

# Pontifícia Universidade Católica de Minas Gerais

## Instituto de Ciências Exatas e Informática

### Unidade Educacional Coração Eucarístico

#### Bacharelado em Engenharia de Software

Nome Do Integrante: Miguel Moreira Chaves Maciel

Matrícula: 853443

Nome do Sistema: Sistema de Gerenciamento de Hotel - Hotel Descanso Garantido















## Apresentação:

O objetivo deste sistema é gerenciar clientes, funcionários, estadias e quartos do Hotel Descanso Garantido, localizado em Itacaré, BA. O sistema permite cadastrar clientes e funcionários, registrar estadias, finalizar estadias, e realizar pesquisas de clientes e funcionários, garantindo uma melhor organização e eficiência nas operações do hotel.

## BackLog do Produto:

O backlog do produto foi organizado utilizando o software ClickUp. Cada função do sistema foi atribuída a um membro do grupo e será desenvolvida em sprints de 7 a 10 dias.

## Exemplo em Lista:

Nome	Responsável	Data de vencimento	Prioridade	
<input type="radio"/> Cadastrar Cliente		jul 10	 Urgente	...
<input type="radio"/> Cadastrar Funcionario		jul 12	 Urgente	...
<input type="radio"/> Cadastrar Estadia		ago 1	 Alta	...
<input checked="" type="radio"/> Baixar Estadia		ago 5	 Normal	...
<input type="radio"/> Pesquisar Cliente		ago 14	 Baixa	...
<input type="radio"/> Pesquisar Funcionario		ago 21	 Baixa	...
<input type="radio"/> Mostrar Estadias Cliente		set 1	 Baixa	...
+ Adicionar Tarefa				

A Figura 1 apresenta a lista de tarefas do backlog do produto organizada por membro do grupo. Cada tarefa inclui a descrição da função a ser desenvolvida, o responsável pela execução e a prioridade. Esta visualização permite acompanhar o progresso individual de cada membro e garantir que todas as demandas estão sendo atendidas conforme planejado.

### Exemplo em Calendário:

A Figura 2 mostra o calendário de atividades que atualiza automaticamente durante o dia e notifica os membros sobre as atividades programadas. Esta visualização ajuda a manter todos os membros informados sobre prazos e entregas diárias, promovendo uma melhor organização e cumprimento das tarefas.

### Exemplo visualização da Função:

Tarefa86a408ydy

Cadastrar Cliente

⌕ Status

✓ Marcar Como Concluída

📅 Data final

jul 10

🕒 Tempo rastreado

➕ Adicionar hora

🔗 Relacionamentos

Vazio

👤 Responsáveis

📅 Prioridade

🏷️ Etiquetas

👤

🔴 Urgente

Vazio

Permite cadastrar novos clientes no sistema

Detalhes

Subtarefas

Itens de ação

Subtarefas

+

Nova tarefa

Criada em jul 1

Compartilhar

Atividade

> Mostrar mais

• Miguel Moreira Chaves Macie (você) changed priority from 🔴 Normal to 🔴 Urgent

3 minutos

Escreva um comentário...

Enviar

A Figura 3 apresenta o detalhamento das funções do sistema, incluindo a descrição de cada função, o responsável pela sua implementação e a prioridade atribuída. Esta visualização permite um acompanhamento detalhado das responsabilidades de cada membro e das funções que estão sendo desenvolvidas, facilitando a gestão do projeto.

## Planejamento das Sprints:

### Sprint 1:

- **Tarefas:**

- Definir assinatura das funções **Cadastrar Cliente** e **Cadastrar Funcionario**.
- Documentar as funções.
- Implementar o caso de sucesso das funções.
- Selecionar casos de teste para verificar o funcionamento das funções.
- Executar os casos de teste manualmente.
- Implementar os testes automatizados usando a biblioteca munit.
- Criar relatório de execução dos testes.

- **Responsáveis:**

- Membro 1: Miguel Moreira
- Membro 2: Miguel Moreira

### Sprint 2:

- **Tarefas:**

- Definir assinatura das funções **cadastrar Estadia** e **baixar Estadia**.
- Documentar as funções.
- Implementar o caso de sucesso das funções.
- Selecionar casos de teste para verificar o funcionamento das funções.
- Executar os casos de teste manualmente.
- Implementar os testes automatizados usando a biblioteca munit.

