

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**  
**INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA**  
**UNIDADE EDUCACIONAL PRAÇA DA LIBERDADE**  
**Bacharelado em Engenharia de software**

**Nome:** Sofia Vasconcelos Moreira e Silva

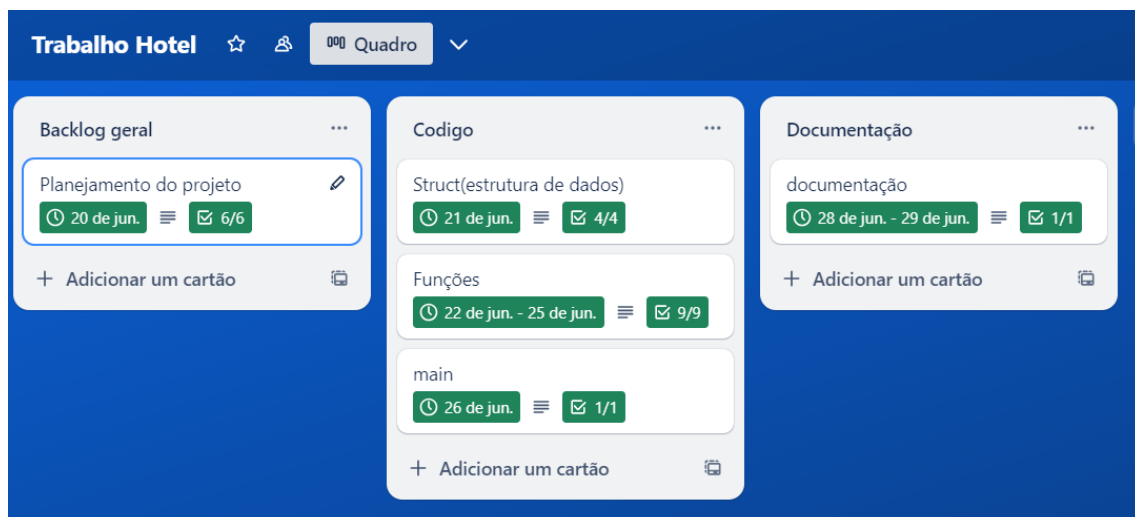
**Nome do sistema:** Hotel Descanso Garantido

### **Apresentação:**

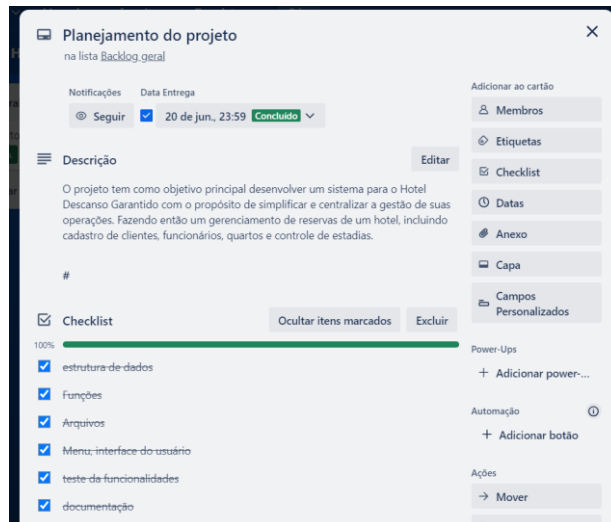
O projeto visa desenvolver um sistema para o Hotel Descanso Garantido, com o objetivo de simplificar e centralizar a gestão. Anteriormente, problemas como duplicidade de reservas e falta de organização eram frequentes. O novo sistema implementado então busca ajudar na administração de operações diárias de um hotel, permitindo o cadastro de clientes e funcionários, além de facilitar a reserva de quartos e o registro de estadias. Além disso, o acesso rápido e atualizado às informações permitirá uma gestão mais ágil. Oferecendo funcionalidades para listar as reservas, verificar o status dos quartos, apagar reservas quando necessário, e ainda permite a gravação e leitura de dados em arquivo para manter um registro organizado das informações do hotel.

