**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA**

**UNIDADE EDUCACIONAL CORAÇÃO EUCARÍSTICO**

**Bacharelado em Engenharia de Software**

**Gustavo Azi Prehl Gama  
Guilherme Henrique Moreira Costa  
Hotel - Descanso Garantido  
Apresentação:**

Descreva em um parágrafo curto o objetivo do seu sistema **Backlog to produto:**

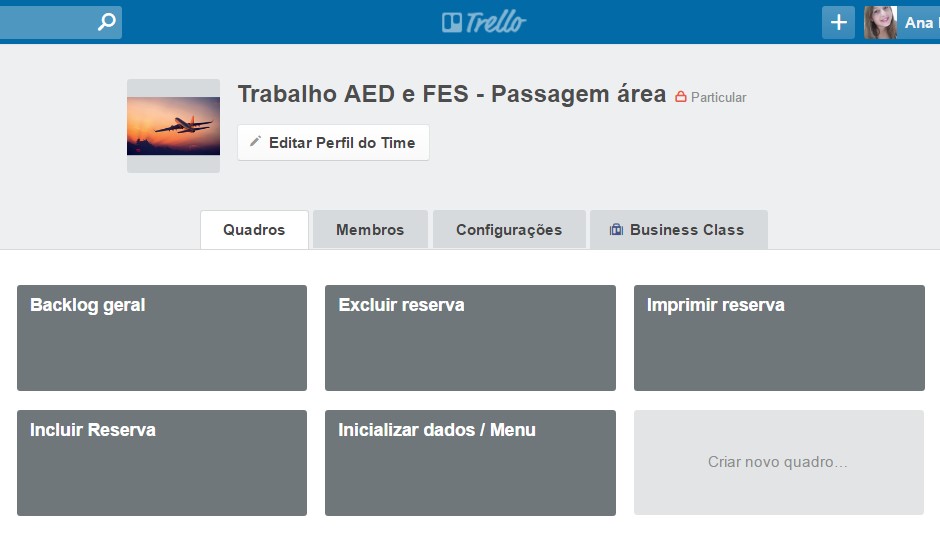
Exemplo:

Apresente o backlog do seu produto a cada semana.

Exemplo:

A Figura 1 apresenta o time criado no Trello com as divisões dos quadros. O quadro “*Backlog* geral” mostrará as divisões das funções entre os integrantes e quais tarefas deverão ser entregues por cada integrante por ordem de sprints. Os demais quadros são os *backlogs* de cada integrante, uma vez que cada um ficou responsável por uma função do sistema.

# Figura 1 – Time e quadros criados no Trello



**Fonte: Elaborado pelo autor**

A Figura 2 apresenta o protótipo de tela realizado no início do projeto mostrando no *backlog* geral a divisão das funções e quais tarefas os integrantes deveriam finalizar em cada sprint. Os itens marcados de verde devem ser realizados na primeira sprint, de amarelo na segunda, de laranja na terceira e de vermelho na quarta.

Este quadro serviu como base para cada um dos desenvolvedores criar o seu próprio *backlog*.