- Criar relatório de execução dos testes.
- **Responsáveis:**
  - Membro 1: Miguel Moreira
  - Membro 2: Miguel Moreira

### **Sprint 3:**

- **Tarefas:**
  - Definir assinatura das funções `pesquisar Cliente` e `pesquisar Funcionario`.
  - Documentar as funções.
  - Implementar o caso de sucesso das funções.
  - Selecionar casos de teste para verificar o funcionamento das funções.
  - Executar os casos de teste manualmente.
  - Implementar os testes automatizados usando a biblioteca munit.
  - Criar relatório de execução dos testes.
- **Responsáveis:**
  - Membro 1: Miguel Moreira
  - Membro 2: Miguel Moreira

### **Sprint 4:**

- **Tarefas:**
  - Definir assinatura da função `mostrar EstadiasCliente`.
  - Documentar a função.
  - Implementar o caso de sucesso da função.
  - Selecionar casos de teste para verificar o funcionamento da função.
  - Executar os casos de teste manualmente.
  - Implementar os testes automatizados usando a biblioteca munit.
  - Criar relatório de execução dos testes.
- **Responsável:**
  - Membro 1: Miguel Moreira

# Lista de Assinaturas das Funções e Parâmetros

## Funções e Parâmetros

### 1. `void cadastrarCliente(Cliente *clientes, int *numClientes)`

- **Descrição:** Permite cadastrar novos clientes no sistema.
- **Parâmetros:**

- `Cliente *clientes`: Array de clientes.
- `int *numClientes`: Ponteiro para o número de clientes cadastrados.

#### **Exemplo de Uso:**

```
Cliente clientes[100];
```

```
int numClientes = 0;
```

```
cadastrarCliente(clientes,  
&numClientes);
```

○

### 2. `void cadastrarFuncionario(Funcionario *funcionarios, int *numFuncionarios)`

- **Descrição:** Permite cadastrar novos funcionários no sistema.
- **Parâmetros:**

- `Funcionario *funcionarios`: Array de funcionários.
- `int *numFuncionarios`: Ponteiro para o número de funcionários cadastrados.

#### **Exemplo de Uso:**

```
Funcionario funcionarios[100];
```

```
int numFuncionarios = 0;
```

```
cadastrarFuncionario(funcionarios,  
&numFuncionarios);
```

3. `void cadastrarEstadia(Estadia *estadias, int *numEstadias, Cliente *clientes, int numClientes, Quarto *quartos, int numQuartos)`

- **Descrição:** Permite cadastrar novas estadias no sistema.
- **Parâmetros:**
  - `Estadia *estadias`: Array de estadias.
  - `int *numEstadias`: Ponteiro para o número de estadias cadastradas.
  - `Cliente *clientes`: Array de clientes cadastrados.
  - `int numClientes`: Número de clientes cadastrados.
  - `Quarto *quartos`: Array de quartos disponíveis.
  - `int numQuartos`: Número de quartos cadastrados.

**Exemplo de Uso:**

```
Estadia estadias[100];

int numEstadias = 0;

Cliente clientes[100];

int numClientes = 5;

Quarto quartos[100] = { {101, 2, 150.0,
"desocupado"}, {102, 2, 150.0,
"desocupado"} };

int numQuartos = 2;

cadastrarEstadia(estadias, &numEstadias,
clientes, numClientes, quartos,
numQuartos);
```

4. `void baixarEstadia(Estadia *estadias, int *numEstadias, Quarto *quartos, int numQuartos)`

- **Descrição:** Permite finalizar uma estadia, calcular o valor total e liberar o quarto.
- **Parâmetros:**
  - `Estadia *estadias`: Array de estadias.
  - `int *numEstadias`: Ponteiro para o número de estadias cadastradas.
  - `Quarto *quartos`: Array de quartos disponíveis.
  - `int numQuartos`: Número de quartos cadastrados.

**Exemplo de Uso:**

```
Estadia estadias[100];

int numEstadias = 5;

Quarto quartos[100] = { {101, 2, 150.0,
"ocupado"}, {102, 2, 150.0, "ocupado"}
};

int numQuartos = 2;

baixarEstadia(estadias, &numEstadias,
quartos, numQuartos);
```

○

5. `void pesquisarCliente(Cliente *clientes, int numClientes)`

- **Descrição:** Permite buscar e exibir informações de um cliente pelo código.
- **Parâmetros:**
  - `Cliente *clientes`: Array de clientes cadastrados.
  - `int numClientes`: Número de clientes cadastrados.

**Exemplo de Uso:**

```
Cliente clientes[100];

int numClientes = 5;

pesquisarCliente(clientes, numClientes);
```

## 6. `void pesquisarFuncionario(Funcionario *funcionarios, int numFuncionarios)`

- **Descrição:** Permite buscar e exibir informações de um funcionário pelo código.
- **Parâmetros:**
  - `Funcionario *funcionarios`: Array de funcionários cadastrados.
  - `int numFuncionarios`: Número de funcionários cadastrados.

