PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA
UNIDADE EDUCACIONAL CORAÇÃO EUCARÍSTICO
Bacharelado em Engenharia de Software

Isabella Luiza Dias dos Santos

Gustavo Viana dos Santos

HotelBooker

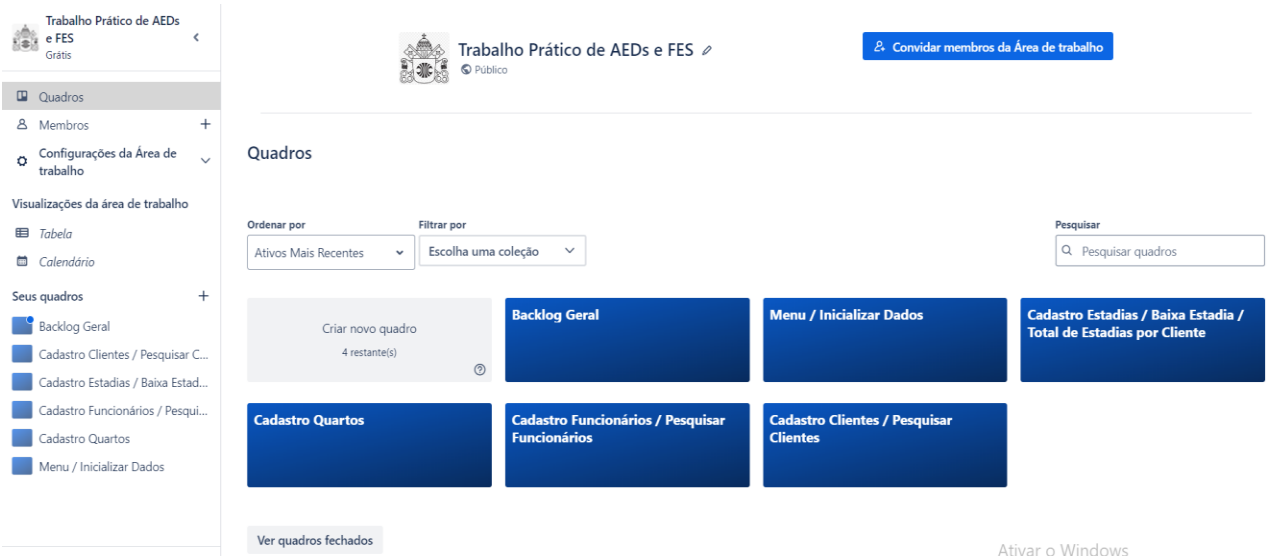
Apresentação:

HotelBooker é um sistema de software desenvolvido para gerenciar e organizar os dados de uma empresa hoteleira de forma eficiente.

Backlog do Produto:

Link: https://trello.com/w/hotelbooker

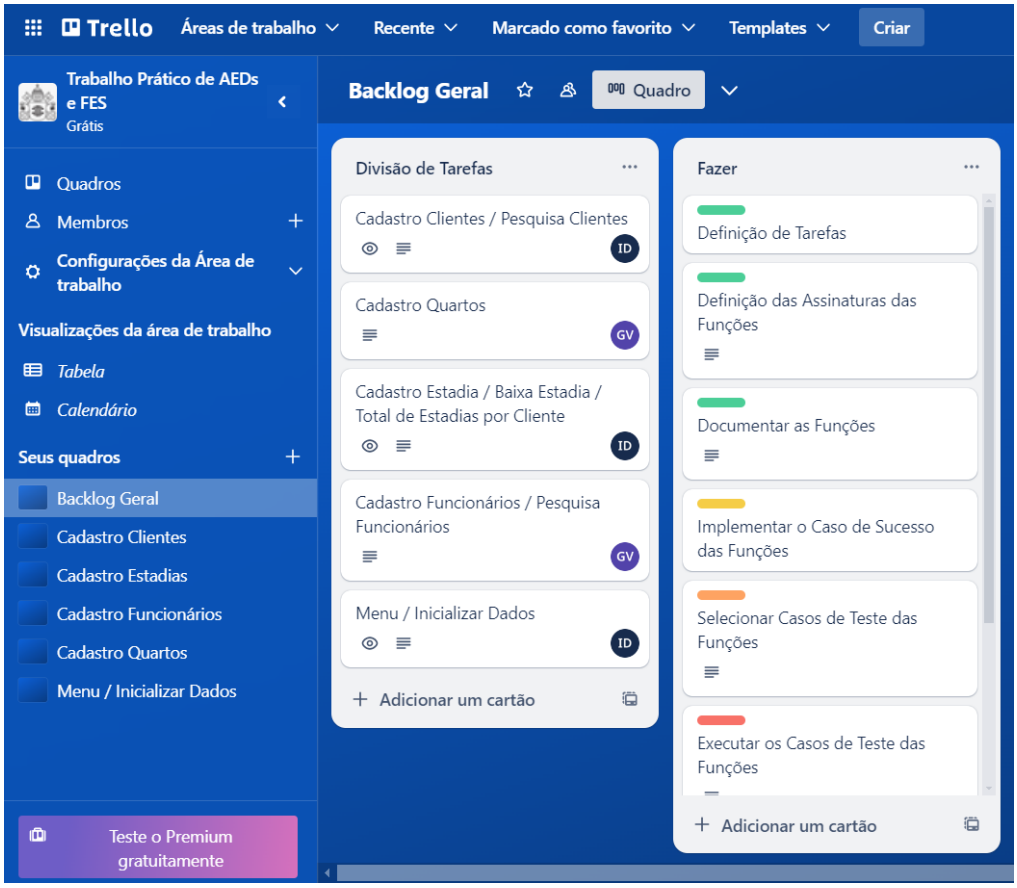
Figura 1 – Time e quadros criados no Trello



Fonte: Elaborado pelos autores

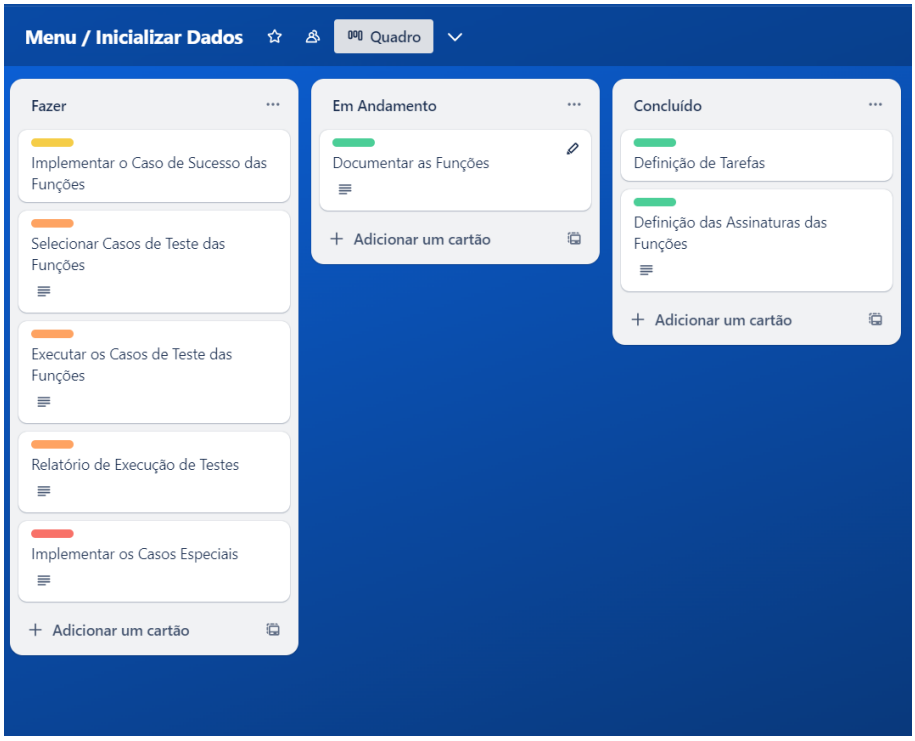
Backlog Geral:

Figura 2 – Backlog geral



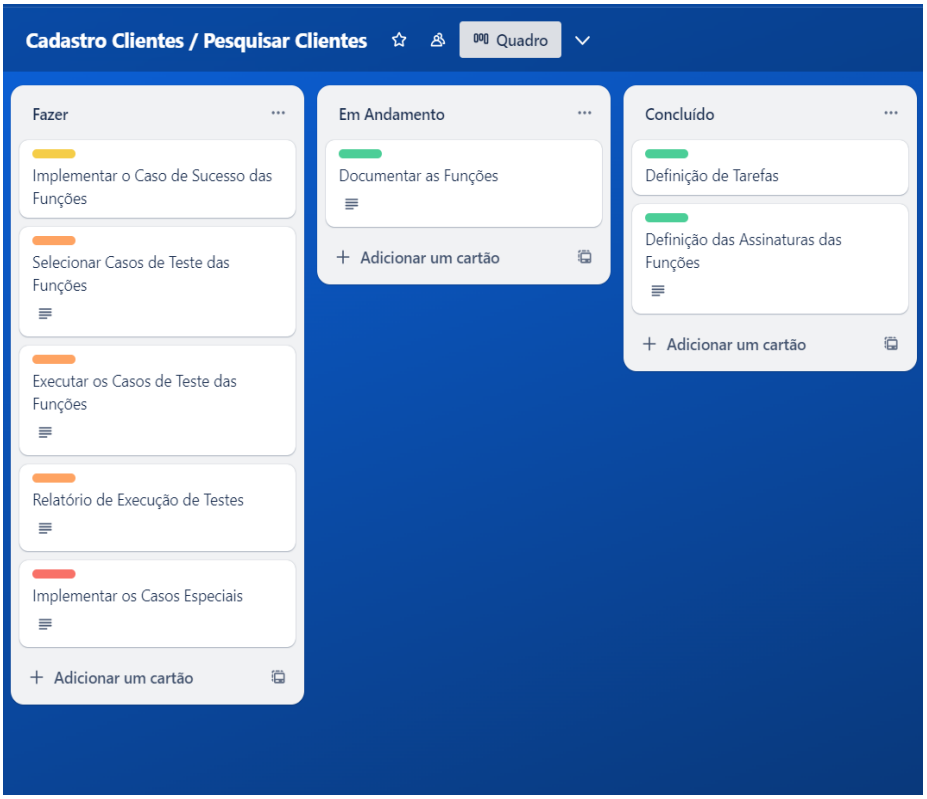
Fonte: Elaborado pelos autores

Figura 3 – Final Primeira Sprint Menu e Inicialização de Dados



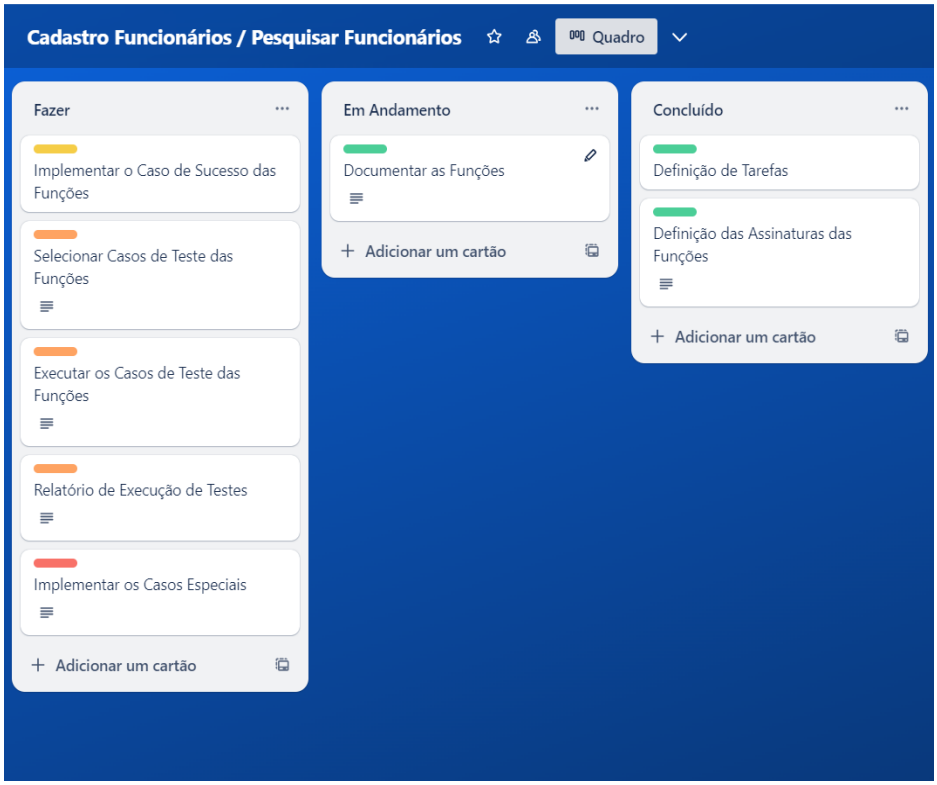
Fonte: Elaborado pelos autores

Figura 4 – Final Primeira Sprint Cadastrar Clientes / Pesquisar Clientes



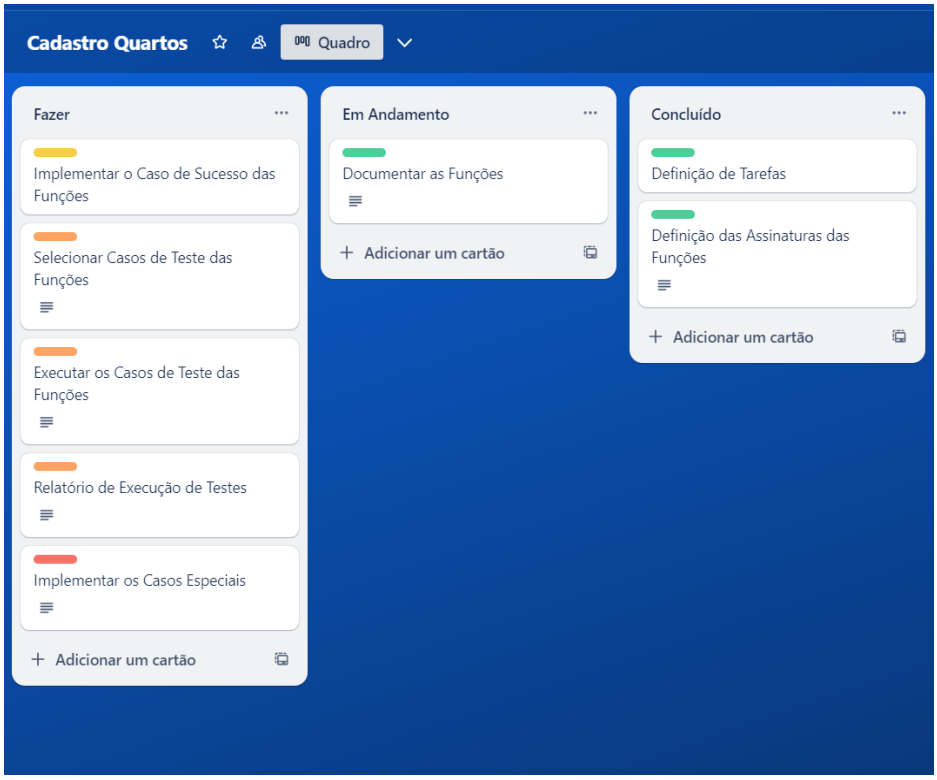
Fonte: Elaborado pelos autores

Figura 5 – Final Primeira Sprint Cadastrar Funcionários / Pesquisar Funcionários



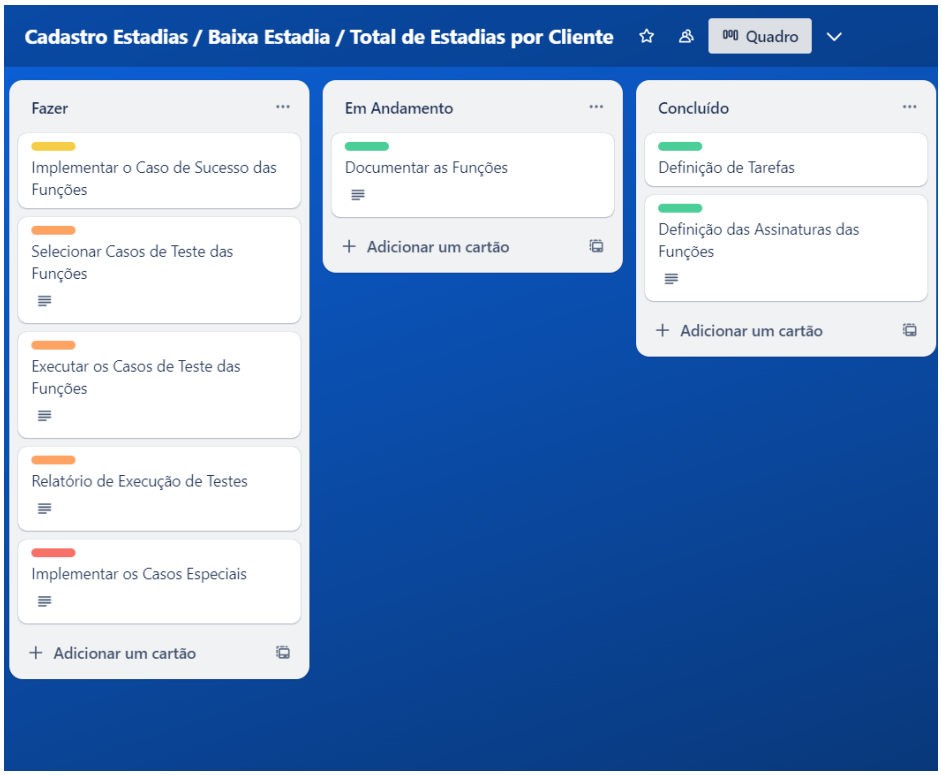
Fonte: Elaborado pelos autores

Figura 6 – Final Primeira Sprint Cadastrar Quartos



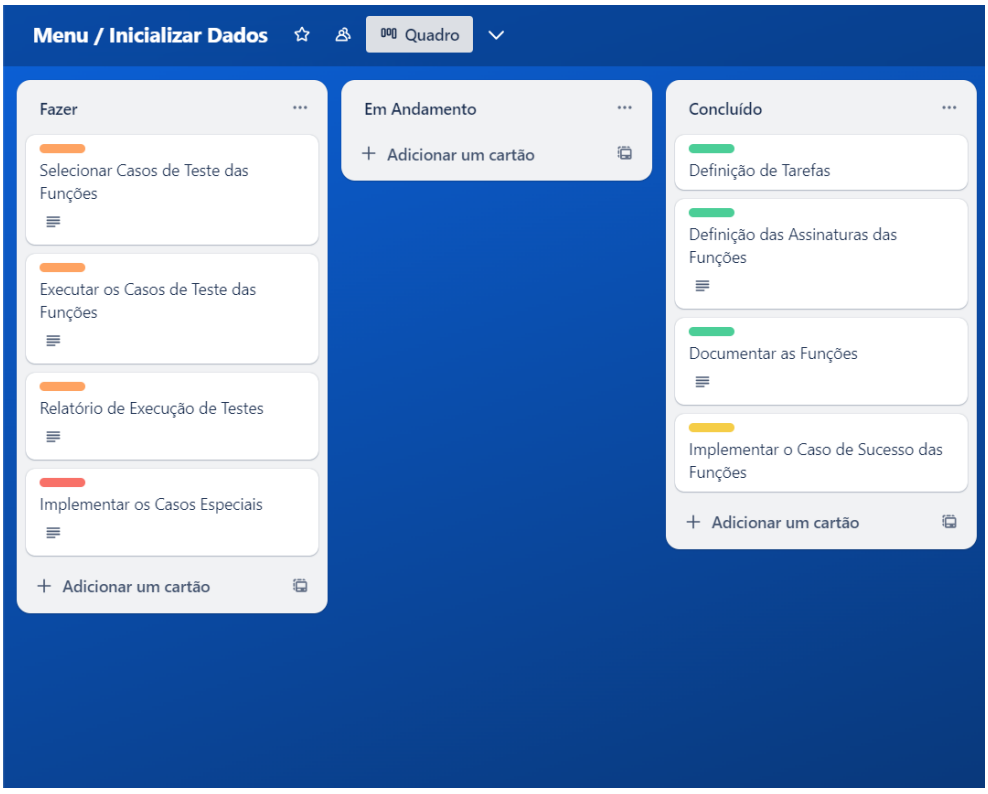
Fonte: Elaborado pelos autores

Figura 7 – Final Primeira Sprint Cadastrar Estadias / Baixa Estadia / Total de Estadias por Cliente