### **Backlog do produto:**

<https://trello.com/b/dhpaAzhb/trabalho-hotel>



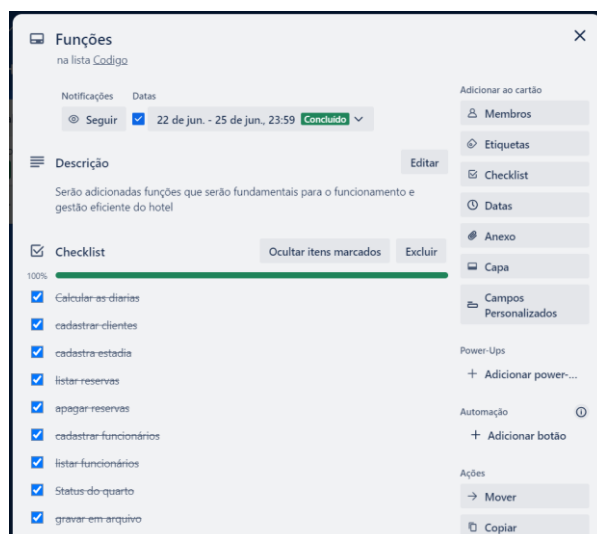
**Fonte: Elaborado pelo autor(a) Sofia Vasconcelos**



Fonte: Elaborado pelo autor(a) Sofia Vasconcelos

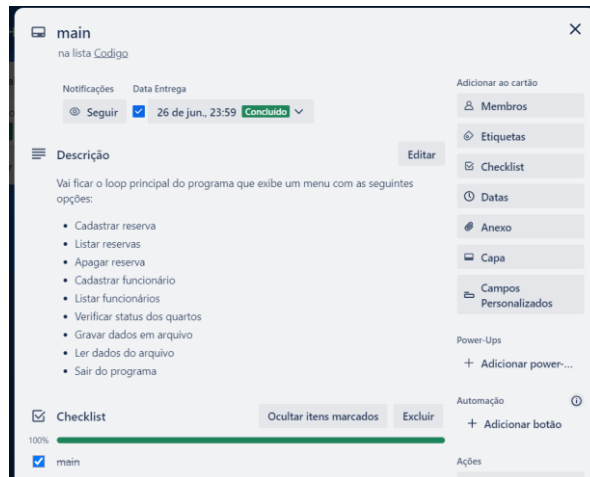


Fonte: Elaborado pelo autor(a) Sofia Vasconcelos

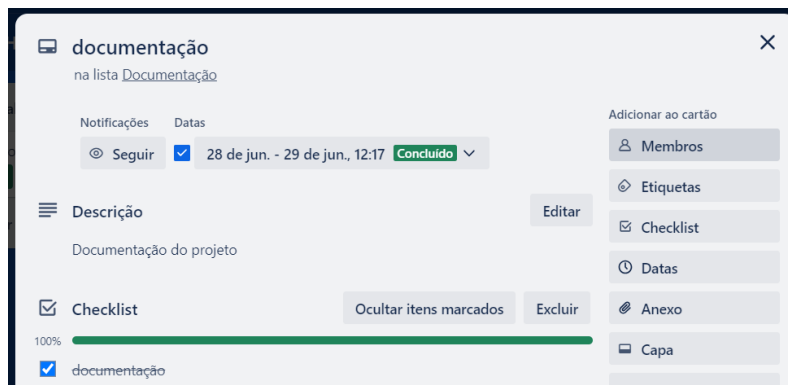


Fonte: Elaborado pelo autor(a) Sofia Vasconcelos

Fonte: Elaborado pelo autor(a) Sofia Vasconcelos



Fonte: Elaborado pelo autor(a) Sofia Vasconcelos



Fonte: Elaborado pelo autor(a) Sofia Vasconcelos

## Lista de assinaturas das funções e parâmetros:

As funções e parâmetros utilizados no programa foram:

### 1. `int calcularDiarias (char dataEntrada[], char dataSaida[]):`

Função para calcular o número de diárias entre duas datas fornecidas como strings no formato "dia/mês/ano" usando a função `difftime`. Essa foi usada para calcular a duração de estadias. Usando da `dataEntrada[]`, uma string representando a data de entrada e uma `dataSaida[]`, uma string representando a data de saída. Dando como retorno o número de dias entre as datas de entrada e saída (diárias).

