPF
3 | Listar Todos
0 | Sair
Escolha uma opcao: 1
Nome: Angelo
CPF (11 digitos): 58725965985
Email: angelo@teste.com.br
Telefone: 1196857875
Hospede cadastrado com sucesso!
```

```
--- MENU CADASTRO DE HOSPEDES ---
1 | Cadastrar Hospede
2 | Buscar por CPF
3 | Listar Todos
0 | Sair
Escolha uma opcao: 1
Nome: Thales
CPF (11 digitos): 48725636952
Email: thales@teste.com.br
Telefone: 11987582596
Hospede cadastrado com sucesso!
```

Sistema em Execução

Sistema Integrado - Cadastro-Hospede

```
--- MENU CADASTRO DE HOSPEDES ---
1 | Cadastrar Hospede
2 | Buscar por CPF
3 | Listar Todos
0 | Sair
Escolha uma opcao: 2
Digite o CPF: 42587596255
Hospede: Luiza Nanni | CPF: 42587596255 | Email: luiza@teste.com.br | Telefone: 11985621548
```

```
--- MENU CADASTRO DE HOSPEDES ---
1 | Cadastrar Hospede
2 | Buscar por CPF
3 | Listar Todos
0 | Sair
Escolha uma opcao: 3
Hospede: Luiza Nanni | CPF: 42587596255 | Email: luiza@teste.com.br | Telefone: 11985621548
Hospede: Angelo | CPF: 58725965985 | Email: angelo@teste.com.br | Telefone: 1196857875
Hospede: Thales | CPF: 48725636952 | Email: thales@teste.com.br | Telefone: 11987582596
```

Sistema em Execução

Sistema Integrado - Hospedagem-Quarto

```
=== Hospedagem Menu ===
1 | Listar quartos disponiveis
2 | Iniciar hospedagem
3 | Encerrar hospedagem
0 | Sair
Escolha uma opcao:
```

```
=== Hospedagem Menu ===
1 | Listar quartos disponiveis
2 | Iniciar hospedagem
3 | Encerrar hospedagem
0 | Sair
Escolha uma opcao: 1
Quarto 101 (QuartoStandard) - DISPONIVEL - R$ 100.0
Quarto 102 (QuartoSuperior) - DISPONIVEL - R$ 150.0
Quarto 201 (QuartoLuxo) - DISPONIVEL - R$ 300.0
```

```
=== Hospedagem Menu ===
1 | Listar quartos disponiveis
2 | Iniciar hospedagem
3 | Encerrar hospedagem
0 | Sair
Escolha uma opcao: 2
ID hospedagem: 001
CPF hospede: 42587596255
Numero quarto: 101
Dias: 2
Hospedagem iniciada: 001
```

Sistema em Execução

Sistema Integrado - Hospedagem-Quarto

```
=== Hospedagem Menu ===
1 | Listar quartos disponiveis
2 | Iniciar hospedagem
3 | Encerrar hospedagem
0 | Sair
Escolha uma opcao: 1
Quarto 102 (QuartoSuperior) - DISPONIVEL - R$ 150.0
Quarto 201 (QuartoLuxo) - DISPONIVEL - R$ 300.0
```

```
=== Hospedagem Menu ===
1 | Listar quartos disponiveis
2 | Iniciar hospedagem
3 | Encerrar hospedagem
0 | Sair
Escolha uma opcao: 3
ID hospedagem: 001
Encerrada com sucesso
```

Sistema em Execução

Sistema Integrado - Controle-Financeiro

```
===== MENU FINANCEIRO =====
1 | Lançar serviço
2 | Lançar produto
3 | Quitar diárias
4 | Avaliar hospedagem
5 | Sair
```

```
===== MENU FINANCEIRO =====
1 | Lançar serviço
2 | Lançar produto
3 | Quitar diárias
4 | Avaliar hospedagem
5 | Sair
Escolha uma opcao: 2
Numero do quarto: 101
Nome do produto: Chocolate
Preco do produto: 10,00
Produto lancado!
```

```
===== MENU FINANCEIRO =====
1 | Lançar serviço
2 | Lançar produto
3 | Quitar diárias
4 | Avaliar hospedagem
5 | Sair
Escolha uma opcao: 1
Numero do quarto: 101
Nome do servico: Massagem
Preco do servico: 150,00
Serviço lancado!
```

Sistema em Execução

Sistema Integrado - Controle-Financeiro

```
===== MENU FINANCEIRO =====
1 | Lançar serviço
2 | Lançar produto
3 | Quitar diárias
4 | Avaliar hospedagem
5 | Sair
Escolha uma opcao: 3
Total a pagar: R$ 160.0
Enviando dados para empresa terceirizada: Cartao de Credito - R$ 152.0
Pagamento realizado: R$ 152.0
Pagamento processado!
```

```
===== MENU FINANCEIRO =====
1 | Lançar serviço
2 | Lançar produto
3 | Quitar diárias
4 | Avaliar hospedagem
5 | Sair
Escolha uma opcao: 4
Avaliação da hospedagem: 5
Avaliação enviada: 5
```

```
===== MENU FINANCEIRO =====
1 | Lançar serviço
2 | Lançar produto
3 | Quitar diárias
4 | Avaliar hospedagem
5 | Sair
Escolha uma opcao: 5
Saindo...
```

Arquivos

- **Partes Individuais:**

Cadastro-Hospede - Luiza

Hospedagem-Quarto - Angelo

Controle-Financeiro - Thales

- **Completo:**

Sistema Hoteleiro

Obrigada!

Angelo Zovaro, Luiza Nanni e Thales Miranda.