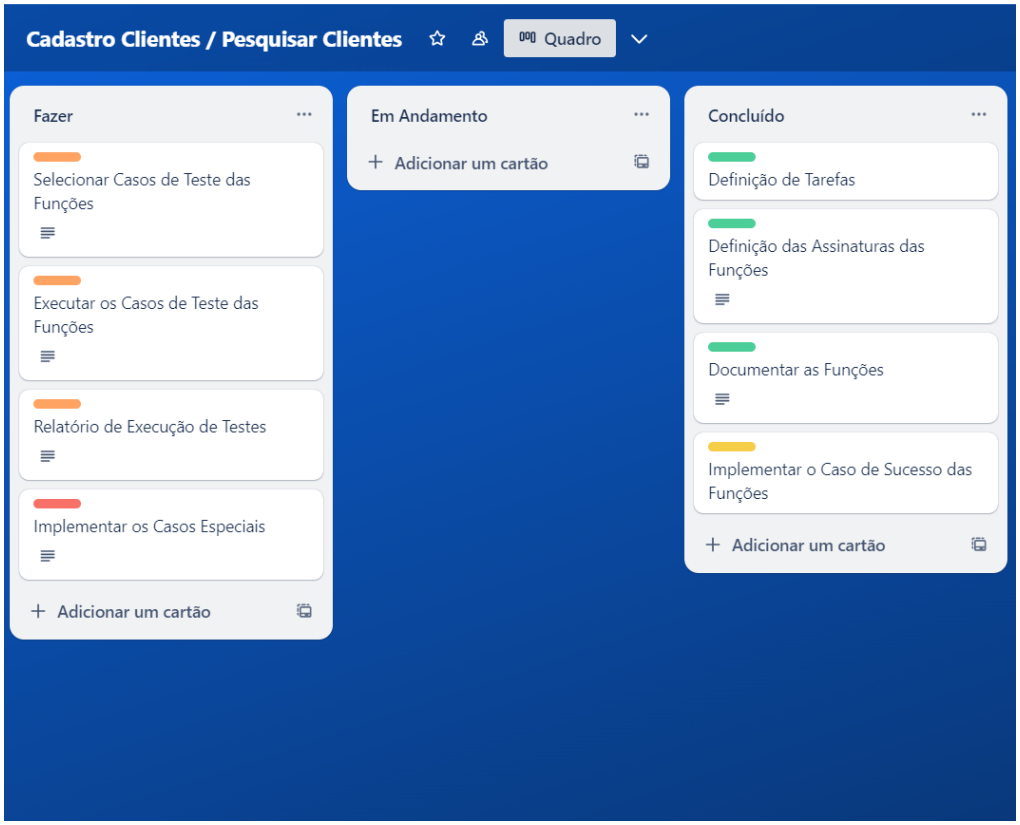
Fonte: Elaborado pelos autores

Figura 8 – Final Segunda Sprint Menu e Inicialização de Dados



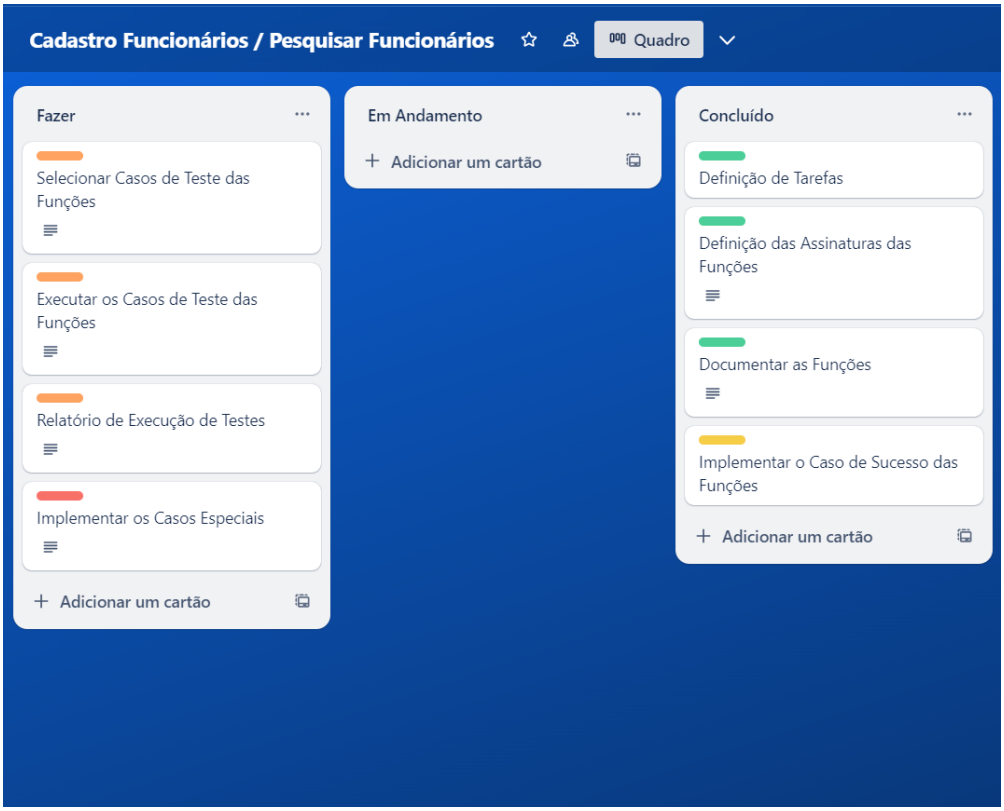
Fonte: Elaborado pelos autores

Figura 9 – Final Segunda Sprint Cadastrar Clientes / Pesquisar Clientes



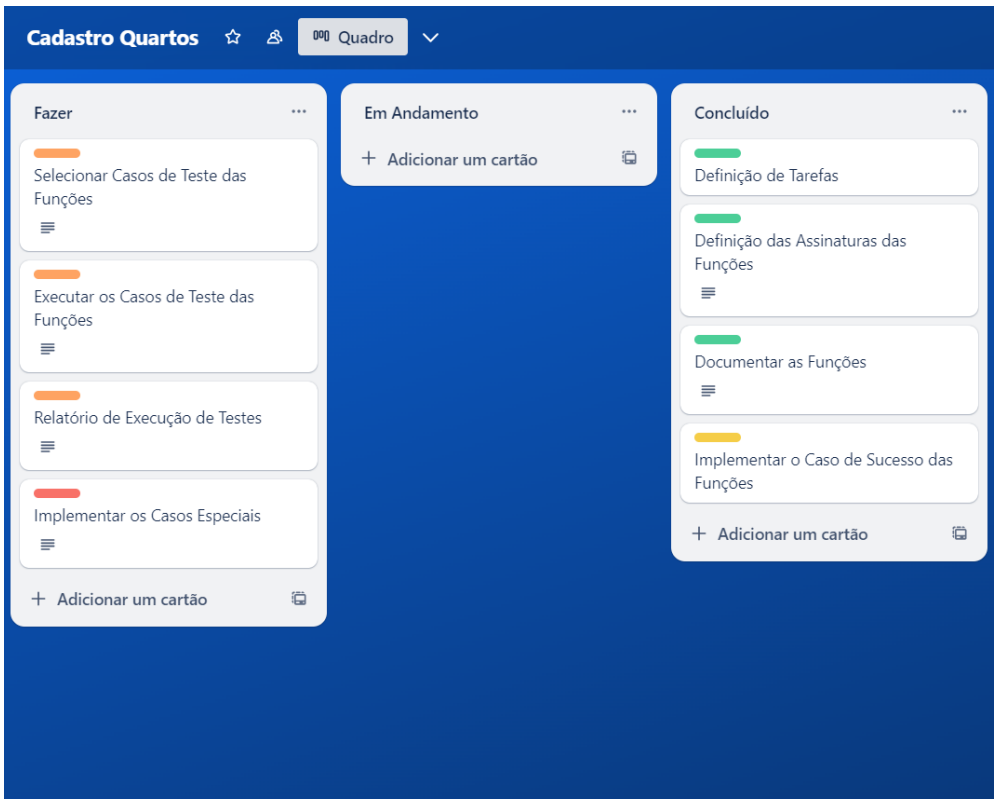
Fonte: Elaborado pelos autores

Figura 10 – Final Segunda Sprint Cadastrar Funcionários / Pesquisar Funcionários



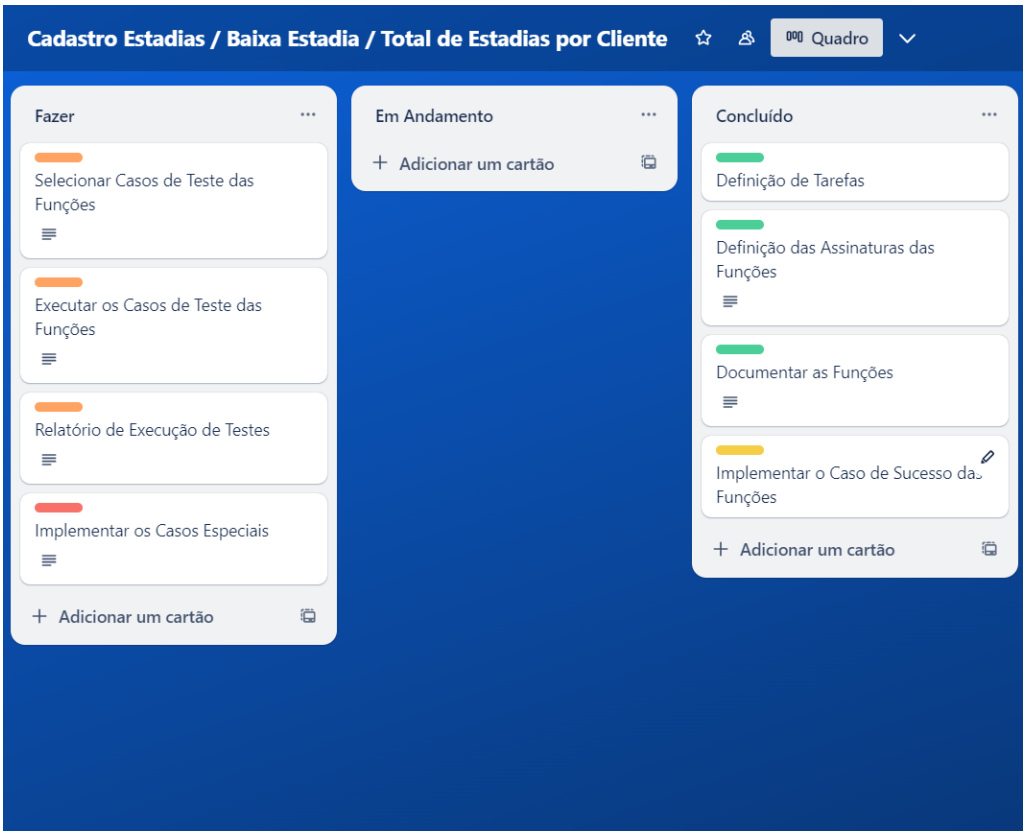
Fonte: Elaborado pelos autores

Figura 11 – Final Segunda Sprint Cadastrar Quartos



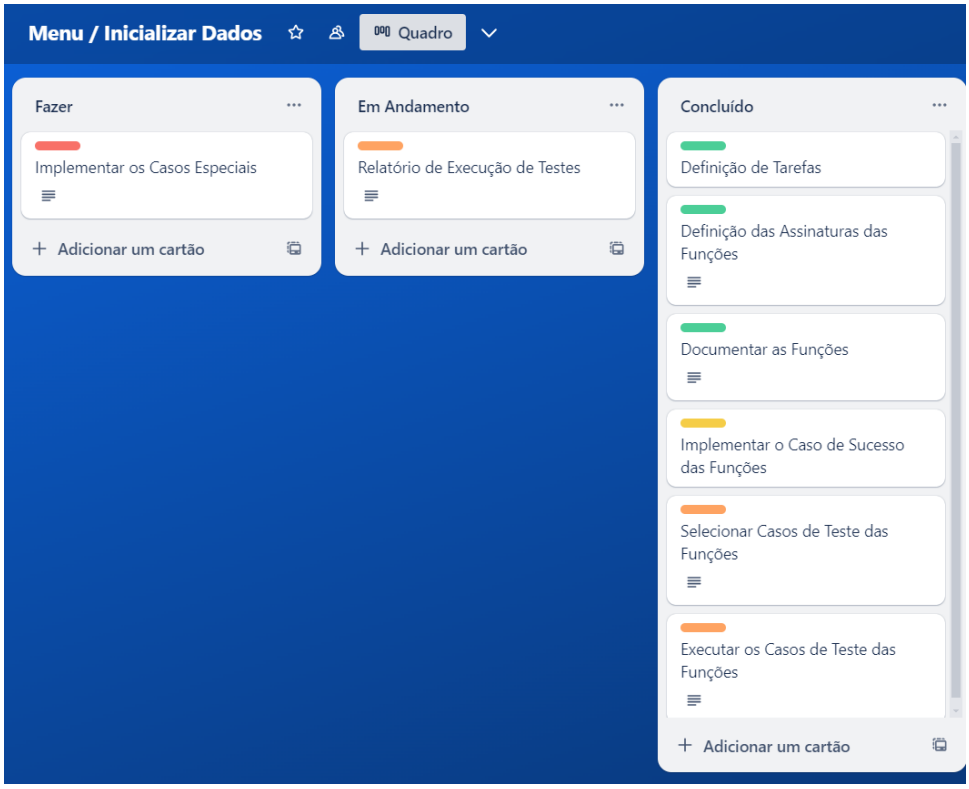
Fonte: Elaborado pelos autores

Figura 12 – Final Segunda Sprint Cadastrar Estadias / Baixa Estadia / Total de Estadias por Cliente



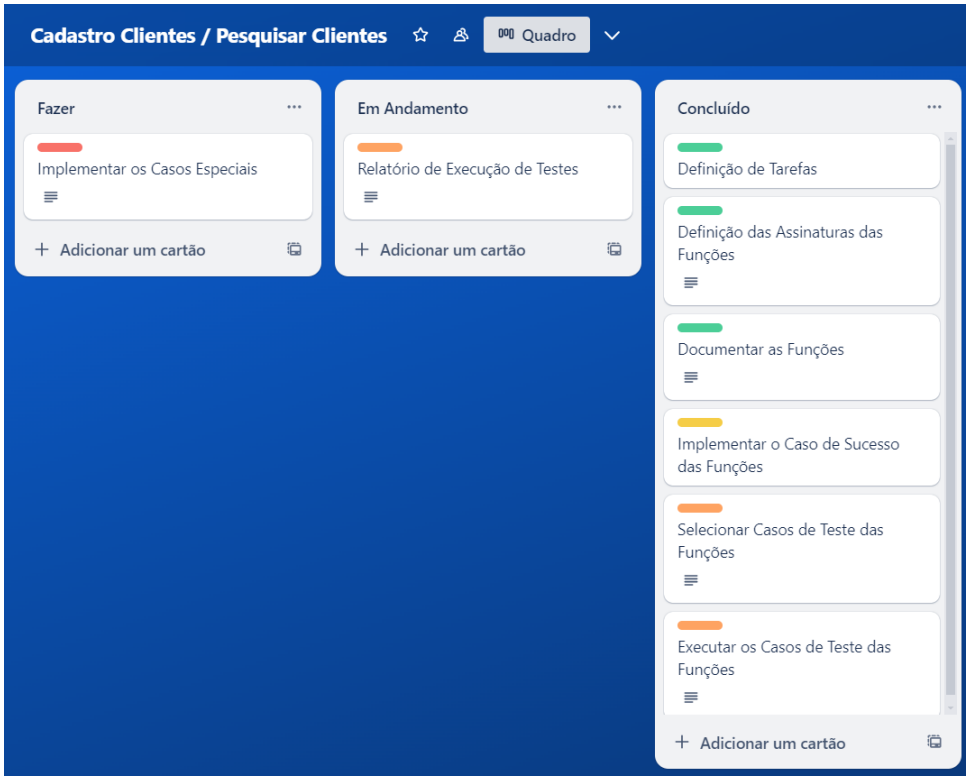
Fonte: Elaborado pelos autores

Figura 13 – Final Terceira Sprint Menu e Inicialização de Dados



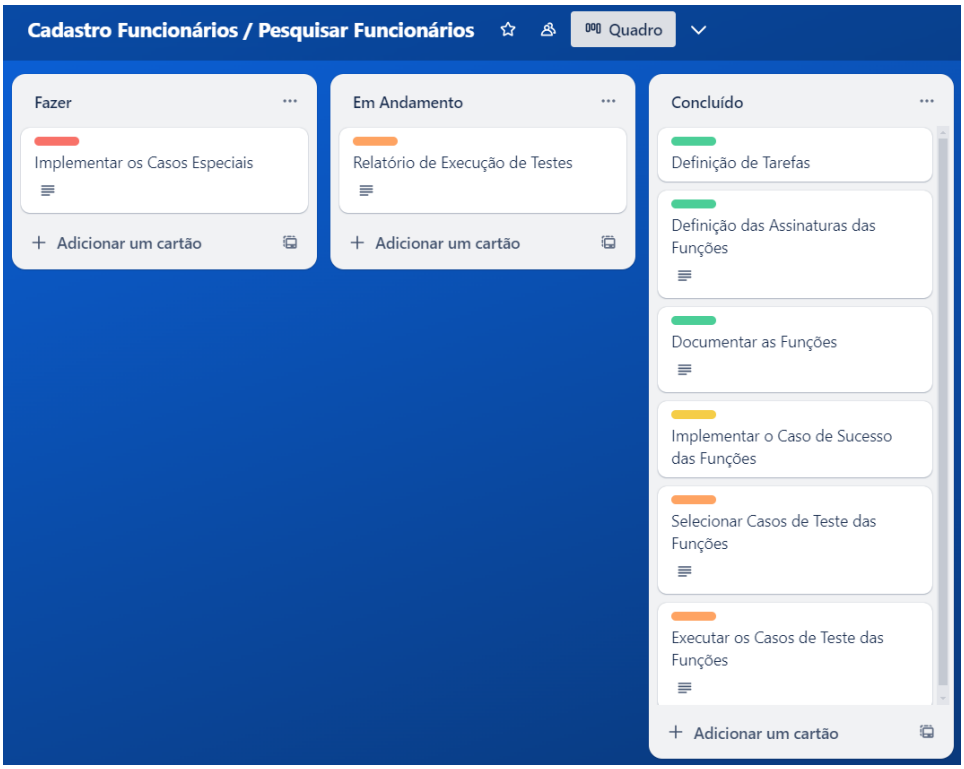
Fonte: Elaborado pelos autores

Figura 14 – Final Terceira Sprint Cadastrar Clientes / Pesquisar Clientes



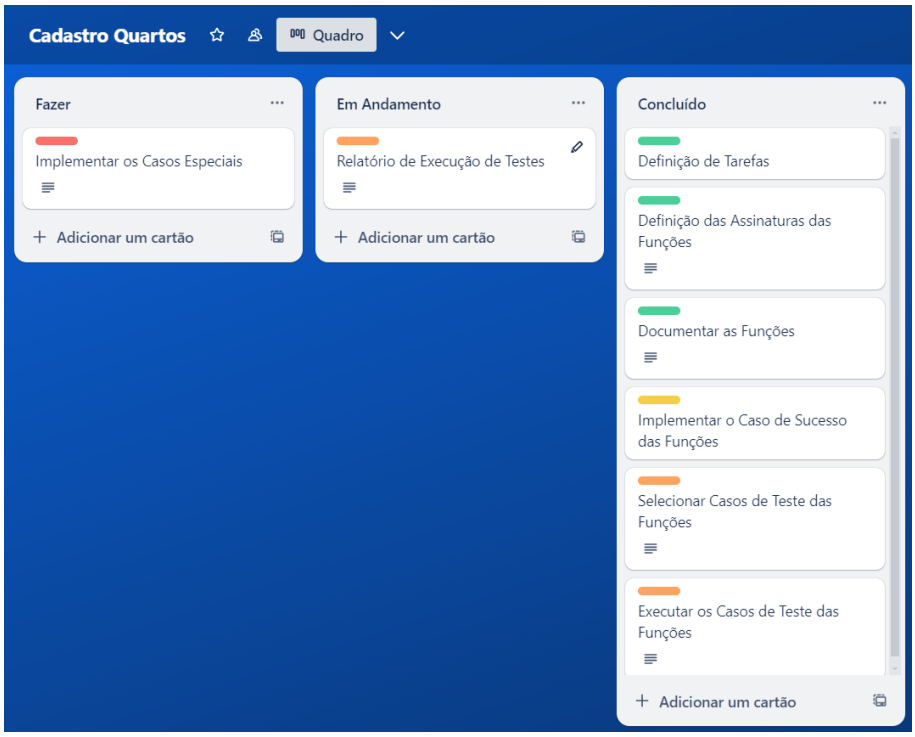
Fonte: Elaborado pelos autores

Figura 15 – Final Terceira Sprint Cadastrar Funcionários / Pesquisar Funcionários



