PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
UNIDADE EDUCACIONAL CORAÇÃO EUCARÍSTICO
Bacharelado em Engenharia de Software

Trabalho interdisciplinar entre Fundamentos de Engenharia de Software e AEDs I

Nome dos integrantes do grupo: Lucas Sousa Aguiar e Vinicius Oliveira Ramos

Hotel Descanso Garantido - Hotel Descanso Garantido Este projeto é um sistema simples de gerenciamento de hotel em C. Ele permite cadastrar clientes, funcionários, e estadias, bem como pesquisar informações e gerenciar estadias.

Funcionalidades Cadastrar Cliente: Adicionar novos clientes ao sistema. Cadastrar Funcionario: Adicionar novos funcionários ao sistema. Cadastrar Estadia: Registrar novas estadias para clientes, incluindo a data de entrada e saída, quantidade de hóspedes, e número do quarto. Dar Baixa em Estadia: Finalizar uma estadia, calculando o valor total a ser pago e liberando o quarto. Pesquisar Cliente: Buscar informações de clientes por código ou nome. Pesquisar Funcionario: Buscar informações de funcionários por código ou nome. Mostrar Estadias de um Cliente: Exibir todas as estadias de um cliente específico. Estrutura de Dados O sistema utiliza as seguintes estruturas de dados:

Cliente: Representa um cliente com código, nome, endereço e telefone. Funcionario: Representa um funcionário com código, nome, telefone, cargo e salário. Quarto: Representa um quarto com número, quantidade de hóspedes, valor da diária e status (ocupado ou desocupado). Estadia: Representa uma estadia com código, data de entrada, data de saída, quantidade de diárias, código do cliente e número do quarto.

Backlog do Produto

Funções Básicas

1. Cadastrar Cliente
2. Cadastrar Funcionário
3. Cadastrar Quarto
4. Cadastrar Estadia
5. Dar Baixa em Estadia
6. Pesquisar Cliente
7. Pesquisar Funcionário
8. Listar Estadias

Distribuição das Funções:

Cada membro será responsável por uma função, e cada função será desenvolvida em sprints seguindo as atividades sugeridas.

Backlog do Produto Organizado em Sprints

Sprint 1: Definição e Documentação das Funções

Responsável: Lucas Aguiar

Atividades:

1. Definir a assinatura das funções.
   - Refletir sobre os parâmetros de entrada e saída de cada função.
   - Comunicar as assinaturas aos membros do grupo.
2. Documentar as funções.
   - Indicar o propósito de cada função.
   - Descrever os parâmetros de entrada e saída.
   - Escolher nomes que reflitam o que cada função faz do ponto de vista do usuário.

Sprint 2: Implementação dos Casos de Sucesso
Responsável: Vinicius Oliveira

Atividades:

1. Implementar os casos de sucesso para cada função definida.
   - Garantir que a funcionalidade principal de cada função seja implementada corretamente.

Sprint 3: Seleção e Execução Inicial de Casos de Teste

Responsável: Lucas Aguiar

Atividades:

1. Selecionar casos de teste para verificar o funcionamento das funções.
   - Incluir valores de entrada e saídas esperadas para cada função.
2. Executar manualmente alguns casos de teste.
   - Verificar se os resultados são os esperados.

Sprint 4: Automação dos Testes e Criação de Relatório de Testes

Responsável: Vinicius Oliveira

Atividades:

1. Implementar a automatização dos testes usando a biblioteca munit.

- Criar scripts de teste automatizados para todas as funções.

2. Criar um relatório de execução de testes.

- Documentar os casos de teste, saídas retornadas e indicar se cada função passou ou não no teste (esperado x real).

Sprint 5: Implementação de Casos Especiais e Execução de Testes de Regressão

Responsável: Lucas Aguiar

Atividades:

1. Implementar casos especiais e exceções nas funções.

- Lidar com situações e entradas fora do comum.

2. Executar casos de teste anteriores para garantir a não quebra de código.

- Verificar a integridade das funções após a implementação de exceções.

3. Criar novos casos de teste para a versão atualizada das funções.

- Garantir cobertura de teste completa.

Funções Básicas do Sistema:

1. cadastrar_cliente()
2. cadastrar_funcionario()
3. cadastrar_estadia()
4. dar_baixa_estadia()
5. pesquisar_cliente()
6. pesquisar_funcionario()
7. cadastrar_quarto()
8. listar_estadias()

Cada uma dessas funções foi abordada dentro das sprints mencionadas acima.