### 2. `void cadastrarCliente(struct cliente clientes[], int *num_clientes):`

Função permite cadastrar um novo cliente para uma reserva. Solicitando dados do cliente como nome, telefone e endereço, além disso esse gera um código aleatório usando da função `rand ()`. O `clientes[]` é uma array de estruturas do tipo `cliente`, onde cada elemento armazena

informações como nome, telefone, endereço e código do cliente. E num\_clientes é um ponteiro para inteiro que indica o número atual de clientes cadastrados no sistema.

**3. void cadastrarEstadia(struct estadia estadias[], struct cliente clientes[], struct quarto quartos[], int \*num\_estadias, int num\_quartos):**

Função para cadastrar uma estadia, associando um cliente a um quarto disponível durante um período específico interagindo com dados de clientes, quartos e estadias. Os Parâmetros desse são, estadias[] esse é um array de estruturas estadia, onde cada elemento armazena informações como datas de entrada e saída, número do quarto, número de cliente e número de diárias, clientes[] que é um array de estruturas cliente, contendo informações sobre os clientes registrados no sistema, quartos[] é um array de estruturas quarto, que descreve os diferentes quartos disponíveis, incluindo capacidade de hóspedes, valor da diária e estado de ocupação, num\_estadias é um ponteiro para inteiro que mantém o número atual de estadias registradas no sistema. Ele é atualizado após cada novo cadastro de estadia e num\_quartos um inteiro que representa o número total de quartos disponíveis no sistema.

**4. void listarReservas(struct cliente clientes[], struct estadia estadias[], struct quarto quartos[], int num\_estadias):**

Função que lista todas as reservas cadastradas, mostrando informações dos clientes, estadias e quartos associados. Os parâmetros desse são clientes[] um array de estruturas cliente, que contém informações detalhadas sobre os clientes cadastrados no sistema, estadias[] que é um array de estruturas estadia, onde cada elemento armazena informações específicas de cada reserva, como código do cliente, datas de entrada e saída, número do quarto e número de diárias, quartos[] este é um array de estruturas quarto, descrevendo cada quarto disponível, incluindo capacidade de hóspedes e valor da diária, num\_estadias um inteiro que representa o número total de estadias (ou reservas) registradas no sistema.

**5. void apagarReserva(struct cliente clientes[], struct estadia estadias[], struct quarto quartos[], int \*num\_estadias, int num\_quartos):**

Função que permite apagar uma reserva existente, liberando o quarto associado e removendo a estadia da lista de reservas. Os parâmetros desse são clientes[] um array de estruturas cliente, contendo informações detalhadas sobre os clientes registrados no sistema, estadias[] que é um array de estruturas estadia, onde cada elemento armazena informações específicas de cada reserva, como datas de entrada e saída, número do quarto, número de diárias e código do cliente, quartos[] um array de estruturas quarto, que descreve cada quarto disponível no sistema, incluindo capacidade de hóspedes e estado de ocupação, num\_estadias é um ponteiro para inteiro que representa o número atual de estadias (ou reservas) registradas no sistema. Este valor é atualizado após a exclusão de uma reserva e num\_quartos um inteiro que indica o número total de quartos disponíveis no sistema.

**6. void cadastrarFuncionario(struct funcionario funcionarios[], int \*num\_funcionarios):**

Esta função permite cadastrar um novo funcionário do hotel, essa permite inserir informações como nome, telefone, cargo e salário. Além disso, gera um código aleatório para o funcionário cadastrado. Seus parâmetros são funcionarios[] um array de estruturas funcionario, que armazena informações detalhadas sobre os funcionários cadastrados no sistema e

num\_funcionarios um ponteiro para inteiro que indica o número atual de funcionários cadastrados no sistema. Após o cadastro de um novo funcionário, este valor é incrementado para refletir o novo número total de funcionários.