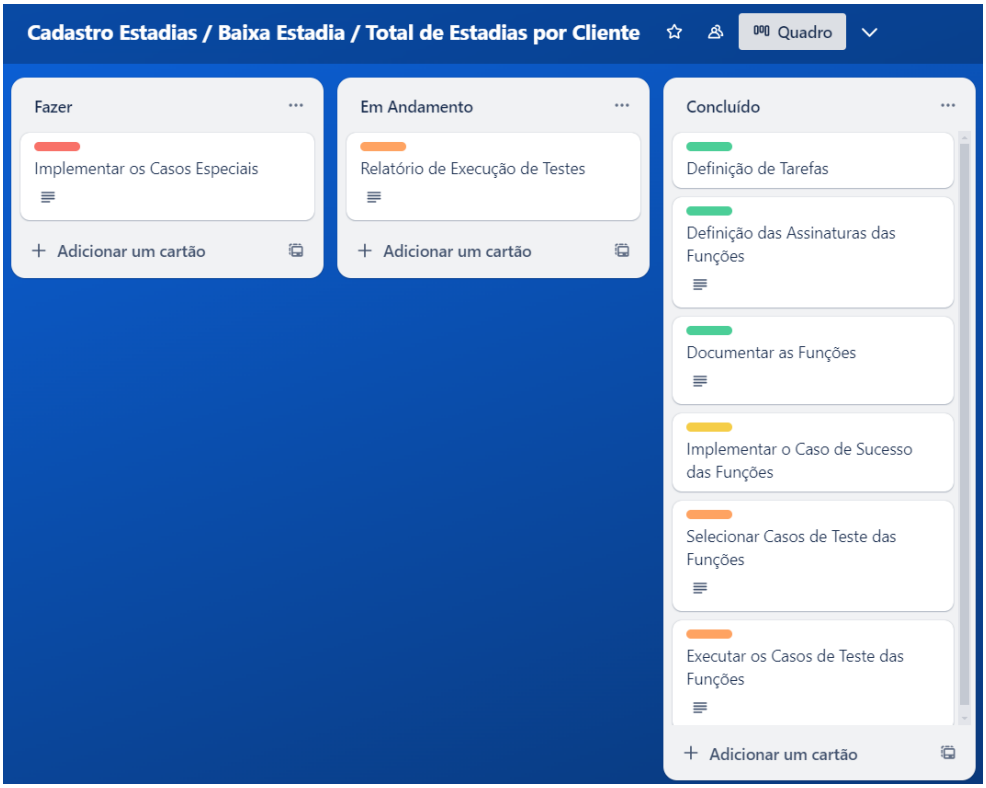
Fonte: Elaborado pelos autores

Figura 16 – Final Terceira Sprint Cadastrar Quartos



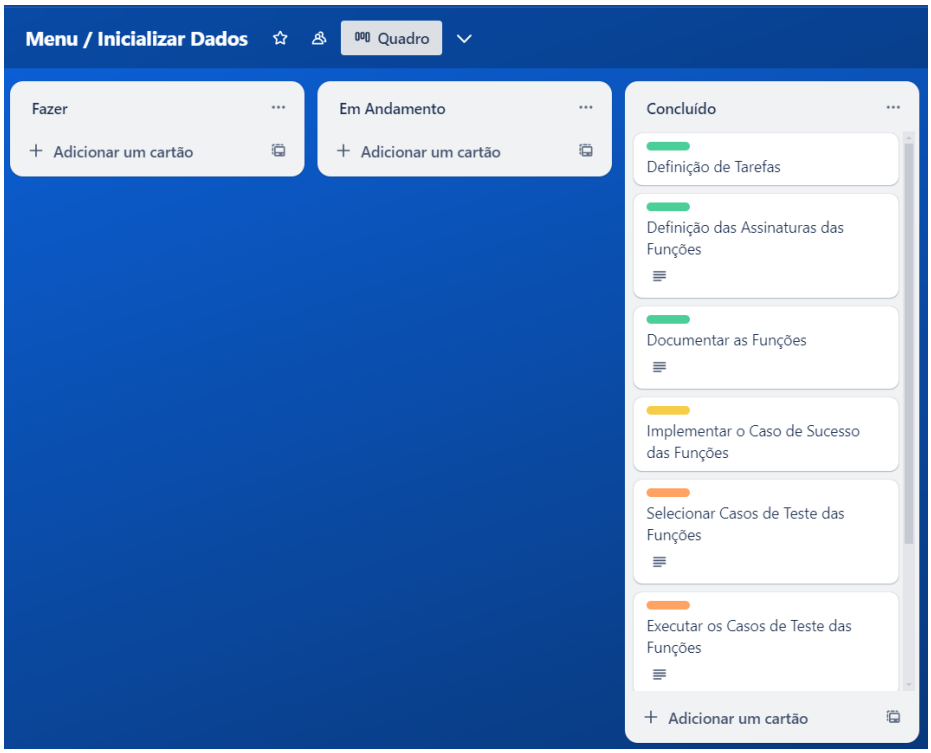
Fonte: Elaborado pelos autores

Figura 17 – Final Terceira Sprint Cadastrar Estadias / Baixa Estadia / Total de Estadias por Cliente