Definição da assinatura de cada função:

1. **cadastrar_cliente()**
    - **Parâmetros de Entrada**: Não há parâmetros explícitos. Os dados do cliente são lidos do usuário através de `scanf`.
    - **Saída**: A função não retorna nenhum valor explícito. Ela adiciona um novo cliente ao array `clientes` global e atualiza `total_clientes`.

```
void cadastrar_cliente();
```

2. **cadastrar_funcionario()**
    - **Parâmetros de Entrada**: Não há parâmetros explícitos. Os dados do funcionário são lidos do usuário através de `scanf`.
    - **Saída**: A função não retorna nenhum valor explícito. Ela adiciona um novo funcionário ao array `funcionarios` global e atualiza `total_funcionarios`.

```
void cadastrar_funcionario();
```

3. **cadastrar_quarto()**
    - **Parâmetros de Entrada**: Não há parâmetros explícitos. Os dados do quarto são lidos do usuário através de `scanf`.
    - **Saída**: A função não retorna nenhum valor explícito. Ela adiciona um novo quarto ao array `quartos` global e atualiza `total_quartos`.

```
void cadastrar_quarto();
```

4. **cadastrar_estadia()**
    - **Parâmetros de Entrada**: Não há parâmetros explícitos. Os dados da estadia são lidos do usuário através de `scanf`.
    - **Saída**: A função não retorna nenhum valor explícito. Ela adiciona uma nova estadia ao array `estadias` global, atualiza `total_estadias`, e marca o quarto como "ocupado".

```
void cadastrar_estadia();
```

5. **dar_baixa_estadia()**
    - **Parâmetros de Entrada**: Não há parâmetros explícitos. O código da estadia é lido do usuário através de `scanf`.
    - **Saída**: A função não retorna nenhum valor explícito. Ela marca a estadia como encerrada, libera o quarto e calcula o valor total a ser pago.

```
void dar_baixa_estadia();
```

6. **pesquisar_cliente_por_codigo()**
    - **Parâmetros de Entrada**: Não há parâmetros explícitos. O código do cliente é lido do usuário através de `scanf`.
    - **Saída**: A função não retorna nenhum valor explícito. Ela imprime as informações do cliente correspondente ao código fornecido.

```
void pesquisar_cliente_por_codigo();
```

7. **pesquisar_cliente_por_nome()**
    - **Parâmetros de Entrada**: Não há parâmetros explícitos. O nome do cliente é lido do usuário através de `scanf`.
    - **Saída**: A função não retorna nenhum valor explícito. Ela imprime as informações de todos os clientes cujo nome corresponde ao fornecido.

```
void pesquisar_cliente_por_nome();
```

8. **pesquisar_funcionario_por_codigo()**
    - **Parâmetros de Entrada**: Não há parâmetros explícitos. O código do funcionário é lido do usuário através de `scanf`.
    - **Saída**: A função não retorna nenhum valor explícito. Ela imprime as informações do funcionário correspondente ao código fornecido.

```
void pesquisar_funcionario_por_codigo();
```

9. **pesquisar_funcionario_por_nome()**
    - **Parâmetros de Entrada**: Não há parâmetros explícitos. O nome do funcionário é lido do usuário através de `scanf`.
    - **Saída**: A função não retorna nenhum valor explícito. Ela imprime as informações de todos os funcionários cujo nome corresponde ao fornecido.

```
void pesquisar_funcionario_por_nome();
```

10. **listar_estadias()**
    - **Parâmetros de Entrada**: Não há parâmetros explícitos.
    - **Saída**: A função não retorna nenhum valor explícito. Ela imprime a lista de todas as estadias cadastradas no sistema, com detalhes como código da estadia, nome do cliente, número do quarto, datas de entrada e saída, e quantidade de diárias.

```
void listar_estadias();
```

Documentação das Funcionalidades do Software

Visão Geral

Este software gerencia um hotel fictício chamado "Hotel Descanso Garantido". As principais funcionalidades incluem o cadastro de clientes, funcionários e estadias, bem como a busca e a gestão das estadias. Os dados são persistidos em arquivos binários para garantir a continuidade das informações entre execuções do programa.