**7. void listarFuncionarios(struct funcionario funcionarios[], int num\_funcionarios):**

Função que lista todos os funcionários cadastrados no hotel, mostrando informações como nome, telefone, cargo, salário e código. Os parâmetros desse são funcionarios[] um array de estruturas funcionario, contendo informações detalhadas sobre os funcionários cadastrados no sistema e num\_funcionarios um inteiro que indica o número total de funcionários cadastrados no sistema. Esse valor determina quantas iterações o loop for fará para listar todos os funcionários.

**8. void listarStatusQuartos(struct quarto quartos[], int num\_quartos):**

Esta função lista o status de todos os quartos do hotel, mostrando detalhes como número do quarto, capacidade de hóspedes, valor da diária e se o quarto está ocupado ou desocupado. Seus parâmetros são quartos[] é um array de estruturas quarto, onde cada elemento armazena informações detalhadas sobre um quarto específico, como número, capacidade de hóspedes, valor da diária e estado de ocupação e num\_quartos um inteiro que indica o número total de quartos cadastrados no sistema. Este valor determina quantas vezes o loop for será executado para listar todos os quartos.

**9. Void gravarEstadiasClientesFuncionariosEmArquivo(struct estadia estadias[], int totalEstadias, struct cliente clientes[], int totalClientes, struct funcionario funcionarios[], int totalFuncionarios):**

Grava em um arquivo de texto (estadias.txt) as informações de estadias, clientes e funcionários cadastrados. Os parâmetros são estadias[] esse é um array de estruturas estadia, contendo informações sobre as estadias cadastradas, totalEstadias um inteiro que indica o número total de estadias cadastradas, clientes[] que é um array de estruturas cliente, contendo informações sobre os clientes cadastrados, totalClientes é um inteiro que indica o número total de clientes cadastrados, funcionarios[] um array de estruturas funcionario, contendo informações sobre os funcionários cadastrados e totalFuncionarios um inteiro que indica o número total de funcionários cadastrados.

**10. void lerEImprimirArquivoEstadias():**

Lê e imprime na tela as informações gravadas no arquivo estadias.txt. Esta função não tem parâmetros aparentes, essa lida diretamente com o arquivo "estadias.txt" definido dentro da função.

## **Testes:**

**Casos de teste do software:**

<b>Entradas</b>	<b>Classes validas</b>	<b>Resultado esperado</b>	<b>Classes invalidas</b>	<b>Resultado esperado</b>
Fazer uma reserva, com nome, telefone, endereço, insira dados da estadia, data de entrada e saída e quantidade de hospedes	Cliente e estadia cadastrada com sucesso.	Salva a reserva, gera um código para o cliente, calcula o número de diárias e associa o cliente a um quarto baseando no número de hospedes	Deixar um campo em branco	Não conseguira continuar para o preenchimento da próxima informação
Apagar a reserva	Reserva apagada com sucesso	Pergunta o número da estadia que deseja apagar e exclui o número selecionado	Se não tiver estadias cadastradas	Não há reservas cadastrads
Cadastrar funcionario, com o nome, telefone, cargo e salario	Funcionario cadastrado com sucesso	Funcionario cadastrado, gera um código para o funcionario	Deixar um campo em branco	Não conseguira continuar para o preenchimento da próxima informação
Verificar o status dos quartos	Mostra informações do quarto	O número do quarto, quantidade de hospedes máximo, valor da diária e se esta desocupado ou ocupado	Cadastrar uma estadia no mesmo quarto	O sistema deve encaminhar apenas para quartos vazios
Gravar dados em arquivo estadias.txt		Dados cadastrados com sucesso		
Ler e imprimir dados do arquivo estadias.txt		Mostra na tela as estadias, cliente e funcionarios que foram cadastrados e salvos no arquivo		
Sair	Fechando o programa...	Encerra o programa		
Menu: Mostra as opções do	Da opção de 1 a 9	Executa a função do	Digitar números menores que 1 e maiores que 9	Dizer que a opção é invalida e pedir para

que o sistema faz		número digitado		escolher novamente
-------------------	--	-----------------	--	--------------------