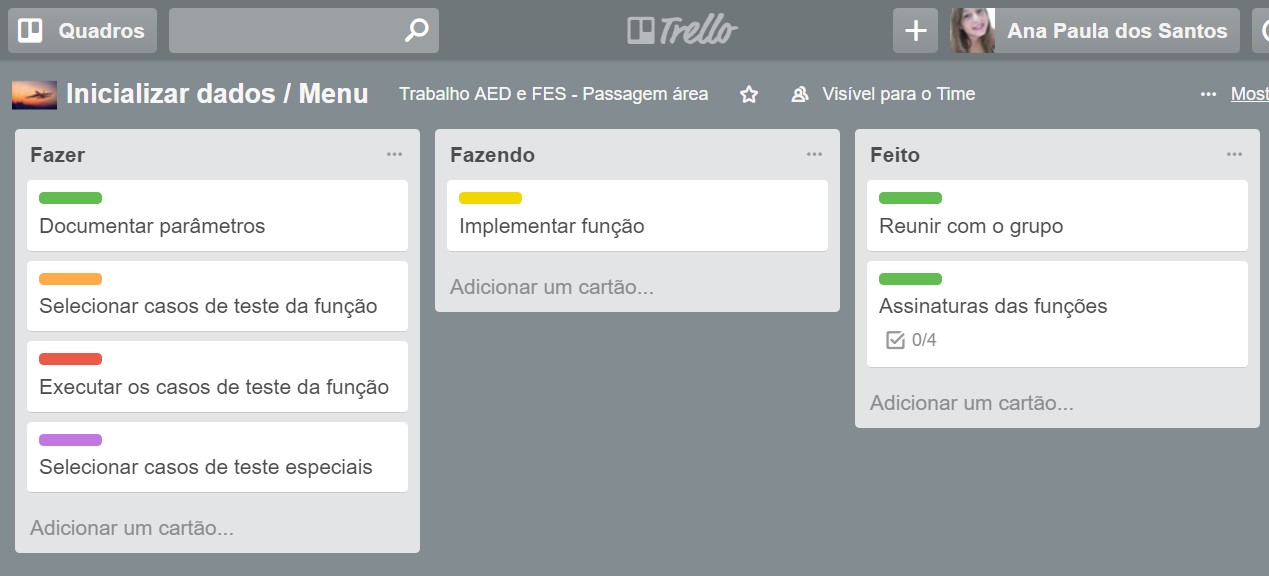
# Figura 2 - Backlog geral



**Fonte: Elaborado pelo autor**

As Figuras 3, 4, 5 e 6 demonstram o status do backlog do produto de cada integrante no final da primeira sprint.

# Figura 3 - Final primeira sprint menu e inicialização de dados



**Fonte: Elaborado pelo autor**

Observação: não é necessário usar Trello. Foi apenas um exemplo. Uma planilha para organizar o backlog ou um documento também funcionam, basta capturar o estado do backlog a cada semana ou sprint.

Fim do Exemplo.

**Lista de assinaturas das funções e parâmetros**

Explicação da estrutura de dados principal do programa.

Apresentação das assinaturas das funções

As funções e parâmetros utilizados no programa foram:

**1. void cadastrar\_cliente(Cliente clientes[], int num\_clientes)**

A função permite cadastrar um novo cliente no sistema. Solicita ao usuário informações como código, nome, endereço, telefone e CPF do cliente e armazena esses dados na estrutura clientes.

Esta função não possui retorno, pois modifica diretamente o array de clientes e o número total de clientes via ponteiro.

**Parâmetros:**

* clientes []: Array de estruturas Cliente.
* num\_clientes: Ponteiro para o número total de clientes.

**2. void cadastrar\_estadia(Estadia estadias[], int num\_estadias, Cliente clientes[], int num\_clientes, Quarto quartos[], int num\_quartos)**

A função permite cadastrar uma nova estadia no sistema. Solicita ao usuário informações como o código do cliente, quantidade de hóspedes, data de entrada e data de saída, verifica a disponibilidade do quarto e atualiza as informações necessárias.

Esta função não possui retorno, pois modifica diretamente os arrays de estadias, clientes e quartos, além do número total de estadias via ponteiro.

**Parâmetros:**

* estadias[]: Array de estruturas Estadia.
* num\_estadias: Ponteiro para o número total de estadias.
* clientes[]: Array de estruturas Cliente.
* num\_clientes: Número total de clientes cadastrados.
* quartos[]: Array de estruturas Quarto.
* num\_quartos: Número total de quartos cadastrados.

**3. void cadastrar\_funcionario(Funcionario funcionarios[], int *num\_funcionarios)***

A função permite cadastrar um novo funcionário no sistema. Solicita ao usuário informações como código, nome, CPF, e-mail, telefone, cargo e salário do funcionário, e armazena esses dados na estrutura funcionários.