Estruturas de Dados

Cliente

```
typedef struct {
   int codigo;
   char nome[50];
   char endereco[100];
   char telefone[15];
} Cliente;
```

codigo: Identificador único do cliente.
nome: Nome do cliente.
endereco: Endereço do cliente.
telefone: Telefone de contato do cliente.
Funcionario

```
typedef struct {
   int codigo;
   char nome[50];
   char telefone[15];
   char cargo[30];
   float salario;
} Funcionario;
```

codigo: Identificador único do funcionário.
nome: Nome do funcionário.
telefone: Telefone de contato do funcionário.
cargo: Cargo do funcionário.
salario: Salário do funcionário.
Quarto

```
typedef struct {
   int numero;
   int quantidade_hospedes;
   float valor_diaria;
   char status[10];  // "ocupado" ou "desocupado"
} Quarto;
```

numero: Número do quarto.
quantidade_hospedes: Capacidade máxima de hóspedes do quarto.
valor_diaria: Valor da diária do quarto.
status: Status do quarto, podendo ser "ocupado" ou "desocupado".
Estadia

```
typedef struct {
   int codigo_estadia;
   char data_entrada[11];
   char data_saida[11];
   int quantidade_diarias;
```

```
    int codigo_cliente;
    int numero_quarto;
} Estadia;
```

codigo_estadia: Identificador único da estadia.
data_entrada: Data de entrada na estadia (formato YYYY-MM-DD).
data_saida: Data de saída da estadia (formato YYYY-MM-DD).
quantidade_diarias: Quantidade de diárias da estadia.
codigo_cliente: Identificador do cliente associado à estadia.
numero_quarto: Número do quarto associado à estadia.
Funções
Funções de Utilidade

strptime(const char* s, const char* format, struct tm* tm): Converte uma string de data para uma estrutura tm.
gerar_codigo_cliente(): Gera um código único para um cliente novo.
gerar_codigo_funcionario(): Gera um código único para um funcionário novo.
gerar_codigo_estadia(): Gera um código único para uma estadia nova.
calcular_diarias(const char* data_entrada, const char* data_saida): Calcula a quantidade de diárias entre duas datas.
Cadastro

cadastrar_cliente(): Registra um novo cliente no sistema.
cadastrar_funcionario(): Registra um novo funcionário no sistema.
cadastrar_estadia(): Registra uma nova estadia, verificando a disponibilidade de quartos.
Gestão de Estadias

dar_baixa_estadia(): Realiza o checkout de uma estadia, calculando o valor total e liberando o quarto.
Busca

pesquisar_cliente(): Permite buscar um cliente pelo código ou nome.
pesquisar_funcionario(): Permite buscar um funcionário pelo código ou nome.
mostrar_estadias_cliente(): Mostra todas as estadias associadas a um cliente específico.
Persistência de Dados

salvar_dados(): Salva os dados de clientes, funcionários, quartos e estadias em arquivos binários.
carregar_dados(): Carrega os dados de clientes, funcionários, quartos e estadias de arquivos binários.
Função Principal
menu(): Exibe o menu principal do sistema, permitindo ao usuário escolher as operações a serem realizadas.
main(): Inicia a execução do programa, chamando a função menu().
Arquitetura do Sistema

O sistema é baseado em um menu interativo que permite ao usuário acessar diferentes funcionalidades do software. As informações são armazenadas em arrays e persistidas em arquivos binários para garantir que os dados sejam mantidos entre as execuções do programa. Cada funcionalidade é implementada como uma função separada, garantindo uma arquitetura modular e fácil de manter.

Relatório de Execução de Testes - Hotel Descanso Garantido

1. Cadastrar Cliente

Caso de Teste 1.1: Cadastro de Cliente com Dados Válidos

- **Entradas:**
  - Nome: "João Silva"
  - Endereço: "Rua das Flores, 123"
  - Telefone: "11987654321"
- **Procedimento de Teste:**
  - Executar a função `cadastrar_cliente()`.
  - Inserir os dados conforme especificado.
- **Saída Esperada:**
  - Mensagem: "Cliente cadastrado com sucesso! Código: 1"
  - Cliente é adicionado ao array de clientes.
- **Resultado Obtido:** Sucesso. Cliente cadastrado corretamente com código 1.

Caso de Teste 1.2: Cadastro de Cliente com Nome Vazio

- **Entradas:**
  - Nome: ""
  - Endereço: "Rua das Flores, 123"
  - Telefone: "11987654321"
- **Procedimento de Teste:**
  - Executar a função `cadastrar_cliente()`.
  - Inserir os dados conforme especificado.

- **Saída Esperada:**
    - Mensagem de erro indicando que o nome não pode estar vazio.
    - Cliente não é adicionado ao array de clientes.
- **Resultado Obtido:** Sucesso. Sistema validou corretamente o nome vazio e não cadastrou o cliente.

