Sistema de Gerenciamento de **Reservas de Hotéis**

Instituto Federal Catarinense (IFC)
Ciência da Computação
Estruturas de Dados II
Professor: Cristhian Heck
Data: Outubro/2025
Aluno: Bernardo França Coelho
coelhobernardo199@gmail.com

1. Resumo

Este trabalho apresenta a implementação de um sistema em Java para gerenciamento de reservas de quartos em uma rede de hotéis, utilizando a estrutura de dados Árvore Rubro-Negra. O sistema permite cadastrar, consultar e cancelar reservas de forma eficiente, organizando automaticamente as reservas por data de check-in. Além disso, mantém um histórico de reservas canceladas e gera relatórios gerenciais sobre ocupação e cancelamentos. O projeto foi estruturado de forma modular, garantindo clareza, robustez e facilidade de manutenção do código.

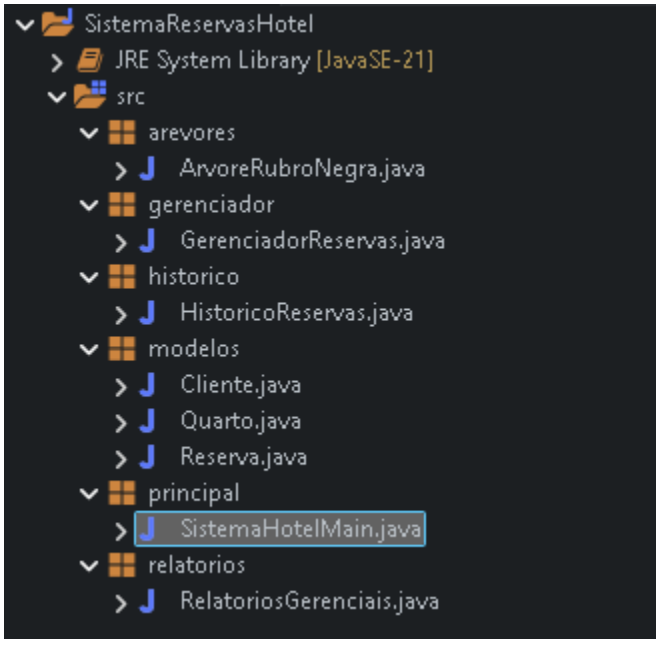
2. Introdução

Este relatório apresenta a implementação de um sistema de gerenciamento de reservas de hotéis em Java, utilizando a Árvore Rubro-Negra como principal estrutura de dados. A escolha dessa estrutura permite operações eficientes de inserção, busca e remoção, organizando automaticamente as reservas por data de check-in e garantindo desempenho mesmo com grande volume de dados.

O sistema permite cadastrar novas reservas, verificar a disponibilidade de quartos, cancelar reservas existentes e consultar informações por cliente. Além disso, mantém um histórico de reservas canceladas, possibilitando consultas futuras e a geração de relatórios gerenciais, como taxa de ocupação, número de cancelamentos e quartos mais e menos reservados.

A arquitetura foi organizada em pacotes para separar responsabilidades:

- **modelos**: classes que representam elementos do sistema, como Cliente, Quarto e Reserva.
- **arevores**: implementação genérica da Árvore Rubro-Negra utilizada para armazenar reservas de forma ordenada e balanceada.
- **gerenciador**: classes que gerenciam reservas ativas e relatórios gerenciais.
- **historico**: mantém o registro de reservas canceladas.
- **principal**: inclui a classe SistemaHotelMain, responsável por interagir com o usuário e coordenar todas as operações.



O desenvolvimento adotou uma abordagem modular, garantindo que cada parte do sistema fosse clara, reutilizável e testável. A implementação priorizou robustez, evitando conflitos de reservas, duplicidade de dados e permitindo controle eficiente de quartos e clientes.

O sistema cumpre os requisitos propostos, demonstrando a aplicação prática de estruturas de dados avançadas no gerenciamento eficiente de informações em um ambiente de hotelaria.

3. Explicação do funcionamento

O sistema funciona como um gerenciador central de reservas de hotéis, permitindo que o usuário realize operações de cadastro, consulta, cancelamento e visualização de reservas de forma eficiente. O núcleo do sistema é a Árvore Rubro-Negra, que organiza automaticamente as reservas em ordem crescente pela data de check-in. Isso garante que as operações de busca e inserção sejam realizadas de forma balanceada, evitando degradação de desempenho mesmo com grande número de reservas.