### Exemplo de Uso:

```
Funcionario funcionarios[100];
```

```
int numFuncionarios = 5;
```

```
pesquisarFuncionario(funcionarios,  
numFuncionarios);
```

## Casos de Teste

### 1. cadastrar Cliente

Entradas	Classes Válidas	Resultado Esperado	Classes Inválidas	Resultado Esperado
Nome: "Joao"	Nome: String não vazia	Cliente cadastrado com sucesso	Nome: String vazia, números, caracteres especiais	Cliente não cadastrado, mensagem de erro exibida
Endereço: "Rua A"	Endereço: String não vazia	Cliente cadastrado com sucesso	Endereço: String vazia, números, caracteres especiais	Cliente não cadastrado, mensagem de erro exibida



Telefone: "123456789"	Telefone: String com 9 a 15 caracteres numéricos	Cliente cadastrado com sucesso	Telefone: String com menos de 9 caracteres ou mais de 15 caracteres, letras, caracteres especiais	Cliente não cadastrado, mensagem de erro exibida
--------------------------	---	--------------------------------------	---	---

## 2. cadastrar Funcionario

Entradas	Classes Válidas	Resultado Esperado	Classes Inválidas	Resultado Esperado
Nome: "Maria"	Nome: String não vazia	Funcionário cadastrado com sucesso	Nome: String vazia, números, caracteres especiais	Funcionário não cadastrado, mensagem de erro exibida

Telefone: "987654321"	Telefone: String com 9 a 15 caracteres numéricos	Funcionário cadastrado com sucesso	Telefone: String com menos de 9 caracteres ou mais de 15 caracteres, letras, caracteres especiais	Funcionário não cadastrado, mensagem de erro exibida
--------------------------	---	--	---	---

Cargo: "Recepcionista"	Cargo: String não vazia	Funcionário cadastrado com sucesso	Cargo: String vazia, números, caracteres especiais	Funcionário não cadastrado, mensagem de erro exibida
---------------------------	-------------------------------	--	--	--

Salário: 1500.00	Salário: Número positivo	Funcionário cadastrado com sucesso	Salário: Número negativo, zero	Funcionário não cadastrado, mensagem de erro exibida
---------------------	--------------------------------	--	-----------------------------------	--

## 3. cadastrar Estadia

Entradas	Classes Válidas	Resultado Esperado	Classes Inválidas	Resultado Esperado
----------	-----------------	--------------------	-------------------	--------------------

Código do Cliente: 1	Código do Cliente existente	Estadia cadastrada com sucesso	Código do Cliente inexistente	Estadia não cadastrada, mensagem de erro exibida
Quantidade de Hóspedes: 2	Quantidade de Hóspedes <= capacidade do quarto	Estadia cadastrada com sucesso	Quantidade de Hóspedes > capacidade do quarto	Estadia não cadastrada, mensagem de erro exibida
Data de Entrada: "01/01/2024"	Data de Entrada: Data válida no formato "dd/mm/aaaa"	Estadia cadastrada com sucesso	Data de Entrada: Data inválida, formato diferente, data passada	Estadia não cadastrada, mensagem de erro exibida
Data de Saída: "05/01/2024"	Data de Saída: Data válida no formato "dd/mm/aaaa", posterior à Data de Entrada	Estadia cadastrada com sucesso	Data de Saída: Data inválida, formato diferente, data anterior à Data de Entrada	Estadia não cadastrada, mensagem de erro exibida

## 4. baixar Estadia

Entradas	Classes Válidas	Resultado Esperado	Classes Inválidas	Resultado Esperado
Código da Estadia: 1	Código da Estadia existente	Estadia baixada com sucesso, valor total exibido, status alterado	Código da Estadia inexistente	Estadia não baixada, mensagem de erro exibida

## 5. pesquisar Cliente

Entradas	Classes Válidas	Resultado Esperado	Classes Inválidas	Resultado Esperado
Código do Cliente: 1	Código do Cliente existente	Informações detalhadas do cliente exibidas	Código do Cliente inexistente	Cliente não encontrado, mensagem de erro exibida

---

## 6. pesquisar Funcionario

Entradas	Classes Válidas	Resultado Esperado	Classes Inválidas	Resultado Esperado
Código do Funcionário: 1	Código do Funcionário existente	Informações detalhadas do funcionário exibidas	Código do Funcionário inexistente	Funcionário não encontrado, mensagem de erro exibida

---

## 7. mostrar Estadias Cliente

Entradas	Classes Válidas	Resultado Esperado	Classes Inválidas	Resultado Esperado
Código do Cliente: 1	Código do Cliente existente	Lista de estadias do cliente exibida	Código do Cliente inexistente	Nenhuma estadia encontrada, mensagem de erro exibida

---

Entradas	Saída Retornada	Resultado
Nome: "Joao"	Cliente cadastrado com sucesso	Passou
Endereço: "Rua A"	Cliente cadastrado com sucesso	Passou
Telefone: "123456789"	Cliente cadastrado com sucesso	Passou
Nome: ""	Cliente não cadastrado, mensagem de erro exibida	Passou
Endereço: ""	Cliente não cadastrado, mensagem de erro exibida	Passou
Telefone: "12"	Cliente não cadastrado, mensagem de erro exibida	Passou