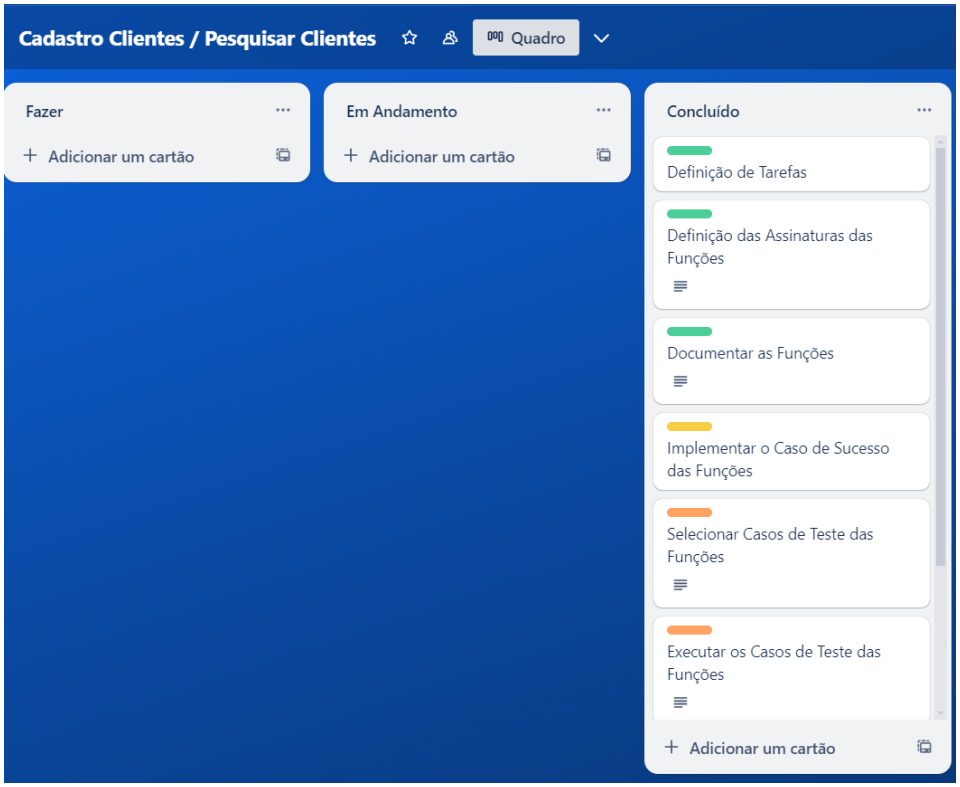
Fonte: Elaborado pelos autores

Figura 18 – Final Quarta Sprint Menu e Inicialização de Dados



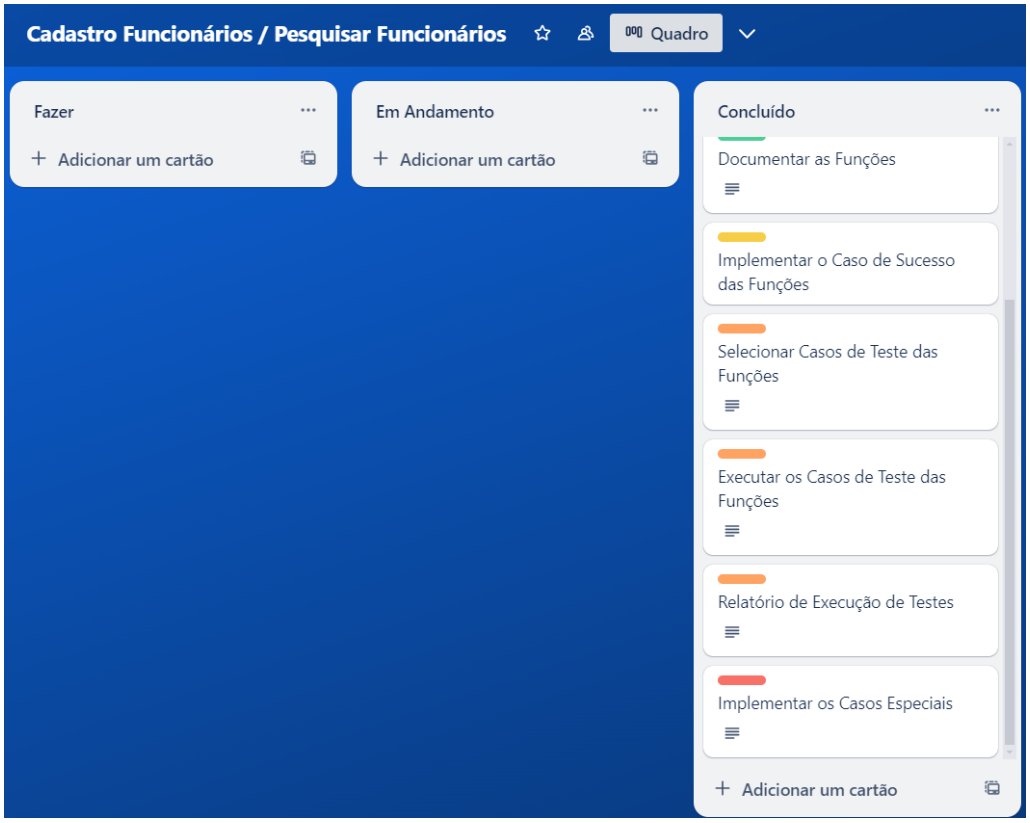
Fonte: Elaborado pelos autores

Figura 19 – Final Quarta Sprint Cadastrar Clientes / Pesquisar Clientes



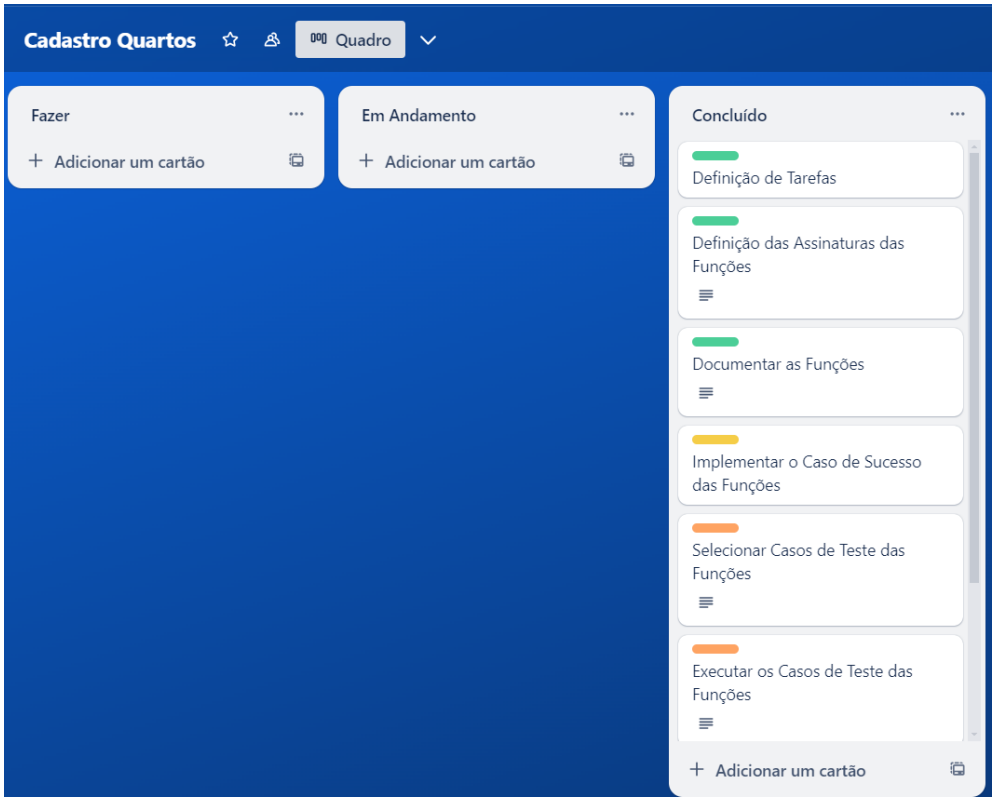
Fonte: Elaborado pelos autores

Figura 20 – Final Quarta Sprint Cadastrar Funcionários / Pesquisar Funcionários



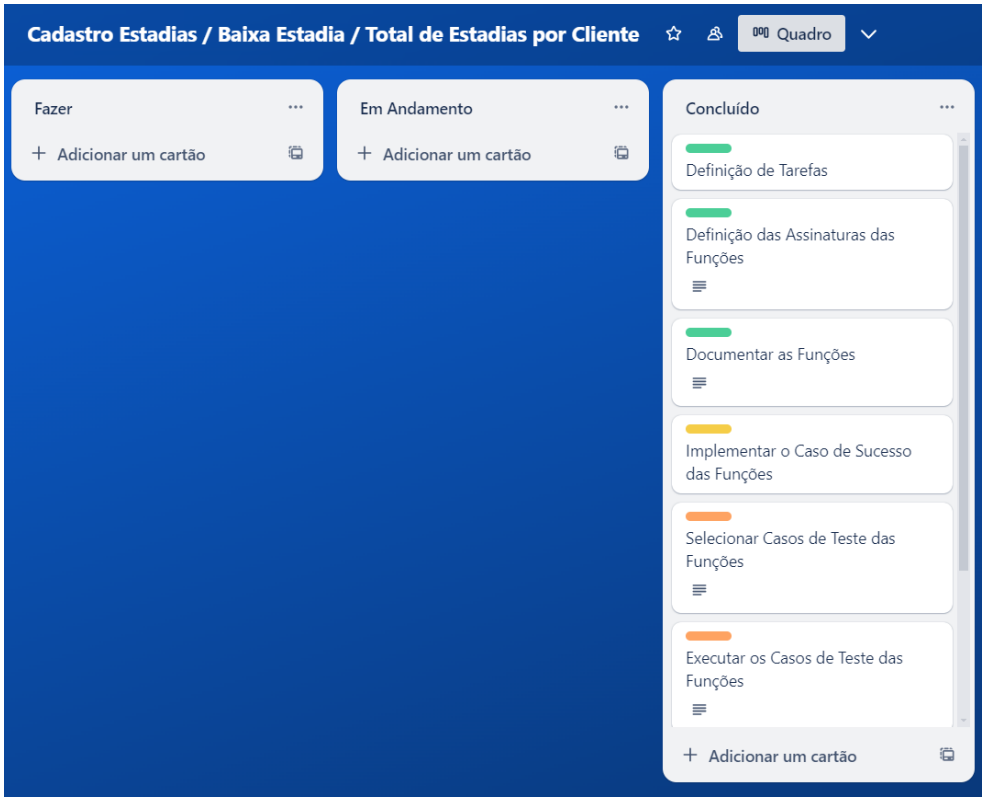
Fonte: Elaborado pelos autores

Figura 21 – Final Quarta Sprint Cadastrar Quartos



Fonte: Elaborado pelos autores

Figura 22 – Final Quarta Sprint Cadastrar Estadias / Baixa Estadia



Fonte: Elaborado pelos autores

Lista de Assinaturas da Funções e Parâmetros

As funções e parâmetros utilizados no programa foram:

1. **void cadastrarClientes(Cliente clientes[], int *numClientes)**

   Função para cadastrar novos clientes. Recebe como parâmetros o array do tipo Cliente, onde novos clientes são armazenados, e o ponteiro que armazena o número atual de clientes. Este valor é incrementado a cada novo cliente cadastrado.
   Essa função não possui retorno.

2. **void carregarClientes(Cliente clientes[], int *numClientes)**

   Função para carregar os dados dos clientes do arquivo Clientes.txt. Recebe como parâmetros o array do tipo Cliente, onde os dados carregados são armazenados, e o ponteiro que armazena o número atual de clientes. Este valor é atualizado com o número de clientes carregados.
   Essa função não possui retorno.

3. void cadastrarFuncionarios(Funcionario funcionarios[], int *numFuncionarios)

Função para cadastrar novos funcionários. Recebe como parâmetros o array do tipo Funcionario, onde novos funcionários são armazenados, e o ponteiro que armazena o número atual de funcionários. Este valor é incrementado a cada novo funcionário cadastrado.
Essa função não possui retorno.

4. void carregarFuncionarios(Funcionario funcionarios[], int *numFuncionarios)

Função para carregar os dados dos funcionários do arquivo Funcionarios.txt. Recebe como parâmetros o array do tipo Funcionario, onde os dados carregados são armazenados, e o ponteiro que armazena o número atual de funcionários. Este valor é atualizado com o número de funcionários carregados.
Essa função não possui retorno.

5. void cadastrarQuartos(Quarto quartos[], int *numQuartos)

Função para cadastrar novos quartos. Recebe como parâmetros o array do tipo Quarto, onde novos quartos são armazenados, e o ponteiro que armazena o número atual de quartos. Este valor é incrementado a cada novo quarto cadastrado.
Essa função não possui retorno.

6. void carregarQuartos(Quarto quartos[], int *numQuartos)

Função para carregar os dados dos quartos do arquivo Quartos.txt. Recebe como parâmetros o array do tipo Quarto, onde os dados carregados são armazenados, e o ponteiro que armazena o número atual de quartos. Este valor é atualizado com o número de quartos carregados.
Essa função não possui retorno.

7. void cadastrarEstadia(Estadia estadias[], int *numEstadias, Cliente clientes[], int numClientes, Quarto quartos[], int numQuartos)

Função para cadastrar novas estadias. Recebe como parâmetros o array do tipo Estadia, onde novas estadias são armazenadas, o ponteiro que armazena o número atual de estadias, um array do tipo Clientes, que associa um cliente à uma estadia, o número total de clientes cadastrados, um array do tipo Quarto, que associa um quarto à estadia e o número total de quartos cadastrados.

Essa função não possui retorno.

8. void carregarEstadia(Estadia estadias[], int *numEstadias, Cliente clientes[], int numClientes, Quarto quartos[], int numQuartos)

Função para carregar os dados das estadias do arquivo Estadia.txt. Recebe como parâmetros o array do tipo Estadia, onde os dados carregados são armazenados, o ponteiro que armazena o número atual de estadias, o array do tipo Cliente, que contém os dados dos clientes, o ponteiro com a quantidade de clientes, o array do tipo Quarto, que contém os dados dos quartos e o ponteiro com a quantidade de quartos cadastrados.
Essa função não possui retorno.

9. void baixaEstadia(Estadia estadias[], int *numEstadias, Quarto quartos[], int *numQuartos)

Função para dar baixa em uma estadia. Recebe como parâmetros o array do tipo Estadia, onde as estadias são armazenadas, o ponteiro que armazena o número atual de estadias, um array do tipo Quarto, onde os quartos são armazenados e um ponteiro que armazena o número atual de quartos.
Essa função não possui retorno.

10. void pesquisarClientes(Cliente clientes[], int *numClientes)

Função para pesquisar clientes cadastrados. Recebe como parâmetros o array do tipo Cliente, onde clientes são armazenados e o ponteiro que armazena o número atual de clientes.
Essa função não possui retorno.

11. void pesquisarFuncionarios(Funcionario funcionarios[], int *numFuncionarios)

Função para pesquisar funcionários cadastrados. Recebe como parâmetros o array do tipo Funcionario, onde funcionários são armazenados e o ponteiro que armazena o número atual de funcionários.
Essa função não possui retorno.

12. void totalEstadias(Estadia estadias[], int *numEstadias, Cliente clientes[], int *numClientes)

Função para calcular o total de estadias de um determinado cliente. Recebe como parâmetros o array do tipo Estadia, onde as estadias são armazenadas, o ponteiro que armazena o número atual de estadias, o array

do tipo Cliente, onde clientes são armazenados e o ponteiro que armazena o número atual de clientes.
Essa função não possui retorno.

TESTES

Casos de Teste do Software:

Funções Testadas	Entradas	Classes Válidas	Resultado Esperado	Classes Inválidas	Resultado Esperado
Cadastro de Clientes	Código: 1, Nome: “Luis Augusto”, Endereço: “7 de Abril”, Telefone: 123456789	Código, Nome, Endereço, Telefone	Cliente Cadastrado	Código Negativo	Erro! Código Inválido!
	Código: 1, Nome: “Isabella Dias”, Endereço: “Castelo Rodrigo”, Telefone: 123456789	Código Repetido	Erro! Código em Uso!	Código Semelhante	Erro! Código em Uso!
Cadastro de Funcionários	Código: 1, Nome: “Joao Silva”, Telefone: 123456789, Cargo: “Recepcionista”, Salário: 2500.00	Código, Nome, Telefone, Cargo, Salário	Funcionário Cadastrado!	Código Negativo	Erro! Código Inválido!
	Código: 1, Nome: “Maria Santos”, Telefone: 123456789, Cargo; “Garçom”, Salário: 2000.00	Código Repetido	Erro! Código em Uso!	Código Semelhante	Erro! Código em Uso!

Cadastro de Quartos	Número do Quarto: 1, Quantidade de Hóspedes: 1, Valor da Diária: 200.00, Status: “Desocupado”	Número do Quarto, Quantidade de Hóspedes, Valor da Diária, Status	Quarto Cadastrado!	Número do Quarto Negativo	Erro! Número Inválido!
	Número do Quarto: 1, Quantidade de Hóspedes: 1, Valor da Diária: 200.00, Status: “Desocupado”	Número Repetido	Erro! Quarto em Uso! Cadastro Não Realizado!	Número do Quarto Semelhante	Erro! Quarto em Uso! Cadastro Não Realizado!
Cadastro de Estadias	Código do Cliente: 1, Número do Quarto: 1, Data de Entrada: 20 07 2024, Data de Saida: 25 07 2024	Código do Cliente, Número do Quarto, Data de Entrada, Data de Saída	Estadia Cadastrada	Cliente Não Cadastrado, Quarto Não Cadastrado ou Status como “Ocupado”	Erro! Cliente Não Encontrado! Erro! Quarto Não Encontrado ou seu Status Está como Ocupado!
	Código do Cliente: 1, Número do Quarto: 2, Data de Entrada: 20 07 2024, Data de Saida: 25 07 2024	Quarto Ocupado	Erro! Quarto Não Encontrado ou seu Status Está como Ocupado!	Quarto com o Status como “Ocupado”	Erro! Quarto Não Encontrado ou seu Status Está como Ocupado!

	Código do Cliente: 6, Número do Quarto: 1, Data de Entrada: 20 07 2024, Data de Saida: 25 07 2024	Cliente Não Encontrado	Erro! Cliente Não Encontrado	Cliente Não Cadastrado	Erro! Cliente Não Encontrado!
Baixa de Estadias	Código da Estadia: 1	Estadia Encontrada	Informações da Estadia	Estadia Não Cadastrada	Estadia Não Encontrada!
	Código da Estadia: 6	Estadia Não Encontrada	Estadia Não Encontrada	Estadia Não Encontrada	Estadia Não Encontrada!
Pesquisar Clientes	Código do Cliente: 1	Código Encontrado	Informações do Cliente	Código Não Encontrado	Cliente Não Encontrado!
	Código do Cliente: 6	Código Não Encontrado	Cliente Não Encontrado	Código Não Encontrado	Cliente Não Encontrado!
	Nome: “Jorge Oliveira”	Nome Encontrado	Informações do Cliente	Nome Não Encontrado	Cliente Não Encontrado!
	Nome: “Marina Ferreira”	Nome Não Encontrado	Cliente Não Encontrado	Nome Não Encontrado	Cliente Não Encontrado!
	Opção: 1	Opção Válida	Digite o Código do Cliente a ser Pesquisado	Opção Diferente de 1 e 2	Opção Inválida
	Opção: 3	Opção Não Válida	Opção Inválida	Opção Diferente de 1 e 2	Opção Inválida
Pesquisar Funcionários	Código do Funcionário: 1	Código Encontrado	Informações do Funcionário	Código Não Encontrado	Funcionário Não Encontrado!
	Código do Funcionário: 6	Código Não	Funcionário Não Encontrado	Código Não Encontrado	Funcionário Não

		Encontrado			Encontrado!
	Nome: “Bruno Pinheiro”	Nome Encontrado	Informações do Funcionário	Nome Não Encontrado	Funcionário Não Encontrado!
	Nome: “Marina Ferreira”	Nome Não Encontrado	Funcionário Não Encontrado	Nome Não Encontrado	Funcionário Não Encontrado!
	Opção: 1	Opção Válida	Digite o Código do Funcionário a ser Pesquisado	Opção Diferente de 1 e 2	Opção Inválida
	Opção: 0	Opção Não Válida	Opção Inválida	Opção Diferente de 1 e 2	Opção Inválida
Total de Estadias	Código do Cliente: 1	Cliente Encontrado	Informações de Todas as Estadias do Cliente	Código Não Encontrado	Cliente Não Encontrado!
	Código do Cliente: 6	Cliente Não Cadastrado	Cliente Não Encontrado	Código Não Encontrado	Cliente Não Encontrado!
	Código da Estadia: 2	Estadia Cadastrada	Informações de Todas as Estadias do Cliente	Estadia Não Cadastrada	Nenhuma Estadia Encontrada!
	Código da Estadia: 6	Estadia Não Cadastrada	Nenhuma Estadia Encontrada	Estadia Não Cadastrada	Nenhuma Estadia Encontrada!
Menu de Opções	Opção: 1	Opção Válida	Cadastrar Clientes	Opção Diferente dos Valores entre 1 e 9	Opção Inválida!
	Opção: 0	Opção Não Válida	Opção Inválida!	Opção Diferente dos Valores entre 1 e 9	Opção Inválida!
	Opção: “a”	Opção Não Válida	Opção Inválida!	Opção Diferente dos Valores entre 1 e 9	Opção Inválida!