Ao cadastrar uma nova reserva, o sistema verifica se o quarto desejado está disponível nas datas especificadas, evitando conflitos. Caso o quarto esteja livre, a reserva é inserida na árvore e passa a ser considerada ativa. Em caso de cancelamento, a reserva é removida da árvore e registrada no histórico,

mantendo um registro de reservas encerradas que pode ser consultado posteriormente.

Consultas por cliente permitem localizar reservas usando o CPF ou ID do cliente, enquanto a verificação de disponibilidade de quartos permite que o usuário liste todos os quartos livres em uma determinada data e categoria. Além disso, o sistema gera relatórios gerenciais com estatísticas como taxa de ocupação, quartos mais e menos reservados e número de cancelamentos, auxiliando na gestão estratégica do hotel.

4. Desenvolvimento

O sistema foi organizado de forma modular, com pacotes que têm responsabilidades claras. No pacote **modelos**, criamos as classes principais: **Cliente** armazena o nome e o CPF, garantindo que os dados estejam corretos; **Quarto** representa os quartos do hotel com número e categoria; e **Reserva** junta cliente, quarto e datas de check-in e check-out, além de implementar **Comparable** para que a Árvore Rubro-Negra consiga organizar as reservas automaticamente por data.

No pacote **arevores**, está a implementação da Árvore Rubro-Negra genérica. Ela mantém as reservas ordenadas e balanceadas, garantindo que buscas, inserções e remoções aconteçam de forma rápida mesmo com muitas reservas. O balanceamento é feito com rotações e recoloração, e a remoção é feita de forma simplificada, mantendo a árvore funcional.

O pacote **gerenciador** centraliza a lógica do sistema. **GerenciadorReservas** gerencia as reservas ativas, cuidando do cadastro, cancelamento, consultas e verificação de conflitos de datas. **RelatoriosGerenciais** gera estatísticas como taxa de ocupação, número de cancelamentos e quartos mais ou menos reservados.

Reservas canceladas são armazenadas em **HistoricoReservas**, usando outra Árvore Rubro-Negra, permitindo consultar o histórico de forma organizada.

O **SistemaHotelMain** é a interface em console. O usuário escolhe a operação que deseja fazer, informa os dados e o sistema realiza a ação, exibindo mensagens de sucesso ou erro. Por exemplo, ao cadastrar uma reserva, o sistema verifica se o quarto está disponível; ao cancelar, atualiza o histórico.

Tudo foi pensado para que o sistema funcione de forma eficiente, clara e fácil de testar. A árvore garante rapidez mesmo com muitas reservas, e cada classe tem funções bem definidas, facilitando a manutenção e compreensão. A interface em console foi simplificada para que qualquer usuário consiga interagir com o sistema sem dificuldades.

5. Estrutura da Árvore Rubro-Negra

A **ArvoreRubroNegra** é usada para organizar as reservas por data de check-in. Cada nó guarda um valor, ponteiros para os filhos e para o pai, e uma cor (vermelho ou preto). Isso garante que a árvore fique sempre balanceada e as operações sejam rápidas.

Funcionalidades principais:

- **Inserção:** Insere um novo valor e ajusta a árvore para manter as regras da rubro-negra usando rotações e recoloração.
- **Rotações:** Rotação à esquerda e à direita reorganizam os nós sem perder a ordem, mantendo a árvore balanceada.
- **Consulta:** Permite verificar se um valor existe e listar todos os elementos em ordem crescente.
- **Remoção:** Remove um nó, substituindo por sucessor quando necessário, e ajusta a cor para manter as propriedades da árvore.
- **Relatórios internos:** Altura da árvore, contagem de nós por cor e listagem em ordem ajudam a verificar se tudo está funcionando corretamente.

O balanceamento automático da árvore após inserção e remoção exigiu cuidado para manter suas regras. A remoção de nós com dois filhos precisou localizar o sucessor e ajustar referências corretamente. Tornar a árvore genérica facilitou a reutilização em diferentes partes do sistema, como reservas ativas e histórico. A implementação foi baseada no código fornecido pelo professor e adaptada para integrar ao nosso sistema de reservas.