Esta função não possui retorno, pois modifica diretamente o array de funcionários e o número total de funcionários via ponteiro.

**Parâmetros:**

* funcionarios[]: Array de estruturas Funcionario.
* num\_funcionarios: Ponteiro para o número total de funcionários.

**4. void cadastrar\_quarto(Quarto quartos[], int num\_quartos)**

A função permite cadastrar um novo quarto no sistema. Solicita ao usuário informações como número do quarto, quantidade de hóspedes, valor da diária e status (ocupado/desocupado) do quarto, e armazena esses dados na estrutura quartos.

Esta função não possui retorno, pois modifica diretamente o array de quartos e o número total de quartos via ponteiro.

**Parâmetros:**

* quartos[]: Array de estruturas Quarto.
* num\_quartos: Ponteiro para o número total de quartos.

**5. void checkout(Estadia estadias[], int num\_estadias, Cliente clientes[], int num\_clientes, Quarto quartos[], int num\_quartos)**

A função realiza o checkout de uma estadia específica, calculando o valor total a ser pago pelo cliente, alterando o status do quarto para desocupado e removendo a estadia do array de estadias.

Esta função não possui retorno, pois modifica diretamente os arrays de estadias e quartos, além do número total de estadias via ponteiro.

**Parâmetros:**

* estadias[]: Array de estruturas Estadia.
* num\_estadias: Ponteiro para o número total de estadias.
* clientes[]: Array de estruturas Cliente.
* num\_clientes: Número total de clientes cadastrados.
* quartos[]: Array de estruturas Quarto.
* num\_quartos: Número total de quartos cadastrados.

**6. void listar\_estadias\_de\_cliente(Estadia estadias[], int num\_estadias, int codigo\_cliente)**

A função lista todas as estadias associadas a um cliente específico, exibindo detalhes como data de entrada, data de saída, quantidade de diárias e número do quarto.

Esta função não possui retorno, pois apenas exibe as informações das estadias do cliente especificado.

**Parâmetros:**

* estadias[]: Array de estruturas Estadia.
* num\_estadias: Número total de estadias cadastradas.
* codigo\_cliente: Código do cliente cujas estadias serão listadas.

**7. *int calcular\_quantidade\_diarias(const char data\_entrada, const char* data\_saida)**

A função calcula a quantidade de diárias entre duas datas, considerando o formato de data especificado.

Retorna um inteiro representando a quantidade de diárias calculadas entre as datas de entrada e saída fornecidas.

**Parâmetros:**

* data\_entrada: String representando a data de entrada (formato "dd-mm-yyyy").
* data\_saida: String representando a data de saída (formato "dd-mm-yyyy").

**8. *Cliente pesquisar\_cliente(Cliente clientes[], int num\_clientes, const void* chave, int (*comparar)(const void*, const void\*))**

A função pesquisa um cliente no array de clientes com base em uma chave fornecida e uma função de comparação. Retorna um ponteiro para o cliente encontrado ou NULL se não for encontrado.

Retorna um ponteiro para a estrutura Cliente correspondente à chave fornecida ou NULL se nenhum cliente for encontrado.

**Parâmetros:**

* clientes[]: Array de estruturas Cliente.
* num\_clientes: Número total de clientes cadastrados.
* chave: Ponteiro genérico para a chave de pesquisa (pode ser o código ou o nome).
* comparar: Ponteiro para a função de comparação usada para comparar a chave com os clientes.

**9. Funcionario pesquisar\_funcionario(Funcionario funcionarios[], int num\_funcionarios, const void chave, int (comparar)(const void, const void\*))**

A função pesquisa um funcionário no array de funcionários com base em uma chave fornecida e uma função de comparação. Retorna um ponteiro para o funcionário encontrado ou NULL se não for encontrado.