2. Cadastrar Funcionário

Caso de Teste 2.1: Cadastro de Funcionário com Dados Válidos

- **Entradas:**
    - Nome: "Maria Oliveira"
    - Telefone: "11987654322"
    - Cargo: "Recepcionista"
    - Salário: 2500.00
- **Procedimento de Teste:**
    - Executar a função `cadastrar_funcionario()`.
    - Inserir os dados conforme especificado.
- **Saída Esperada:**
    - Mensagem: "Funcionário cadastrado com sucesso! Código: 1"
    - Funcionário é adicionado ao array de funcionários.
- **Resultado Obtido:** Sucesso. Funcionário cadastrado corretamente com código 1.

Caso de Teste 2.2: Cadastro de Funcionário com Salário Negativo

- **Entradas:**
    - Nome: "Carlos Santos"
    - Telefone: "11987654323"
    - Cargo: "Gerente"
    - Salário: -3000.00
- **Procedimento de Teste:**
    - Executar a função `cadastrar_funcionario()`.
    - Inserir os dados conforme especificado.
- **Saída Esperada:**
    - Mensagem de erro indicando que o salário não pode ser negativo.
    - Funcionário não é adicionado ao array de funcionários.
- **Resultado Obtido:** Sucesso. Sistema validou corretamente o salário negativo e não cadastrou o funcionário.

3. Cadastrar Estadia

Caso de Teste 3.1: Cadastro de Estadia com Dados Válidos

- **Entradas:**
  - Código do Cliente: 1
  - Quantidade de Hóspedes: 2
  - Data de Entrada: "2024-07-01"
  - Data de Saída: "2024-07-05"
- **Procedimento de Teste:**
  - Executar a função `cadastrar_estadia()`.
  - Inserir os dados conforme especificado.
- **Saída Esperada:**
  - Mensagem: "Estadia cadastrada com sucesso! Código da Estadia: 1"
  - Estadia é adicionada ao array de estadias.
  - Status do quarto é alterado para "ocupado".
- **Resultado Obtido:** Sucesso. Estadia cadastrada corretamente com código 1.

Caso de Teste 3.2: Cadastro de Estadia com Cliente Inexistente

- **Entradas:**
  - Código do Cliente: 999
  - Quantidade de Hóspedes: 2
  - Data de Entrada: "2024-07-01"
  - Data de Saída: "2024-07-05"
- **Procedimento de Teste:**
  - Executar a função `cadastrar_estadia()`.
  - Inserir os dados conforme especificado.
- **Saída Esperada:**
  - Mensagem: "Cliente não encontrado!"
  - Estadia não é adicionada ao array de estadias.
- **Resultado Obtido:** Sucesso. Sistema indicou corretamente que o cliente não foi encontrado e não cadastrou a estadia.

4. Dar Baixa em Estadia

Caso de Teste 4.1: Baixa de Estadia com Código Válido

- **Entradas:**
  - Código da Estadia: 1
- **Procedimento de Teste:**

    - Executar a função `dar_baixa_estadia()`.
    - Inserir o código da estadia conforme especificado.
- **Saída Esperada:**
    - Mensagem: "Valor total a ser pago: [valor calculado]"
    - Estadia é removida do array de estadias.
    - Status do quarto é alterado para "desocupado".
- **Resultado Obtido:** Sucesso. Baixa da estadia realizada corretamente.

Caso de Teste 4.2: Baixa de Estadia com Código Inexistente

- **Entradas:**
    - Código da Estadia: 999
- **Procedimento de Teste:**
    - Executar a função `dar_baixa_estadia()`.
    - Inserir o código da estadia conforme especificado.
- **Saída Esperada:**
    - Mensagem: "Estadia não encontrada!"
    - Nenhuma alteração no array de estadias.
- **Resultado Obtido:** Sucesso. Sistema indicou corretamente que a estadia não foi encontrada.

5. Pesquisar Cliente

Caso de Teste 5.1: Pesquisa de Cliente por Código Válido

- **Entradas:**
    - Código do Cliente: 1
- **Procedimento de Teste:**
    - Executar a função `pesquisar_cliente()`.
    - Inserir o código do cliente conforme especificado.
- **Saída Esperada:**
    - Mensagem: "Cliente encontrado: [detalhes do cliente]"
- **Resultado Obtido:** Sucesso. Cliente encontrado corretamente com o código fornecido.

Caso de Teste 5.2: Pesquisa de Cliente por Nome Válido

- **Entradas:**
    - Nome do Cliente: "João Silva"
- **Procedimento de Teste:**
    - Executar a função `pesquisar_cliente()`.
    - Inserir o nome do cliente conforme especificado.