### Relatório de execução de testes:

Entradas	Resultado	Aprovado?
Valor: 1	Fazer uma reserva	Sim
Valor: 2	Ver reservas cadastradas	Sim
Valor: 3	Apagar reserva	Sim
Valor: 4	Cadastrar funcionario	Sim
Valor: 5	Ver funcionarios cadastrados	Sim
Valor: 6	Verificar status do quarto	Sim
Valor: 7	Gravar dados em arquivo estadias.txt	Sim
Valor: 8	Ler e imprimir dados do arquivo estadias.txt	Sim
Valor: 9	Sair	Sim

### Link do repositório do GitHub:

<https://github.com/sofiavasconcelosms/Hotel-descanso-garantido>

### Codigo:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <time.h>
```

```
struct cliente {
```

```
    char nome[70];
```

```
    char endereco[100];
```

```
    char telefone[20];
```

```
    int codigo;
```

```
};
```

```
struct funcionario {
```

```
char nome[70];  
char telefone[30];  
char cargo[50];  
float salario;  
int codigo;  
};
```

```
struct estadia {  
    char dataEntrada[70];  
    char dataSaida[30];  
    int Diarias;  
    int nQuarto;  
    int codigo;  
};
```

```
struct quarto {  
    int Nquarto;  
    int qtdHospedes;  
    float valorDiaria;  
    char status[30];  
    int ocupado;  
    int tipo;  
};
```

```
// Função para calcular o número de diárias entre duas datas
```

```
int calcularDiarias(char dataEntrada[], char dataSaida[]) {
```

```
    struct tm tmEntrada = {0};
```

```
    struct tm tmSaida = {0};
```

```
    time_t timeEntrada, timeSaida;
```

```
    // Converter as strings de data para estruturas tm
```



```

    strtotime(dataEntrada, "%d/%m/%Y", &tmEntrada);
    strtotime(dataSaida, "%d/%m/%Y", &tmSaida);

    // Converter tm para time_t
    timeEntrada = mktime(&tmEntrada);
    timeSaida = mktime(&tmSaida);

    // Calcular a diferença de dias
    double diferenca = difftime(timeSaida, timeEntrada);
    int diarias = diferenca / (60 * 60 * 24); // Converter segundos para dias

    return diarias;
}

// Função para cadastrar um cliente
void cadastrarCliente(struct cliente clientes[], int *num_clientes) {
    printf("\nInsira os dados do cliente:\n");
    printf("Nome: ");
    scanf("%s", clientes[*num_clientes].nome);
    printf("Telefone: ");
    scanf("%s", clientes[*num_clientes].telefone);
    printf("Endereço: ");
    scanf("%s", clientes[*num_clientes].endereco);

    clientes[*num_clientes].codigo = rand() % (999999999 - 1000 + 1) + 1000;

    printf("\nCliente cadastrado com sucesso.\n");
    (*num_clientes)++;
}

// Função para cadastrar uma estadia

```

```

void cadastrarEstadia(struct estadia estadias[], struct cliente clientes[],
                     struct quarto quartos[], int *num_estadias,
                     int num_quartos) {
    printf("\nAgora insira os dados para a estadia:\n");
    printf("Data de entrada: ");
    scanf("%[^\\n]s", estadias[*num_estadias].dataEntrada);
    printf("Data de saída: ");
    scanf("%[^\\n]s", estadias[*num_estadias].dataSaida);
    estadias[*num_estadias].Diarias = calcularDiarias(
        estadias[*num_estadias].dataEntrada, estadias[*num_estadias].dataSaida);

    int hospedes_desejado;
    do {
        printf("Quantidade de hóspedes: ");
        scanf("%d", &hospedes_desejado);

        if (hospedes_desejado <= 0) {
            printf("Quantidade inválida. Insira um número maior que zero.\n");
        } else {
            break; // Saia do loop se a quantidade for válida
        }
    } while (1);

    int numero_quarto = -1;