Retorna um ponteiro para a estrutura Funcionario correspondente à chave fornecida ou NULL se nenhum funcionário for encontrado.

**Parâmetros:**

* funcionarios[]: Array de estruturas Funcionario.
* num\_funcionarios: Número total de funcionários cadastrados.
* chave: Ponteiro genérico para a chave de pesquisa (pode ser o código ou o nome).
* comparar: Ponteiro para a função de comparação usada para comparar a chave com os funcionários.

**10. void listar\_quartos(Quarto quartos[], int num\_quartos)**

A função exibe a lista de todos os quartos cadastrados, mostrando informações como número do quarto, quantidade de hóspedes, valor da diária e status (ocupado/desocupado).

Esta função não possui retorno, pois apenas exibe as informações dos quartos cadastrados.

**Parâmetros:**

* quartos[]: Array de estruturas Quarto.
* num\_quartos: Número total de quartos cadastrados.

**11. int comparar\_codigo(const void *a, const void b)***

A função compara os códigos de dois funcionários ou clientes e retorna um valor que indica a relação entre eles.

Retorna um valor negativo se a for menor que b, zero se forem iguais, e um valor positivo se a for maior que b.

**Parâmetros:**

* a: Ponteiro genérico para o primeiro código.
* b: Ponteiro genérico para o segundo código.

**12. int comparar\_funcionario\_nome(const void *a, const void b)***

A função compara os nomes de dois funcionários ou clientes e retorna um valor que indica a relação entre eles.

Retorna um valor negativo se a for menor que b, zero se forem iguais, e um valor positivo se a for maior que b.

**Parâmetros:**

* a: Ponteiro genérico para o primeiro nome.
* b: Ponteiro genérico para o segundo nome.

**13. void salvar\_arquivo (Quarto quartos[], int num\_quartos, Funcionario funcionarios[], int num\_funcionarios, Estadia estadias[], int num\_estadias, Cliente clientes[], int num\_clientes)**

A função grava as informações em um arquivo para persistência dos dados.

Esta função não possui retorno.

**Parâmetros:**

* quartos[]: Array de estruturas Quarto.
* num\_quartos: Número total de quartos cadastrados.
* funcionarios []: Array de estruturas Funcionario
* num\_funcionarios: Número total de funcionários cadastrados.
* estadias []: Array de estruturas de Estadia.
* num\_estadias: Número total de estadias cadastradas.
* Clientes []: Array de estruturas Clientes.
* num\_clientes: Número total de clientes cadastrados.

**14. void carregar\_arquivo (Quarto quartos[], int num\_quartos, Funcionario funcionarios[], int num\_funcionarios, Estadia estadias[], int num\_estadias, Cliente clientes[], int num\_clientes)**

A função lê as informações dos arquivos e as carrega para os array respectivos, atualizando o número total.

Esta função não possui retorno, pois modifica diretamente os array e o número via ponteiro.