6. Gerenciador de Reservas

A classe **GerenciadorReservas** é responsável por administrar todas as reservas ativas do hotel, garantindo que não haja conflitos de datas e mantendo o histórico de reservas canceladas. Para isso, ela utiliza uma **Árvore Rubro-Negra** para armazenar as reservas de forma ordenada, permitindo buscas e inserções rápidas, mesmo com grande número de registros.

O gerenciamento é dividido em três funcionalidades principais:

1. **Cadastro de reservas**: Antes de inserir uma reserva, o sistema verifica se o quarto desejado já está ocupado no período informado. Caso não haja conflito, a reserva é adicionada à árvore, garantindo que todas as consultas posteriores fiquem ordenadas por data de check-in.

2. **Cancelamento de reservas**: Quando uma reserva é cancelada, ela é removida da árvore e adicionada ao **histórico de reservas**. A remoção é balanceada automaticamente pela árvore, mantendo suas propriedades rubro-negras e garantindo eficiência nas operações seguintes.

3. **Consultas e listagens**: O gerenciador permite consultar reservas por cliente, listar todas as reservas ativas e verificar quartos disponíveis para uma data e categoria específicas. As reservas são sempre acessadas em ordem, permitindo relatórios claros e rápidos.

Essa estrutura modular facilita a manutenção e testes do sistema, além de permitir que o histórico de reservas canceladas seja facilmente acessível. A implementação foi baseada no código de árvore disponibilizado pelo professor, adaptando as funcionalidades para integração completa com o sistema de reservas.

7. Histórico de Reservas

O sistema mantém um histórico de todas as reservas canceladas utilizando a **Árvore Rubro-Negra**, garantindo que as consultas continuem rápidas e que as reservas estejam sempre ordenadas por data de check-in.

A classe **HistoricoReservas** é responsável por:

- Armazenar reservas canceladas em uma árvore balanceada;
- Permitir a listagem de todas as reservas canceladas em ordem;
- Consultar se um cliente específico já teve alguma reserva cancelada;
- Retornar o total de reservas canceladas.

Quando uma reserva é cancelada no **GerenciadorReservas**, ela é automaticamente adicionada ao histórico, mantendo a integridade dos dados e possibilitando futuras análises.

8. Sistema de Gerenciamento de Reservas de Hotel

O pacote modelos contém as classes que representam os **objetos principais do sistema de hotel**: clientes, quartos e reservas.

1. Cliente

Representa um hóspede do hotel.

- **Atributos principais:**
    - **nome**: nome completo do cliente;
    - **cpf**: identificador único do cliente (11 dígitos);
- **Validação:** garante que o nome não seja vazio e que o CPF tenha 11 dígitos;
- **Sobrescritos:** equals e hashCode para que dois clientes com o mesmo CPF sejam considerados iguais;
- **Uso:** usado para identificar quem fez a reserva.

2. Quarto

Representa um quarto do hotel.

- **Atributos principais:**
    - **numero**: número do quarto;
    - **categoria**: tipo do quarto (simples, luxo, suíte etc.);
- **Uso:** cada reserva está vinculada a um quarto específico, e a categoria ajuda na busca de quartos disponíveis.

3. Reserva

Representa uma reserva de quarto feita por um cliente.

- **Atributos principais:**

    - **cliente**: referência ao cliente que fez a reserva;
    - **quarto**: referência ao quarto reservado;
    - **dataCheckIn** e **dataCheckOut**: datas da estadia;

- **Validação:** garante que as datas não sejam nulas e que o check-out não seja anterior ao check-in;
  Comparação: implementa Comparable<Reserva> usando a data de check-in (e número do quarto em caso de empate), para permitir ordenação;
  Sobrescritos: equals e hashCode para comparação de reservas; toString para exibição legível.

9. Relatórios Gerenciais

O pacote relatorios contém classes responsáveis pela **geração de informações gerenciais e estatísticas** sobre o hotel, utilizando os dados das reservas e dos quartos. A classe RelatoriosGerenciais centraliza métodos que permitem **analisar a ocupação do hotel, identificar padrões de reserva e monitorar cancelamentos**.