Relatório de Execução de Testes

Tabela 2 – Teste do Cadastro de Clientes

Teste 1: Cadastro de Clientes				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
Código do Cliente: Número Inteiro e Positivo.	Número entre 0 e 100. (Não Repetido)	Confirmação de Cliente Cadastrado	Código do Cliente: Número Inteiro e Negativo.	Erro! Código Inválido! Erro! Código em Uso!

Relatório de Execução de Testes		
Entradas	**Resultado**	**Aprovado?**
Código: 1	Cliente Cadastrado	Sim
Código: -1	Cliente Não Cadastrado	Sim
Código: 1	Cliente Não Cadastrado	Sim

Fonte: Elaborado pelos autores

Tabela 3 – Teste do Cadastro de Funcionários

Teste 2: Cadastro de Funcionários				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
Código do Funcionário: Número Inteiro e Positivo.	Número entre 0 e 100. (Não Repetido)	Confirmação de Funcionário Cadastrado	Código do Funcionário: Número Inteiro e Negativo.	Erro! Código Inválido! Erro! Código em Uso!

Relatório de Execução de Testes		
Entradas	**Resultado**	**Aprovado?**
Código: 1	Funcionário Cadastrado	Sim
Código: -1	Funcionário Não Cadastrado	Sim
Código: 1	Funcionário Não Cadastrado	Sim

Fonte: Elaborado pelos autores

Tabela 4 – Teste do Cadastro de Quartos

Teste 3: Cadastro de Quartos				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**

Número do Quarto: Número Inteiro e Positivo.	Número entre 0 e 200. (Não Repetido)	Confirmação de Quarto Cadastrado	Número do Quarto: Número Inteiro e Negativo.	Erro! Número Inválido! Erro! Quarto em Uso!
Relatório de Execução de Testes				
Entradas		**Resultado**		**Aprovado?**
Número: 1		Quarto Cadastrado		Sim
Número: -1		Quarto Não Cadastrado		Sim
Número: 1		Quarto Não Cadastrado		Sim

Fonte: Elaborado pelos autores

Tabela 5 – Teste do Cadastro de Estadias

Teste 4: Cadastro de Estadias				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
Número do Quarto e Código do Cliente: Ambos Previamente Cadastrados	Número do Quarto e Código do Cliente Presentes nos Arquivos	Confirmação de Estadia Cadastrada	Número do Quarto ou Código do Cliente Ausentes nos Arquivos	Erro! Quarto Não Encontrado ou seu Status Está como “Ocupado”. Erro! Cliente Não Encontrado!
Relatório de Execução de Testes				
Entradas		**Resultado**		**Aprovado?**
Código do Cliente: 1 Número do Quarto: 1		Estadia Cadastrada		Sim
Código do Cliente: 6 Número do Quarto: 1		Estadia Não Cadastrada		Sim
Código do Cliente: 1 Número do Quarto: 6		Estadia Não Cadastrada		Sim

Fonte: Elaborado pelos autores

Tabela 6 – Teste da Baixa de Estadia

Teste 5: Baixa de Estadia

Entradas	Classes Válidas	Resultado Esperado	Classes Inválidas	Resultado Esperado
Código da Estadia: Cadastrado Previamente	Código da Estadia Presente no Arquivo	Confirmação da Baixa da Estadia	Código da Estadia Não Cadastrada	Estadia Não Encontrada
Relatório de Execução de Testes				
Entradas		**Resultado**		**Aprovado?**
Código: 1		Estadia Encontrada		Sim
Código: 6		Estadia Não Encontrada		Sim

Fonte: Elaborado pelos autores

Tabela 7 – Teste da Pesquisa por Clientes

Teste 6: Pesquisa por Clientes				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
Código do Cliente e Nome do Cliente Previamente Cadastrados Opção: Número entre 1 e 2	Código e Nome do Cliente Presentes no Arquivo Opção Válida	Informações do Cliente Pesquisado	Código e Nome do Cliente Não Presentes no Arquivo Opção Diferente de 1 e 2	Cliente Não Encontrado! Opção Inválida!
Relatório de Execução de Testes				
Entradas		**Resultado**		**Aprovado?**
Código: 1		Cliente Encontrado		Sim
Código: 6		Cliente Não Encontrado		Sim
Nome: “Jorge Oliveira”		Cliente Encontrado		Sim
Nome: “Marina Ferreira”		Cliente Não Encontrado		Sim
Opção: 1		Opção Válida		Sim
Opção: 3		Opção Inválida		Sim

Fonte: Elaborado pelos autores

Tabela 8 – Teste da Pesquisa por Funcionários

Teste 7: Pesquisa por Funcionários				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
Código do Funcionário e Nome do Funcionário Previamente Cadastrados Opção: Número entre 1 e 2	Código e Nome do Funcionário Presentes no Arquivo Opção Válida	Informações do Funcionário Pesquisado	Código e Nome do Funcionário Não Presentes no Arquivo Opção Diferente de 1 e 2	Funcionário Não Encontrado! Opção Inválida!

Relatório de Execução de Testes		
Entradas	**Resultado**	**Aprovado?**
Código: 1	Funcionário Encontrado	Sim
Código: 6	Funcionário Não Encontrado	Sim
Nome: “Bruno Pinheiro”	Funcionário Encontrado	Sim
Nome: “Marina Ferreira”	Funcionário Não Encontrado	Sim
Opção: 1	Opção Válida	Sim
Opção: 0	Opção Inválida	Sim

Fonte: Elaborado pelos autores

Tabela 9 – Teste do Total de Estadias por Cliente

Teste 8: Total de Estadias por Cliente				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
Código do Cliente e Código da Estadia Previamente Cadastrados	Código do Cliente e Código da Estadia Presentes no Arquivo	Informações de Todas as Estadias do Cliente	Código do Cliente e Código da Estadia Não Presentes no Arquivo	Cliente Não Encontrado! Nenhuma Estadia Encontrada!

Relatório de Execução de Testes		
Entradas	**Resultado**	**Aprovado?**

Código do Cliente: 1	Cliente Encontrado	Sim
Código do Cliente: 6	Cliente Não Encontrado	Sim
Código da Estadia: 1	Estadia Encontrada	Sim
Código da Estadia: 0	Estadia Não Encontrada	Sim

Fonte: Elaborado pelos autores

Tabela 10 – Teste do Menu de Opções

Teste 9: Menu de Opções				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
Número Inteiro Positivo	Número Inteiro Positivo entre 1 e 9	Entrada na Função Correspondente à Opção Digitada	Número Diferente do Conjunto dos Números entre 1 e 9	Opção Inválida!

Relatório de Execução de Testes		
Entradas	**Resultado**	**Aprovado?**
Opção: 1	Função Cadastrar Clientes	Sim
Opção: 2	Função Cadastrar Funcionários	Sim
Opção: 3	Função Cadastrar Quartos	Sim
Opção: 4	Função Cadastrar Estadias	Sim
Opção: 5	Função Baixa Estadia	Sim
Opção: 6	Função Pesquisar Clientes	Sim
Opção: 7	Função Pesquisar Funcionários	Sim
Opção: 8	Função Total Estadias	Sim
Opção: 9	Sair	Sim
Opção: -1	Opção Inválida	Sim
Opção: ‘a’	Opção Inválida	Sim