    // Busca o primeiro quarto disponível com capacidade suficiente
    for (int i = 0; i < num_quartos; i++) {
        if (!quartos[i].ocupado && quartos[i].qtdHospedes >= hospedes_desejado) {
            numero_quarto = quartos[i].Nquarto;
            quartos[i].ocupado = 1; // Marca o quarto como ocupado
            break;
        }
    }
}

```

```

    }
}

if (numero_quarto == -1) {
    printf("Não há quartos disponíveis com capacidade para %d hóspede(s).\n",
        hospedes_desejado);
    return;
}

estadias[*num_estadias].nQuarto = numero_quarto;
estadias[*num_estadias].codigo = clientes[*num_estadias].codigo;

// Captura dados do quarto
printf("Valor da diária: %.2f\n", quartos[numero_quarto - 1].valorDiaria);

printf("\nEstadia cadastrada com sucesso.\n");

(*num_estadias)++;
}

// Função para listar reservas cadastradas
void listarReservas(struct cliente clientes[], struct estadia estadias[],
    struct quarto quartos[], int num_estadias) {
    printf("\n----- Reservas Cadastradas ----- \n");
    for (int i = 0; i < num_estadias; i++) {
        printf("Reserva %d:\n", i + 1);
        printf("\nCódigo do cliente: %d\n", estadias[i].codigo);
        printf("Nome: %s\n", clientes[i].nome);
        printf("Telefone: %s\n", clientes[i].telefone);
        printf("Endereço: %s\n", clientes[i].endereco);
        printf("\nData de entrada: %s\n", estadias[i].dataEntrada);
    }
}

```

```

printf("Data de saída: %s\n", estadias[i].dataSaida);
printf("Diárias: %d\n", estadias[i].Diarias);
printf("\nNúmero do quarto: %d\n", estadias[i].nQuarto);
printf("Quantidade de hóspedes: %d\n",
       quartos[estadias[i].nQuarto - 1].qtdHospedes);
printf("Valor da diária: %.2f\n",
       quartos[estadias[i].nQuarto - 1].valorDiaria);
printf("-----\n");
}
}

// Função para apagar reserva
void apagarReserva(struct cliente clientes[], struct estadia estadias[],
                  struct quarto quartos[], int *num_estadias,
                  int num_quartos) {
    if (*num_estadias == 0) {
        printf("Não há reservas cadastradas.\n");
    } else {
        listarReservas(clientes, estadias, quartos, *num_estadias);
        int numero_reserva;
        do {
            printf("Digite o número da reserva que deseja apagar (1 a %d): ",
                   *num_estadias);
            scanf("%d", &numero_reserva);

            if (numero_reserva < 1 || numero_reserva > *num_estadias) {
                printf("Número de reserva inválido. Digite novamente.\n");
            } else {
                // Liberar o quarto
                int num_quarto = estadias[numero_reserva - 1].nQuarto;
                quartos[num_quarto - 1].ocupado = 0;
            }
        } while (numero_reserva < 1 || numero_reserva > *num_estadias);
    }
}

```

```

// Remover a reserva
for (int i = numero_reserva - 1; i < *num_estadias - 1; i++) {
    clientes[i] = clientes[i + 1];
    estadias[i] = estadias[i + 1];
}
(*num_estadias)--;

printf("Reserva apagada com sucesso.\n");
break;
}
} while (1);
}
}

// Função para cadastrar um funcionário
void cadastrarFuncionario(struct funcionario funcionarios[], int *num_funcionarios) {
    printf("\nInsira os dados do funcionário:\n");
    printf("Nome: ");
    scanf("%[^\n]s", funcionarios[*num_funcionarios].nome);
    printf("Telefone: ");
    scanf("%[^\n]s", funcionarios[*num_funcionarios].telefone);
    printf("Cargo (repcionista, auxiliar de limpeza, garçom, gerente): ");
    scanf("%[^\n]s", funcionarios[*num_funcionarios].cargo);
    printf("Salário: ");
    scanf("%f", &funcionarios[*num_funcionarios].salario);

    funcionarios[*num_funcionarios].codigo = rand() % 999 + 1;