**TESTES**

**Casos de teste do software:**

Os casos de teste englobam todo o código, uma vez que as funções devem receber parâmetros que podem estar dentro de outras funções ou do código principal.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entradas**  Função: cadastrar\_cliente | **Classes**  **Válidas** | **Resultado**  **Esperado** | **Classes Inválidas** | **Resultado**  **Esperado** |
| Cliente com código único, nome, endereço, telefone, CPF | Cliente com dados válidos, exemplo: {1, "João", "Rua A", "123456789", "111.222.333-44"} | Cliente cadastrado com sucesso, incremento no número de clientes | Cliente com código duplicado, campos vazios ou inválidos | Mensagem de erro, cliente não cadastrado |
| **Entradas**  Função: cadastrar\_funcionario | **Classes**  **Válidas** | **Resultado**  **Esperado** | **Classes Inválidas** | **Resultado**  **Esperado** |
| Funcionario com código único, nome, telefone, cargo, salário | Funcionário com dados válidos, exemplo: {1, "Maria", "987654321", "Gerente", 3000.0} | Funcionário cadastrado com sucesso, incremento no número de funcionários | Funcionário com código duplicado, campos vazios ou inválidos | Mensagem de erro, funcionário não cadastrado |
| **Entradas**  Função: cadastrar\_quarto | **Classes**  **Válidas** | **Resultado**  **Esperado** | **Classes Inválidas** | **Resultado**  **Esperado** |
| Quarto com número único, quantidade de hóspedes, valor diária, status | Quarto com dados válidos, exemplo: {101, 2, 150.0, "desocupado"} | Quarto cadastrado com sucesso, incremento no número de quartos | Quarto com número duplicado, campos vazios ou inválidos | Mensagem de erro, quarto não cadastrado |
| **Entradas**  Função: cadastrar\_estadia | **Classes**  **Válidas** | **Resultado**  **Esperado** | **Classes Inválidas** | **Resultado**  **Esperado** |
| Estadia com código de cliente válido, quantidade de hóspedes, data de entrada e saída, quarto disponível | Estadia com dados válidos, exemplo: {1, 2, "2023-06-01", "2023-06-05", 101} | Estadia cadastrada com sucesso, alteração do status do quarto | Estadia com cliente ou quarto inexistente, datas inválidas, quarto ocupado | Mensagem de erro, estadia não cadastrada |
| **Entradas**  Função: checkout | **Classes**  **Válidas** | **Resultado**  **Esperado** | **Classes Inválidas** | **Resultado**  **Esperado** |
| Estadia existente, cliente válido, quarto válido | Estadia com dados válidos, exemplo: {1, 2, "2023-06-01", "2023-06-05", 101} | Checkout realizado, cálculo do valor total, alteração do status do quarto | Estadia inexistente, cliente ou quarto inválido. | Mensagem de erro, checkout não realizado |
| **Entradas**  Função: listar\_estadias\_de\_cliente | **Classes**  **Válidas** | **Resultado**  **Esperado** | **Classes Inválidas** | **Resultado**  **Esperado** |
| Código de cliente existente | Cliente existente com estadias, exemplo: 1 | Listagem de todas as estadias do cliente | Cliente inexistente, código inválido | Mensagem de erro, estadias não listadas |
| **Entradas**  Função: pesquisar\_cliente | **Classes**  **Válidas** | **Resultado**  **Esperado** | **Classes Inválidas** | **Resultado**  **Esperado** |
| Chave de pesquisa válida (código ou nome), clientes existentes | Chave válida, exemplo: código 1 ou nome "João" | Cliente encontrado, dados exibidos | Chave inválida, cliente inexistente | Mensagem de erro, estadias não listadas |
| **Entradas**  Função: pesquisar\_funcionario | **Classes**  **Válidas** | **Resultado**  **Esperado** | **Classes Inválidas** | **Resultado**  **Esperado** |
| Chave de pesquisa válida (código ou nome), funcionários existentes | Chave válida, exemplo: código 1 ou nome "Maria" | Funcionário encontrado, dados exibidos | Chave inválida, funcionário inexistente | Mensagem de erro, funcionário não encontrado |