Fonte: Elaborado pelos autores

Código em C:

```
/*****************************************************************************************************************************
* FILENAME : HotelBooker.c
* DESCRIPTION : Hotel management system for handling client, room, and stay records, including functions for registration,
*               search, and management of data.
* PUBLIC FUNCTIONS :
*       void cadastrarClientes(Cliente clientes[], int *numClientes)
*       void cadastrarFuncionarios(Funcionario funcionarios[], int *numFuncionarios)
*       void cadastrarQuartos(Quarto quartos[], int *numQuartos)
*       void cadastrarEstadia(Estadia estadias[], int *numEstadias, Cliente clientes[], int numClientes, Quarto quartos[], int numQuartos)
*       void baixaEstadia(Estadia estadias[], int *numEstadias, Quarto quartos[], int *numQuartos)
*       void pesquisarClientes(Cliente clientes[], int *numClientes)
*       void pesquisarFuncionarios(Funcionario funcionarios[], int *numFuncionarios)
*       void totalEstadias(Estadia estadias[], int *numEstadias, Cliente clientes[], int *numClientes)
* NOTES :
*       This program manages hotel operations through file handling and interactive user options.
*
* AUTHOR : Isabella Dias
* AUTHOR : Gustavo Viana
* START DATE : 18 Jun 24
*****************************************************************************************************************************/
```

```
//Declaração da Bibliotecas Necessárias
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
//-------------------------------------

//Estrutura Data
typedef struct{
    int dia;
    int mes;
    int ano;
}Data;
//-------------------------------------

//Estrutura Cliente
typedef struct{
    int codigoCliente;
    char nomeCliente[200];
    char endereco[200];
    char telefoneCliente[100];
}Cliente;
//-------------------------------------

//Estrutura Funcionario;
typedef struct{
    int codigoFuncionario;
    char nomeFuncionario[200];
    char telefoneFuncionario[100];
    char cargo[200];
```

```
    float salario;
}Funcionario;
//-------------------------------------

//Estrutura Quarto
typedef struct{
    int numeroQuarto;
    int quantidadeHospedes;
    float valorDiaria;
    char status[200];
}Quarto;
//-------------------------------------

//Estrutura Estadia
typedef struct{
    int codigoEstadia;
    Data dataEntrada;
    Data dataSaida;
    int quantidadeDiarias;
    Cliente codigoCliente;
    Quarto numeroQuarto;
}Estadia;
//--------------------------------------

//Declaração de Funções e Parâmetros
void cadastrarClientes(Cliente clientes[], int *numClientes);
void carregarClientes(Cliente clientes[], int *numClientes);
void cadastrarFuncionarios(Funcionario funcionarios[], int *numFuncionarios);
void carregarFuncionarios(Funcionario funcionarios[], int *numFuncionarios);
void cadastrarQuartos(Quarto quartos[], int *numQuartos);
```

```
void carregarQuartos(Quarto quartos[], int *numQuartos);

void cadastrarEstadia(Estadia estadias[], int *numEstadias, Cliente cliente[], int numClientes, Quarto quartos[], int numQuartos);

void carregarEstadias(Estadia estadias[], int *numEstadias, Cliente cliente[], int numClientes, Quarto quartos[], int numQuartos);

void baixaEstadia(Estadia estadias[], int *numEstadias, Quarto quartos[], int *numQuartos);

void pesquisarClientes(Cliente clientes[], int *numClientes);

void pesquisarFuncionarios(Funcionario funcionarios[], int *numFuncionarios);

void totalEstadias(Estadia estadias[], int *numEstadias, Cliente clientes[], int *numClientes);

//---------------------------------------


/*Cadastra Novos Clientes, Armazenando os Dados em Memória e em um Arquivo "Clientes.txt"*/

//Função Cadastrar Clientes

void cadastrarClientes(Cliente clientes[], int *numClientes) {

    Cliente novoCliente;

    int codigoExistente = 0;


    //Entrada do Código do Novo Cliente

    printf("\nDigite o Codigo do Cliente:\n");

    scanf(" %d", &novoCliente.codigoCliente);


    if (novoCliente.codigoCliente < 0) {

        printf("Erro! Codigo Invalido!\n");

        return;

    }


    //Garantir que o Código Não Está em Uso

    for (int i = 0; i < *numClientes; i++) {
```

```
    if (clientes[i].codigoCliente == novoCliente.codigoCliente) {
        codigoExistente = 1;
        break;
    }
}

if (codigoExistente) {
    printf("Erro! Codigo em Uso!\n");
    return;
}

printf("Digite o Nome do Cliente: ");
scanf(" %[^\n]", novoCliente.nomeCliente);
printf("Digite o Endereco do Cliente: ");
scanf(" %[^\n]", novoCliente.endereco);
printf("Digite o Telefone do Cliente: ");
scanf(" %[^\n]", novoCliente.telefoneCliente);

//Atualizar Clientes Cadastrados
clientes[*numClientes] = novoCliente;
(*numClientes)++;

//Armazenamento no Arquivo Clientes.txt
FILE *file = fopen("Clientes.txt", "a");
if (file == NULL) {
    printf("\nErro ao Abrir o Arquivo\n");
    return;
}

fprintf(file, "%d,", novoCliente.codigoCliente);
```

```
    fprintf(file, "%s,", novoCliente.nomeCliente);
    fprintf(file, "%s,", novoCliente.endereco);
    fprintf(file, "%s\n", novoCliente.telefoneCliente);
    fclose(file);

    printf("Cliente Cadastrado!\n");
}
//---------------------------------------

/*Carrega Dados dos Clientes a Partir do Arquivo "Clientes.txt", Incrementa um
Array de Estruturas de Clientes
e Atualiza o Contador de Clientes.*/
//Carregar Arquivo Clientes
void carregarClientes(Cliente clientes[], int *numClientes) {

    FILE *file = fopen("Clientes.txt", "r");

    if (file == NULL) {
        return;
    }

    //Variáveis Temporárias
    char linha[800];
    int codigo;
    char nome[200];
    char endereco[200];
    char telefone[100];

    // Lê o Arquivo Linha por Linha
```

```
    while (fscanf(file, "%d,%[^,],%[^,],%[^\n]\n", &codigo, nome, endereco, telefone) != EOF) {

        clientes[*numClientes].codigoCliente = codigo;
        strcpy(clientes[*numClientes].nomeCliente, nome);
        strcpy(clientes[*numClientes].endereco, endereco);
        strcpy(clientes[*numClientes].telefoneCliente, telefone);

        //Contador de Clientes
        (*numClientes)++;

    }

    //Fecha o Arquivo
    fclose(file);
}
//---------------------------------------

/*Cadastra Novos Funcionários, Armazenando os Dados em Memória e em um Arquivo "Funcionarios.txt"*/
//Função Cadastrar Funcionários
void cadastrarFuncionarios(Funcionario funcionarios[], int *numFuncionarios) {
    Funcionario novoFuncionario;
    int codigoExistente = 0;

    //Entrada do Código do Novo Funcionário
    printf("\nDigite o Codigo do Funcionario:\n");
    scanf(" %d", &novoFuncionario.codigoFuncionario);

    if (novoFuncionario.codigoFuncionario < 0) {
```

```
    printf("Erro! Codigo Invalido!\n");
    return;
}

//Garantir que o Código Não Está em Uso
for (int i = 0; i < *numFuncionarios; i++) {
    if (funcionarios[i].codigoFuncionario == novoFuncionario.codigoFuncionario) {
        codigoExistente = 1;
        break;
    }
}

if (codigoExistente) {
    printf("Erro! Codigo em Uso!\n");
    return;
}

printf("Digite o Nome do Funcionario: ");
scanf(" %[^\n]", novoFuncionario.nomeFuncionario);
printf("Digite o Telefone do Funcionario: ");
scanf(" %[^\n]", novoFuncionario.telefoneFuncionario);
printf("Digite o Cargo do Funcionario: ");
scanf(" %[^\n]", novoFuncionario.cargo);
printf("Digite o Salario do Funcionario: ");
scanf(" %f", &novoFuncionario.salario);

//Atualizar Funcionários Cadastrados
funcionarios[*numFuncionarios] = novoFuncionario;
(*numFuncionarios)++;
```

```
    //Armazenamento no Arquivo Funcionarios.txt
    FILE *file = fopen("Funcionarios.txt", "a");
    if (file == NULL) {
        printf("\nErro ao Abrir o Arquivo\n");
        return;
    }

    fprintf(file, "%d,", novoFuncionario.codigoFuncionario);
    fprintf(file, "%s,", novoFuncionario.nomeFuncionario);
    fprintf(file, "%s,", novoFuncionario.telefoneFuncionario);
    fprintf(file, "%s,", novoFuncionario.cargo);
    fprintf(file, "%.2f\n", novoFuncionario.salario);
    fclose(file);

    printf("Funcionario Cadastrado!\n");
}
//---------------------------------------

/*Carrega Dados dos Funcionários a Partir do Arquivo "Funcionarios.txt", Incrementa
um Array de Estruturas de Funcionários
e Atualiza o Contador de Funcionários.*/
//Carregar Arquivo Funcionários
void carregarFuncionarios(Funcionario funcionarios[], int *numFuncionarios) {

    FILE *file = fopen("Funcionarios.txt", "r");

    if (file == NULL) {
        return;
    }
```

```
    //Variáveis Temporárias
    int codigo;
    char nome[200];
    char telefone[100];
    char cargo[100];
    float salario;

    // Lê o Arquivo Linha por Linha
    while (fscanf(file, "%d,%[^,],%[^,],%[^,],%f\n", &codigo, nome, telefone, cargo, &salario) != EOF) {

        funcionarios[*numFuncionarios].codigoFuncionario = codigo;
        strcpy(funcionarios[*numFuncionarios].nomeFuncionario, nome);
        strcpy(funcionarios[*numFuncionarios].telefoneFuncionario, telefone);
        strcpy(funcionarios[*numFuncionarios].cargo, cargo);
        funcionarios[*numFuncionarios].salario = salario;

        //Contador de Funcionários
        (*numFuncionarios)++;
    }

    //Fecha o Arquivo
    fclose(file);
}
//---------------------------------------

/*Cadastra Novos Quartos, Armazenando os Dados em Memória e em um Arquivo "Quartos.txt"*/
//Cadastrar Quartos
```

```
void cadastrarQuartos(Quarto quartos[], int *numQuartos){
    Quarto novoQuarto;

    //Entrada do Número do Novo Quarto
    printf("\nDigite o Numero do Quarto:\n");
    scanf(" %d", &novoQuarto.numeroQuarto);

    if (novoQuarto.numeroQuarto < 0) {
        printf("Erro! Numero Invalido!\n");
        return;
    }

    //Garantir que o Número do Quarto Não Está em Uso
    for(int i = 0; i < *numQuartos; i++){
        if(quartos[i].numeroQuarto == novoQuarto.numeroQuarto){
            printf("\nErro! Quarto em Uso! Cadastro Nao Realizado!\n");
            return;
        }
    }

    printf("\nDigite a Quantidade de Hospedes: ");
    scanf(" %d", &novoQuarto.quantidadeHospedes);
    printf("\nDigite oValor da Diaria: ");
    scanf(" %f", &novoQuarto.valorDiaria);
    printf("\nDigite o Status do Quarto: (Ocupado/Desocupado): ");
    scanf(" %[^\n]", novoQuarto.status);

    //Atualizar Quartos Cadastrados
    quartos[*numQuartos] = novoQuarto;
    (*numQuartos)++;
```

```
    //Armazenamento no Arquivo Quartos.txt
    FILE *file = fopen("Quartos.txt", "a");
    if (file == NULL) {
        printf("\nErro ao Abrir o Arquivo\n");
        return;
    }
    fprintf(file, "%d,", novoQuarto.numeroQuarto);
    fprintf(file, "%d,", novoQuarto.quantidadeHospedes);
    fprintf(file, "%.2f,", novoQuarto.valorDiaria);
    fprintf(file, "%s\n", novoQuarto.status);
    fclose(file);

    printf("Quarto Cadastrado!\n");
}
//---------------------------------------

/*Carrega Dados dos Quartos a Partir do Arquivo "Quartos.txt", Incrementa um Array de Estruturas de Quartos
e Atualiza o Contador de Quartos.*/
//Carregar Arquivo Quartos
void carregarQuartos(Quarto quartos[], int *numQuartos) {

    FILE *file = fopen("Quartos.txt", "r");

    if (file == NULL) {
        return;
    }

    //Variáveis Temporárias
```

```
    int numero;
    int hospedes;
    float diaria;
    char status[200];

    // Lê o Arquivo Linha por Linha
    while (fscanf(file, "%d,%d,%f,%s[^\n]\n", &numero, &hospedes, &diaria, status) != EOF) {

        quartos[*numQuartos].numeroQuarto = numero;
        quartos[*numQuartos].quantidadeHospedes = hospedes;
        quartos[*numQuartos].valorDiaria = diaria;
        strcpy(quartos[*numQuartos].status, status);

        //Contador de Quartos
        (*numQuartos)++;
    }

    //Fecha o Arquivo
    fclose(file);
}
//---------------------------------------


/*Cadastra uma Nova Estadia Associando um Cliente a um Quarto, Registrando Datas de Entrada e Saída, Calculando a Quantidade
de Diárias e Atualizando Arquivos de Estadias e Quartos.*/
//Cadastrar Estadia
void cadastrarEstadia(Estadia estadias[], int *numEstadias, Cliente clientes[], int numClientes, Quarto quartos[], int numQuartos){
    Estadia novaEstadia;
    int codigoCliente, numeroQuarto;
```

```
    int clienteEncontrado = 0, quartoEncontrado = 0;

    //Entrada do Código do Cliente
    printf("\nDigite o Codigo do Cliente:\n");
    scanf("%d", &codigoCliente);

    //Garantir que o Cliente Existe
    for(int i = 0; i < numClientes; i++){
        if(clientes[i].codigoCliente == codigoCliente){
            novaEstadia.codigoCliente = clientes[i];
            clienteEncontrado = 1;
            break;
        }
    }

    if(!clienteEncontrado){
        printf("\nErro! Cliente Nao Encontrado!\n");
        return;
    }

    printf("\nDigite o Numero do Quarto: ");
    scanf(" %d", &numeroQuarto);

    //Garantir que o Quarto Existe e que seu Status Está Como Desocupado
    for(int i = 0; i < numQuartos; i++){
        if(quartos[i].numeroQuarto == numeroQuarto && strcmp(quartos[i].status,
"Desocupado") == 0){
            novaEstadia.numeroQuarto = quartos[i];
            quartoEncontrado = 1;
            strcpy(quartos[i].status, "Ocupado");
```

```
            break;
        }
    }

    if(!quartoEncontrado){
        printf("\nErro! Quarto Nao Encontrado ou seu Status Esta como Ocupado!\n");
        return;
    }

    printf("\nDigite a Data de Entrada (dd mm aaaa): \n");
    scanf("%d %d %d", &novaEstadia.dataEntrada.dia,
&novaEstadia.dataEntrada.mes, &novaEstadia.dataEntrada.ano);

    printf("\nDigite a Data de Saida (dd mm aaaa): \n");
    scanf("%d %d %d", &novaEstadia.dataSaida.dia, &novaEstadia.dataSaida.mes,
&novaEstadia.dataSaida.ano);

    //Cálculo da Quantidade de Diárias
    novaEstadia.quantidadeDiarias = (novaEstadia.dataSaida.dia -
novaEstadia.dataEntrada.dia) + (novaEstadia.dataSaida.mes -
novaEstadia.dataEntrada.mes) * 30 + (novaEstadia.dataSaida.ano -
novaEstadia.dataEntrada.ano) * 365;

    //Atualizar Estadias Cadastradas
    novaEstadia.codigoEstadia = *numEstadias;
    estadias[*numEstadias] = novaEstadia;
    (*numEstadias)++;
    novaEstadia.codigoEstadia = *numEstadias;

    //Armazenamento no Arquivo Estadia.txt
    FILE *file = fopen("Estadia.txt", "a");
```

```
if (file == NULL) {
    printf("\nErro ao Abrir o Arquivo\n");
    return;
}
fprintf(file, " %d,", novaEstadia.codigoEstadia);
fprintf(file, " %d,", novaEstadia.codigoCliente.codigoCliente);
fprintf(file, " %d,", novaEstadia.numeroQuarto.numeroQuarto);
fprintf(file, " %d,", novaEstadia.dataEntrada.dia);
fprintf(file, " %d,", novaEstadia.dataEntrada.mes);