Funcionalidades principais

1. **Cálculo da taxa de ocupação**

    - Método: calcularTaxaOcupacao
    - Recebe todas as reservas ativas, todos os quartos e um período de datas.
    - Verifica, para cada quarto, se há alguma reserva que se sobrepõe ao período informado.
    - Retorna o percentual de quartos ocupados, permitindo medir a eficiência da ocupação.

2. **Identificação dos quartos mais e menos reservados**

    - Métodos: quartosMaisReservados e quartosMenosReservados
    - Contabilizam quantas vezes cada quarto foi reservado.
    - Retornam listas ordenadas, permitindo identificar quais quartos são mais populares e quais têm baixa demanda.

3. **Contagem de cancelamentos**

    - Método: contarCancelamentos
    - Utiliza o histórico de reservas canceladas (HistoricoReservas) e um período específico.
    - Conta quantas reservas foram canceladas no período, permitindo acompanhar perdas e ajustar estratégias.

4. **Alertas de ocupação**

    - Método: alertaCapacidade
    - Verifica se a taxa de ocupação em uma data específica ultrapassa um limite definido.
    - Retorna uma mensagem de alerta caso o limite seja atingido, ajudando na gestão de capacidade e planejamento de recursos.

Relação com o pacote modelos

- Os métodos utilizam **objetos Reserva e Quarto** do pacote modelos.
- Cada análise é baseada nas informações contidas nas reservas (datas e quartos) e nas características dos quartos disponíveis.

- Combinando RelatoriosGerenciais com HistoricoReservas e as classes de modelo, é possível **tomar decisões estratégicas**, como promoções, manutenção de quartos e previsão de demanda.

10. Controle

A classe SistemaHotelMain é a **classe principal do sistema de gerenciamento de reservas do hotel**. Ela atua como um **controlador**, coordenando a execução das operações e servindo como interface entre o usuário e os outros pacotes do sistema.

Funcionalidades principais

1. **Menu interativo**
    - Apresenta opções para cadastrar reservas, cancelar reservas, consultar reservas por cliente, listar reservas e quartos disponíveis, e gerar relatórios gerenciais.
    - Recebe e valida entradas do usuário via console, garantindo que os dados informados estejam corretos.

2. **Controle de reservas**
    - Utiliza o GerenciadorReservas para **cadastrar, consultar, cancelar e listar reservas**.
    - Integra o histórico de reservas canceladas por meio do HistoricoReservas.

3. **Gestão de quartos**
    - Mantém a lista de quartos do hotel (todosQuartos).
    - Permite consultar quartos disponíveis para uma data e categoria específicas.

4. **Geração de relatórios**
    - Conecta-se à classe RelatoriosGerenciais para calcular taxa de ocupação, listar quartos mais/menos reservados e contar cancelamentos.

- Verifica alertas de ocupação, auxiliando na tomada de decisões.

5. **Interação com o usuário**

    - Lê dados de clientes e datas de reservas, tratando erros de entrada (números inválidos ou datas em formato errado).
    - Exibe resultados de forma clara no console.

Relação com os outros pacotes

- **modelos**: Cria objetos Cliente, Quarto e Reserva a partir da entrada do usuário.
- **gerenciador**: Encaminha as reservas para cadastro, cancelamento ou consulta.
- **historico**: Acompanha reservas canceladas.
- **relatorios**: Fornece dados gerenciais para acompanhamento da ocupação e popularidade dos quartos.

11. Conclusão

O sistema de gerenciamento de reservas de hotel desenvolvido cumpriu os objetivos propostos, permitindo o controle completo de clientes, reservas e quartos, além da geração de relatórios gerenciais sobre ocupação, cancelamentos e frequência de reservas por quarto. Durante a implementação, os principais desafios foram garantir que não houvesse sobreposição de reservas em um mesmo quarto, validar corretamente as datas de check-in e check-out, assegurar a integridade dos dados dos clientes e manter uma estrutura de código organizada e modular. A documentação detalhada de cada classe e método facilitou o entendimento do funcionamento do sistema, garantindo que todas as funcionalidades fossem implementadas de forma consistente e confiável.

12. Repositório do Trabalho

Em anexo disponibilizo o código do trabalho contendo a documentação e explicação dos principais aspectos explicados durante o trabalho.

Link: https://github.com/bernardo1310/Sistemas_ReservasHotel.git