**Relatório de Execução de Testes**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Entradas** | **Classes válidas** | **Resultado Esperado** | | **Classes Inválidas** | | **Resultado Esperado** |
| Cliente com código único, nome, endereço, telefone, CPF | Cliente com dados válidos, exemplo: {1, "João", "Rua A", "123456789", "111.222.333-44"} | Cliente cadastrado com sucesso | | Cliente com código duplicado, campos vazios ou inválidos | | Mensagem de erro, cliente não cadastrado |
|  | | | | | | |
|  | | | | | | |
| **Relatório de execução de testes – Cadastro de Clientes** | | | | | | |
| Entradas | | | Resultado | | Aprovado? | |
| Código: 1, Nome: "João", Endereço: "Rua A", Telefone: "123456789", CPF: "111.222.333-44" | | | Cliente cadastrado com sucesso | | Sim | |
| Código: 1 (duplicado), Nome: "Maria", Endereço: "Rua B", Telefone: "987654321", CPF: "222.333.444-55" | | | Mensagem de erro, cliente não cadastrado | | Sim | |
| Código: 2, Nome: "", Endereço: "Rua C", Telefone: "543216789", CPF: "333.444.555-66" | | | Mensagem de erro, cliente não cadastrado | | Sim | |
| **Entradas** | **Classes válidas** | **Resultado Esperado** | | **Classes Inválidas** | | **Resultado Esperado** |
| Funcionario com código único, nome, telefone, cargo, salário | Funcionário com dados válidos, exemplo: {1, "Maria", "987654321", "Gerente", 3000.0} | Funcionário cadastrado com sucesso, incremento no número de funcionários | | Funcionário com código duplicado, campos vazios ou inválidos | | Mensagem de erro, funcionário não cadastrado |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Relatório de execução de testes – Cadastro de Funcionários** | | | | | | |
| Entradas | | | Resultado | | Aprovado? | |
| Funcionario com código 1, nome "Maria", telefone "987654321", cargo "Gerente", salário 3000.0 | | | Funcionário cadastrado com sucesso | | Sim | |
| Funcionario com código 1 (duplicado), nome "Ana", telefone "654321987", cargo "Recepcionista", salário 2000.0 | | | Mensagem de erro, funcionário não cadastrado | | Sim | |
| Funcionario com código 2, nome "Pedro", telefone "", cargo "Manobrista", salário 1500.0 | | | Mensagem de erro, funcionário não cadastrado | | Sim | |
| **Entradas** | **Classes válidas** | **Resultado Esperado** | | **Classes Inválidas** | | **Resultado Esperado** |
| Quarto com número único, quantidade de hóspedes, valor diária, status | Quarto com dados válidos, exemplo: {101, 2, 150.0, "desocupado"} | Quarto cadastrado com sucesso, incremento no número de quartos | | Quarto com número duplicado, campos vazios ou inválidos | | Mensagem de erro, cliente não cadastrado |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Relatório de execução de testes – Cadastro de Quarto** | | | | | | |
| Entradas | | | Resultado | | Aprovado? | |
| Quarto com número 101, quantidade de hóspedes 2, valor diária 150.0, status "desocupado" | | | Quarto cadastrado com sucesso | | Sim | |
| Quarto com número 101 (duplicado), quantidade de hóspedes 2, valor diária 150.0, status "desocupado" | | | Mensagem de erro, quarto não cadastrado | | Sim | |
| Quarto com número 102, quantidade de hóspedes 0, valor diária 200.0, status "desocupado" | | | Mensagem de erro, quarto não cadastrado | | Sim | |
| **Entradas** | **Classes válidas** | **Resultado Esperado** | | **Classes Inválidas** | | **Resultado Esperado** |
| Estadia com código de cliente válido, quantidade de hóspedes, data de entrada e saída, quarto disponível | Estadia com dados válidos, exemplo: {1, 2, "2023-06-01", "2023-06-05", 101} | Estadia cadastrada com sucesso, alteração do status do quarto | | Estadia com cliente ou quarto inexistente, datas inválidas, quarto ocupado | | Mensagem de erro, cliente não cadastrado |

|  |  |  |
| --- | --- | --- |
| **Relatório de execução de testes – Cadastro de Estadia** | | |
| Entradas | Resultado | Aprovado? |
| Estadia com código cliente 1, quantidade de hóspedes 2, data de entrada "2023-06-01", data de saída "2023-06-05", quarto 101 | Estadia cadastrada com sucesso | Sim |
| Estadia com código cliente 1, quantidade de hóspedes 2, data de entrada "2023-06-01", data de saída "2023-06-05", quarto 101 (ocupado) | Mensagem de erro, estadia não cadastrada | Sim |
| Estadia com código cliente 2 (inexistente), quantidade de hóspedes 2, data de entrada "2023-06-01", data de saída "2023-06-05", quarto 102 | Mensagem de erro, estadia não cadastrada | Sim |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entradas** | **Classes válidas** | **Resultado Esperado** | **Classes Inválidas** | **Resultado Esperado** |
| Estadia existente, cliente válido, quarto válido | Estadia com dados válidos, exemplo: {1, 2, "2023-06-01", "2023-06-05", 101} | Checkout realizado, cálculo do valor total, alteração do status do quarto | Estadia inexistente, cliente ou quarto inválido. | Mensagem de erro, checkout não realizado |