fprintf(file, " %d,", novaEstadia.dataEntrada.ano);
fprintf(file, " %d,", novaEstadia.dataSaida.dia);
fprintf(file, " %d,", novaEstadia.dataSaida.mes);
fprintf(file, " %d\n", novaEstadia.dataSaida.ano);
fclose(file);

//Atualizar o Status do Quarto no Arquivo Quartos.txt
file = fopen("Quartos.txt", "w");
if (file == NULL) {
    printf("\nErro ao Abrir o Arquivo\n");
    return;
}
for (int i = 0; i < numQuartos; i++) {
    fprintf(file, "%d,%d,%.2f,%s\n",
            quartos[i].numeroQuarto,
            quartos[i].quantidadeHospedes,
            quartos[i].valorDiaria,
            quartos[i].status);
}
fclose(file);
```

```
    printf("Estadia Cadastrada!\n");
}
//---------------------------------------

/*Carrega Informações de Estadias Registradas a Partir de um Arquivo "Estadia.txt", Associando Clientes e Quartos às Estadias,
Calculando a Quantidade de Diárias com Base nas Datas de Entrada e Saída.*/
//Carregar Arquivo Estadia
void carregarEstadias(Estadia estadias[], int *numEstadias, Cliente clientes[], int numClientes, Quarto quartos[], int numQuartos) {

    FILE *file = fopen("Estadia.txt", "r");

    if (file == NULL) {
        return;
    }

    //Variáveis Temporárias
    int codigoEstadia, codigoCliente, numeroQuarto;
    int diaEntrada, mesEntrada, anoEntrada;
    int diaSaida, mesSaida, anoSaida;
    int clienteIndex, quartoIndex;

    // Lê o Arquivo Linha por Linha
    while (fscanf(file, "%d,%d,%d,%d,%d,%d,%d,%d,%d\n", &codigoEstadia, &codigoCliente, &numeroQuarto, &diaEntrada, &mesEntrada, &anoEntrada, &diaSaida, &mesSaida, &anoSaida) != EOF) {

        clienteIndex = -1;
        quartoIndex = -1;
```

```
//Encontrar Cliente pelo Código
for(int i = 0; i < numClientes; i++){
    if(clientes[i].codigoCliente == codigoCliente){
        clienteIndex = i;
        break;
    }
}

//Encontrar Quarto pelo Número
for(int i = 0; i < numeroQuarto; i++){
    if(quartos[i].numeroQuarto == numeroQuarto){
        quartoIndex = i;
        break;
    }
}

//Garante que o Quarto e o Cliente Foram Encontrados
if(clienteIndex == -1 || quartoIndex == -1){
    continue;
}

estadias[*numEstadias].codigoEstadia = codigoEstadia;
estadias[*numEstadias].codigoCliente = clientes[clienteIndex];
estadias[*numEstadias].numeroQuarto = quartos[quartoIndex];
estadias[*numEstadias].dataEntrada.dia = diaEntrada;
estadias[*numEstadias].dataEntrada.mes = mesEntrada;
estadias[*numEstadias].dataEntrada.ano = anoEntrada;
estadias[*numEstadias].dataSaida.dia = diaSaida;
estadias[*numEstadias].dataSaida.mes = mesSaida;
```

```
        estadias[*numEstadias].dataSaida.ano = anoSaida;

        //Cálculo da Quantidade de Diárias
        estadias[*numEstadias].quantidadeDiarias = (diaSaida - diaEntrada) +
(mesSaida - mesEntrada) * 30 + (anoSaida - anoEntrada) * 365;

        //Contador de Estadias
        (*numEstadias)++;
    }

    //Fecha o Arquivo
    fclose(file);

}
//---------------------------------------

//Teste: Printar Estadia
void printarEstadia(Estadia estadias[], int *numEstadias){
    for(int i = 0; i < *numEstadias; i++){
        printf("Estadia ID: %d\n", estadias[i].codigoEstadia);
    }
}
//---------------------------------------

/*Realiza o Encerramento de uma Estadia, Atualiza o Status do Quarto para
"Desocupado", Calcula o Valor Total da Estadia e Remove
a Estadia do Registro.*/
//Baixa em Alguma Estadia
void baixaEstadia(Estadia estadias[], int *numEstadias, Quarto quartos[], int
*numQuartos) {
```

```
    int codigoEstadia = 0;

    printf("\nDigite o Codigo da Estadia: \n");
    scanf(" %d", &codigoEstadia);

    //Procurando Estadia pelo Código
    int estadiaIndex = -1;
    for (int i = 0; i < *numEstadias; i++) {
        if (estadias[i].codigoEstadia == codigoEstadia) {
            estadiaIndex = i;
            break;
        }
    }
    if (estadiaIndex == -1) {
        printf("\nEstadia Nao Encontrada!\n");
        return;
    }

    //Imprimir as Informações da Estadia
    Estadia *estadia = &estadias[estadiaIndex];
    printf("\nEstadia Encontrada:\n");
    printf("Codigo Estadia: %d\n", estadia->codigoEstadia);
    printf("Codigo Cliente: %d\n", estadia->codigoCliente.codigoCliente);
    printf("Numero Quarto: %d\n", estadia->numeroQuarto.numeroQuarto);
    printf("Data Entrada: %d/%d/%d\n", estadia->dataEntrada.dia, estadia-
>dataEntrada.mes, estadia->dataEntrada.ano);
    printf("Data Saida: %d/%d/%d\n", estadia->dataSaida.dia, estadia-
>dataSaida.mes, estadia->dataSaida.ano);
    printf("Quantidade Diarias: %d\n", estadia->quantidadeDiarias);
```

```
    //Calculo do Valor Total da Estadia do Cliente
    float valorTotal = estadia->quantidadeDiarias * estadia-
>numeroQuarto.valorDiaria;
    printf("\nValor Total da Estadia: R$ %.2f\n", valorTotal);

    //Atualizar o Status do Quarto para "Desocupado"
    int quartoIndex = -1;
    for (int i = 0; i < *numQuartos; i++) {
        if (quartos[i].numeroQuarto == estadia->numeroQuarto.numeroQuarto) {
            quartoIndex = i;
            break;
        }
    }

    if (quartoIndex != -1) {
        strcpy(quartos[quartoIndex].status, "Desocupado");
    }

    //printarEstadia(estadias, numEstadias); (Teste Antes da Remoção)
    //Remover a Estadia do Array
    for (int i = estadiaIndex; i < *numEstadias; i++) {
        estadias[i] = estadias[i + 1];
    }
    (*numEstadias)--;
    //printarEstadia(estadias, numEstadias); (Teste Após a Remoção)


    //Atualizar o Arquivo Estadia.txt
    FILE *file = fopen("Estadia.txt", "w");
    if (file == NULL) {
```

```
    printf("\nErro ao Abrir o Arquivo\n");
    return;
}

for (int i = 0; i < *numEstadias; i++) {
    fprintf(file, "%d,%d,%d,%d,%d,%d,%d,%d,%d\n",
            estadias[i].codigoEstadia,
            estadias[i].codigoCliente.codigoCliente,
            estadias[i].numeroQuarto.numeroQuarto,
            estadias[i].dataEntrada.dia,
            estadias[i].dataEntrada.mes,
            estadias[i].dataEntrada.ano,
            estadias[i].dataSaida.dia,
            estadias[i].dataSaida.mes,
            estadias[i].dataSaida.ano);
}
fclose(file);

//Atualizar o Arquivo Quartos.txt
file = fopen("Quartos.txt", "w");
if (file == NULL) {
    printf("\nErro ao Abrir o Arquivo\n");
    return;
}

for (int i = 0; i < *numQuartos; i++) {
    fprintf(file, "%d,%d,%.2f,%s\n",
            quartos[i].numeroQuarto,
            quartos[i].quantidadeHospedes,
            quartos[i].valorDiaria,
```

```
            quartos[i].status);
    }
    fclose(file);

    printf("\nQuarto Desocupado!\n");
}
//---------------------------------------

/*Pesquisa Clientes Cadastrados pelo Código ou Nome, Exibindo Suas Informações se Encontrados.*/
//Pesquisar Clientes
void pesquisarClientes(Cliente clientes[], int *numClientes) {
    int opcao;
    printf("Deseja Pesquisar Cliente:\n");
    printf("\n1 - Por Codigo");
    printf("\n2 - Por Nome\nOp: ");
    scanf("%d", &opcao);

    if (opcao == 1) {
        int codigo;
        printf("Digite o Codigo do Cliente que Deseja Pesquisar:\n");
        scanf(" %d", &codigo);

        for (int i = 0; i < *numClientes; i++) {
            if (clientes[i].codigoCliente == codigo) {
                printf("Cliente Encontrado:\n");
                printf("Codigo: %d\n", clientes[i].codigoCliente);
                printf("Nome: %s\n", clientes[i].nomeCliente);
                printf("Endereco: %s\n", clientes[i].endereco);
                printf("Telefone: %s\n", clientes[i].telefoneCliente);
```

```
            return;
        }
    }
    printf("Cliente Nao Encontrado!\n");
} else if (opcao == 2) {
    char nome[200];
    printf("Digite o Nome do Cliente a ser Pesquisado: ");
    scanf(" %[^\n]", nome);

    for (int i = 0; i < *numClientes; i++) {
        if (strcmp(clientes[i].nomeCliente, nome) == 0) {
            printf("Cliente Encontrado:\n");
            printf("Codigo: %d\n", clientes[i].codigoCliente);
            printf("Nome: %s\n", clientes[i].nomeCliente);
            printf("Endereco: %s\n", clientes[i].endereco);
            printf("Telefone: %s\n", clientes[i].telefoneCliente);
            return;
        }
    }
    printf("Cliente Nao Encontrado.\n");
} else {
    printf("Opcao Invalida!\n");
}
}
//---------------------------------------

/*Pesquisa Funcionários Cadastrados pelo Código ou Nome, Exibindo Suas Informações se Encontrados.*/
//Pesquisar Funcionários
void pesquisarFuncionarios(Funcionario funcionarios[], int *numFuncionarios) {
```

```
int opcao;
printf("Deseja Pesquisar Funcionario:\n");
printf("\n1 - Por Codigo");
printf("\n2 - Por Nome\nOp: ");
scanf(" %d", &opcao);

if (opcao == 1) {
    int codigo;
    printf("Digite o Codigo do Funcionario que Deseja Pesquisar:\n");
    scanf(" %d", &codigo);

    for (int i = 0; i < *numFuncionarios; i++) {
        if (funcionarios[i].codigoFuncionario == codigo) {
            printf("Funcionario Encontrado:\n");
            printf("Codigo: %d\n", funcionarios[i].codigoFuncionario);
            printf("Nome: %s\n", funcionarios[i].nomeFuncionario);
            printf("Telefone: %s\n", funcionarios[i].telefoneFuncionario);
            printf("Cargo: %s\n", funcionarios[i].cargo);
            printf("Salario: %.2f\n", funcionarios[i].salario);
            return;
        }
    }
    printf("Funcionario Nao Encontrado!\n");
} else if (opcao == 2) {
    char nome[200];
    printf("Digite o Nome do Funcionario a ser Pesquisado: ");
    scanf(" %[^\n]", nome);

    for (int i = 0; i < *numFuncionarios; i++) {
        if (strcmp(funcionarios[i].nomeFuncionario, nome) == 0) {
```

```
            printf("Funcionario Encontrado:\n");
            printf("Codigo: %d\n", funcionarios[i].codigoFuncionario);
            printf("Nome: %s\n", funcionarios[i].nomeFuncionario);
            printf("Telefone: %s\n", funcionarios[i].telefoneFuncionario);
            printf("Cargo: %s\n", funcionarios[i].cargo);
            printf("Salario: %.2f\n", funcionarios[i].salario);
            return;
        }
    }
    printf("Funcionario Nao Encontrado.\n");
  } else {
    printf("Opcao Invalida!\n");
  }
}
//----------------------------------------

/*Calcula e Exibi Todas as Estadias Registradas para um Cliente Específico, Identificado pelo seu Código.*/
//Total de Estadias de um Cliente
void totalEstadias(Estadia estadias[], int *numEstadias, Cliente clientes[], int *numClientes){
  int codigoCliente = 0;

  printf("\nDigite o Codigo do Cliente: \n");
  scanf(" %d", &codigoCliente);

  //Procurando Cliente pelo Código
  int clienteIndex = -1;
  for(int i = 0; i < *numClientes; i++){
    if(clientes[i].codigoCliente == codigoCliente){
```

```
            clienteIndex = i;
            break;
        }
    }
    if(clienteIndex == -1){
        printf("\nCliente Nao Encontrado!\n");
        return;
    }

    printf("\nEstadias do Cliente %s: \n", clientes[clienteIndex].nomeCliente);
    int estadiasEncontradas = 0;
    for(int i = 0; i < *numEstadias; i++){
        if(estadias[i].codigoCliente.codigoCliente == codigoCliente){
            printf("\nCodigo da Estadia: %d\n", estadias[i].codigoEstadia);
            printf("\nData da Entrada: %02d/%02d/%04d\n", estadias[i].dataEntrada.dia,
estadias[i].dataEntrada.mes, estadias[i].dataEntrada.ano);
            printf("\nData da Saida: %02d/%02d/%04d\n", estadias[i].dataSaida.dia,
estadias[i].dataSaida.mes, estadias[i].dataSaida.ano);

            printf("\nQuantidade de Diarias: %d\n", estadias[i].quantidadeDiarias);
            printf("\nNumero do Quarto: %d\n",
estadias[i].numeroQuarto.numeroQuarto);
            printf("\n");

            estadiasEncontradas++;
        }
    }
    if(estadiasEncontradas == 0){
        printf("\nNenhuma Estadia Encontrada!\n");
    }
}
```

```
//---------------------------------------


/*Oferece Opções para Registrar Novas Entradas, Realizar Baixas em Estadias e
Buscar Informações de Clientes e Funcionários.

Utiliza Arquivos para Salvar Dados e Arrays para Gerenciar Informações.*/

//Função Principal

int main() {

    int opcao;

    Cliente clientes[100];

    Quarto quartos[200];

    Funcionario funcionarios[100];

    Estadia estadias[100];

    int numClientes = 0, numQuartos = 0, numFuncionarios = 0, numEstadias = 0;


    //Carregar o Arquivo Clientes

    carregarClientes(clientes, &numClientes);


    //Carregar o Arquivo Funcionarios

    carregarFuncionarios(funcionarios, &numFuncionarios);


    //Carregar o Arquivo Quartos

    carregarQuartos(quartos, &numQuartos);


    //Carregar o Arquivo Estadia

    carregarEstadias(estadias, &numEstadias, clientes, numClientes, quartos,
numQuartos);


    //printarEstadia(estadias, &numEstadias);


    do {
```

```
        printf("\n--------------------Hotel Descanso Garantido--------------------\n");
        printf("\nSistema HotelBooker, O que Voce Deseja?\n");
        printf("1 - Cadastrar Cliente\n");
        printf("2 - Cadastrar Funcionario\n");
        printf("3 - Cadastrar Quarto\n");
        printf("4 - Cadastrar Estadia\n");
        printf("5 - Dar Baixa em Alguma Estadia\n");
        printf("6 - Pesquisar por Cliente\n");
        printf("7 - Pesquisar por Funcionario\n");
        printf("8 - Total de Estadias por Cliente\n");
        printf("9 - Sair\n\nOp: ");
        scanf(" %d", &opcao);

        switch (opcao) {
            case 1:
                cadastrarClientes(clientes, &numClientes);
                break;
            case 2:
                cadastrarFuncionarios(funcionarios, &numFuncionarios);
                break;
            case 3:
                cadastrarQuartos(quartos, &numQuartos);
                break;
            case 4:
                cadastrarEstadia(estadias, &numEstadias, clientes, numClientes, quartos,
numQuartos);
                break;
            case 5:
                baixaEstadia(estadias, &numEstadias, quartos, &numQuartos);
                break;
```

```
            case 6:
                pesquisarClientes(clientes, &numClientes);
                break;
            case 7:
                pesquisarFuncionarios(funcionarios, &numFuncionarios);
                break;
            case 8:
                totalEstadias(estadias, &numEstadias, clientes, &numClientes);
                break;
            case 9:
                printf("\nEncerrando o Programa\n");
                break;
            default:
                printf("\nOpcao invalida!\n");

                //Limpar Buffer
                char buffer[200];
                scanf(" %[^\n]", buffer);
        }
    } while (opcao != 9);

    return 0;
}
```