    printf("\nFuncionário cadastrado com sucesso.\n");
}

```

```
(*num_funcionarios)++;  
}
```

```
// Função para listar funcionários cadastrados
```

```
void listarFuncionarios(struct funcionario funcionarios[],  
                        int num_funcionarios) {  
    printf("\n----- Funcionários Cadastrados -----\\n");  
    for (int i = 0; i < num_funcionarios; i++) {  
        printf("Nome: %s\\n", funcionarios[i].nome);  
        printf("Telefone: %s\\n", funcionarios[i].telefone);  
        printf("Cargo: %s\\n", funcionarios[i].cargo);  
        printf("Salário: %.2f\\n", funcionarios[i].salario);  
        printf("Código: %d\\n", funcionarios[i].codigo);  
        printf("-----\\n");  
    }  
}
```

```
// Função para listar status dos quartos
```

```
void listarStatusQuartos(struct quarto quartos[], int num_quartos) {  
    printf("\n----- Status dos Quartos -----\\n");  
    for (int i = 0; i < num_quartos; i++) {  
        printf("Número do quarto: %d\\n", quartos[i].Nquarto);  
        printf("Quantidade de hóspedes: %d\\n", quartos[i].qtdHospedes);  
        printf("Valor da diária: %.2f\\n", quartos[i].valorDiaria);  
        printf("Status: %s\\n", quartos[i].ocupado ? "Ocupado" : "Desocupado");  
        printf("-----\\n");  
    }  
}
```

```
void gravarEstadiasClientesFuncionariosEmArquivo(struct estadia estadias[], int totalEstadias,  
struct cliente clientes[], int totalClientes, struct funcionario funcionarios[], int  
totalFuncionarios) {
```

```

FILE *arquivo;

arquivo = fopen("estadias.txt", "w");

if (arquivo == NULL) {
    printf("Erro ao abrir o arquivo!\n");
    return;
}

// Gravar estadias cadastradas
fprintf(arquivo, "\nEstadias cadastradas:\n");

for (int i = 0; i < totalEstadias; i++) {
    fprintf(arquivo, "Código do cliente: %d\n", estadias[i].codigo);
    fprintf(arquivo, "Número do quarto: %d\n", estadias[i].nQuarto);
    fprintf(arquivo, "Data de entrada: %s\n", estadias[i].dataEntrada);
    fprintf(arquivo, "Data de saída: %s\n", estadias[i].dataSaida);
    fprintf(arquivo, "Número de diárias: %d\n", estadias[i].Diarias);
    fprintf(arquivo, "-----\n");
}

// Gravar clientes cadastrados
fprintf(arquivo, "\nClientes cadastrados:\n");

for (int i = 0; i < totalClientes; i++) {
    fprintf(arquivo, "Código do cliente: %d\n", clientes[i].codigo);
    fprintf(arquivo, "Nome: %s\n", clientes[i].nome);
    fprintf(arquivo, "Endereço: %s\n", clientes[i].endereco);
    fprintf(arquivo, "Telefone: %s\n", clientes[i].telefone);
    fprintf(arquivo, "-----\n");
}

// Gravar funcionários cadastrados
fprintf(arquivo, "\nFuncionários cadastrados:\n");

for (int i = 0; i < totalFuncionarios; i++) {

```

```

        fprintf(arquivo, "Código do funcionário: %d\n", funcionarios[i].codigo);
        fprintf(arquivo, "Nome: %s\n", funcionarios[i].nome);
        fprintf(arquivo, "Telefone: %s\n", funcionarios[i].telefone);
        fprintf(arquivo, "Cargo: %s\n", funcionarios[i].cargo);
        fprintf(arquivo, "Salário: %.2f\n", funcionarios[i].salario);
        fprintf(arquivo, "-----\n");
    }

    fclose(arquivo);
}

void lerImprimirArquivoEstadias() {
    FILE *arquivo;
    arquivo = fopen("estadias.txt", "r");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo!\n");
        return;
    }

    char linha[200];
    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        printf("%s", linha);
    }

    fclose(arquivo);
}

int main(void) {
    struct cliente clientes[100];

```



```
struct estadia estadias[100];

struct quarto quartos[10];

struct funcionario funcionarios[100];


int num_clientes = 0;

int num_estadias = 0;

int num_funcionarios = 0;

int num_quartos = 10;


srand((unsigned int)time(NULL));