- **Saída Esperada:**
  - Mensagem: "Cliente encontrado: [detalhes do cliente]"
- **Resultado Obtido:** Sucesso. Cliente encontrado corretamente com o nome fornecido.

Caso de Teste 5.3: Pesquisa de Cliente por Código Inexistente

- **Entradas:**
  - Código do Cliente: 999
- **Procedimento de Teste:**
  - Executar a função `pesquisar_cliente()`.
  - Inserir o código do cliente conforme especificado.
- **Saída Esperada:**
  - Mensagem: "Cliente não encontrado!"
- **Resultado Obtido:** Sucesso. Sistema indicou corretamente que o cliente não foi encontrado.

Caso de Teste 5.4: Pesquisa de Cliente por Nome Inexistente

- **Entradas:**
  - Nome do Cliente: "Pedro Silva"
- **Procedimento de Teste:**
  - Executar a função `pesquisar_cliente()`.
  - Inserir o nome do cliente conforme especificado.
- **Saída Esperada:**
  - Mensagem: "Cliente não encontrado!"
- **Resultado Obtido:** Sucesso. Sistema indicou corretamente que o cliente não foi encontrado.

6. Pesquisar Funcionário

Caso de Teste 6.1: Pesquisa de Funcionário por Código Válido

- **Entradas:**
  - Código do Funcionário: 1
- **Procedimento de Teste:**
  - Executar a função `pesquisar_funcionario()`.
  - Inserir o código do funcionário conforme especificado.
- **Saída Esperada:**
  - Mensagem: "Funcionário encontrado: [detalhes do funcionário]"
- **Resultado Obtido:** Sucesso. Funcionário encontrado corretamente com o código fornecido.

Caso de Teste 6.2: Pesquisa de Funcionário por Nome Válido

- **Entradas:**
  - Nome do Funcionário: "Maria Oliveira"
- **Procedimento de Teste:**
  - Executar a função `pesquisar_funcionario()`.
  - Inserir o nome do funcionário conforme especificado.
- **Saída Esperada:**
  - Mensagem: "Funcionário encontrado: [detalhes do funcionário]"
- **Resultado Obtido:** Sucesso. Funcionário encontrado corretamente com o nome fornecido.

Caso de Teste 6.3: Pesquisa de Funcionário por Código Inexistente

- **Entradas:**
  - Código do Funcionário: 999
- **Procedimento de Teste:**
  - Executar a função `pesquisar_funcionario()`.
  - Inserir o código do funcionário conforme especificado.
- **Saída Esperada:**
  - Mensagem: "Funcionário não encontrado!"
- **Resultado Obtido:** Sucesso. Sistema indicou corretamente que o funcionário não foi encontrado.

Caso de Teste 6.4: Pesquisa de Funcionário por Nome Inexistente

- **Entradas:**
  - Nome do Funcionário: "Carlos Santos"
- **Procedimento de Teste:**
  - Executar a função `pesquisar_funcionario()`.
  - Inserir o nome do funcionário conforme especificado.
- **Saída Esperada:**
  - Mensagem: "Funcionário não encontrado!"
- **Resultado Obtido:** Sucesso. Sistema indicou corretamente que o funcionário não foi encontrado.

7. Mostrar Estadias de um Cliente

Caso de Teste 7.1: Mostrar Estadias de Cliente com Estadias

- **Entradas:**
  - Código do Cliente: 1

- **Procedimento de Teste:**
  - Executar a função `mostrar_estadias_cliente()`.
  - Inserir o código do cliente conforme especificado.
- **Saída Esperada:**
  - Lista de estadias associadas ao cliente com detalhes (datas de entrada e saída, número do quarto, etc.).
- **Resultado Obtido:** Sucesso. Estadias do cliente foram mostradas corretamente.

Caso de Teste 7.2: Mostrar Estadias de Cliente sem Estadias

- **Entradas:**
  - Código do Cliente: 2
- **Procedimento de Teste:**
  - Executar a função `mostrar_estadias_cliente()`.
  - Inserir o código do cliente conforme especificado.
- **Saída Esperada:**
  - Mensagem: "Nenhuma estadia encontrada para este cliente."
- **Resultado Obtido:** Sucesso. Sistema indicou corretamente que não há estadias para o cliente especificado.

Conclusão:

Todos os casos de teste foram executados com sucesso, abrangendo diferentes funcionalidades do software "Hotel Descanso Garantido". O sistema demonstrou comportamento adequado ao validar entradas, processar dados e fornecer saídas esperadas conforme especificado nos testes. Este relatório pode ser usado para verificar a funcionalidade e a robustez do sistema durante o desenvolvimento e manutenção.