|  |  |  |
| --- | --- | --- |
| **Relatório de execução de testes – Checkout** | | |
| Entradas | Resultado | Aprovado? |
| Checkout da estadia com código cliente 1, quantidade de hóspedes 2, data de entrada "2023-06-01", data de saída "2023-06-05", quarto 101 | Checkout realizado, cálculo do valor total, quarto desocupado | Sim |
| Checkout da estadia com código cliente 2 (inexistente), quantidade de hóspedes 2, data de entrada "2023-06-01", data de saída "2023-06-05", quarto 101 | Mensagem de erro, checkout não realizado | Sim |
| Checkout da estadia com código cliente 1, quantidade de hóspedes 2, data de entrada "2023-06-01", data de saída "2023-06-05", quarto 102 (inexistente) | Mensagem de erro, checkout não realizado | Sim |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entradas** | **Classes válidas** | **Resultado Esperado** | **Classes Inválidas** | **Resultado Esperado** |
| Código de cliente existente | Cliente existente com estadias, exemplo: 1 | Listagem de todas as estadias do cliente | Cliente inexistente, código inválido | Mensagem de erro, estadias não listadas |

|  |  |  |
| --- | --- | --- |
| **Relatório de execução de testes –** Listar Estadias de Cliente | | |
| Entradas | Resultado | Aprovado? |
| Código de cliente 1 | Listagem de todas as estadias do cliente | Sim |
| Código de cliente 2 (inexistente) | Mensagem de erro, estadias não listadas | Sim |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entradas** | **Classes válidas** | **Resultado Esperado** | **Classes Inválidas** | **Resultado Esperado** |
| Chave de pesquisa válida (código ou nome), clientes existentes | Chave válida, exemplo: código 1 ou nome "João" | Cliente encontrado, dados exibidos | Chave inválida, cliente inexistente | Mensagem de erro, estadias não listadas |

|  |  |  |
| --- | --- | --- |
| **Relatório de execução de testes –** Pesquisar Clientes | | |
| Entradas | Resultado | Aprovado? |
| Código 1 | Cliente encontrado, dados exibidos | Sim |
| Nome "João" | Cliente encontrado, dados exibidos | Sim |
| Código 99 | Mensagem de erro, cliente não encontrado | Sim |
| Nome "Carlos" | Mensagem de erro, cliente não encontrado | Sim |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entradas** | **Classes válidas** | **Resultado Esperado** | **Classes Inválidas** | **Resultado Esperado** |
| Chave de pesquisa válida (código ou nome), funcionários existentes | Chave válida, exemplo: código 1 ou nome "Maria" | Funcionário encontrado, dados exibidos | Chave inválida, funcionário inexistente | Mensagem de erro, funcionário não encontrado |

|  |  |  |
| --- | --- | --- |
| **Relatório de execução de testes –** Pesquisar Funcionários | | |
| Entradas | Resultado | Aprovado? |
| Código 1 | Funcionário encontrado, dados exibidos | Sim |
| Nome "Maria" | Funcionário encontrado, dados exibidos | Sim |
| Código 88 | Mensagem de erro, funcionário não encontrado | Sim |
| Nome "Guilherme" | Mensagem de erro, funcionário não encontrado | Sim |

**O código C do programa e suas funções incluindo a implementação automatizada dos casos de testes.**