// Inicializando os quartos com valores fixos de diária e capacidade
for (int i = 0; i < 10; i++) {
    quartos[i].Nquarto = i + 1;
    quartos[i].ocupado = 0; // 0 para desocupado


    if (i + 1 >= 7 && i + 1 <= 10) {
        quartos[i].qtdHospedes = 6;
        quartos[i].valorDiaria = 310.0;
        quartos[i].tipo = 3;
    } else if (i + 1 >= 4 && i + 1 <= 6) {
        quartos[i].qtdHospedes = 4;
        quartos[i].valorDiaria = 240.0;
        quartos[i].tipo = 2;
    } else {
        quartos[i].qtdHospedes = 2;
        quartos[i].valorDiaria = 190.0;
        quartos[i].tipo = 1;
    }
}
```

```

int opcao;

do {
    printf("\n-----\n");
    printf("    Hotel Descanso Garantido\n");
    printf("-----\n");
    printf("\n---- Menu ----\n");
    printf("1. Fazer uma reserva\n");
    printf("2. Ver reservas cadastradas\n");
    printf("3. Apagar reserva\n");
    printf("4. Cadastrar funcionário\n");
    printf("5. Ver funcionários cadastrados\n");
    printf("6. Verificar status dos quartos\n");
    printf("7. Gravar dados em arquivo estadias.txt\n");
    printf("8. Ler e imprimir dados do arquivo estadias.txt\n");
    printf("9. Sair\n");
    printf("Escolha uma opção: ");
    scanf("%d", &opcao);

    switch (opcao) {
    case 1:
        // Opção para fazer uma reserva
        if (num_clientes < 100 && num_estadias < 100) {
            cadastrarCliente(clientes, &num_clientes);
            if (num_clientes > 0) {
                cadastrarEstadia(estadias, clientes, quartos, &num_estadias,
                                num_quartos);
            }
        } else {
            printf("Número máximo de clientes ou estadias atingido.\n");
        }
        break;

```

case 2:

```
// Opção para listar reservas cadastradas
if (num_estadias > 0) {
    listarReservas(clientes, estadias, quartos, num_estadias);
} else {
    printf("Não há reservas cadastradas.\n");
}
break;
```

case 3:

```
// Opção para apagar reserva
apagarReserva(clientes, estadias, quartos, &num_estadias, num_quartos);
break;
```

case 4:

```
// Opção para cadastrar funcionário
if (num_funcionarios < 100) {
    cadastrarFuncionario(funcionarios, &num_funcionarios);
} else {
    printf("Número máximo de funcionários atingido.\n");
}
break;
```

case 5:

```
// Opção para listar funcionários cadastrados
if (num_funcionarios > 0) {
    listarFuncionarios(funcionarios, num_funcionarios);
} else {
    printf("Não há funcionários cadastrados.\n");
}
```

```
break;
```

```
case 6:
```

```
// Opção para verificar status dos quartos
```

```
listarStatusQuartos(quartos, num_quartos);
```

```
break;
```

```
case 7:
```

```
// Função para gravar dados em arquivo estadias.txt
```

```
gravarEstadiasClientesFuncionariosEmArquivo(estadias, num_estadias, clientes,  
num_clientes, funcionarios, num_funcionarios);
```

```
printf("Dados gravados com sucesso em estadias.txt.\n");
```

```
break;
```

```
case 8:
```

```
// Função para ler e imprimir dados do arquivo estadias.txt
```

```
lerImprimirArquivoEstadias();
```

```
break;
```

```
case 9:
```

```
printf("Fechando o programa...\n");
```

```
break;
```

```
default:
```

```
printf("Opção inválida. Escolha novamente.\n");
```

```
}
```

```
} while (opcao != 9);
```

```
return 0;
```

