ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Projeto de Laboratório de Programação

Licenciatura em Segurança Informática em Redes de Computadores

2019/2020

Christopher Meder – 8170022

Índice

Introdução 4
Funcionalidades 5
Requeridas 5
Propostas 5
Estrutura do projeto 7
Estruturas 7
Funcionalidades Implementadas 8
Conclusão 22

Figura 1 - Estruturas 7
Figura 2 - Função Main 2 8
Figura 3 - Função Main 8
Figura 4 – Menu Funcionário 9
Figura 5 – Funcao add Reserva 9
Figura 6 - Função add Reserva 2 10
Figura 7 - Função add Reserva 3 10
Figura 8 - Função Listar Compras 11
Figura 9 - Função gravar ficheiro Reserva 11
Figura 10 - Função carregar ficheiro 12
Figura 12 - Função add Cliente 12
Figura 11 - Função listar Reserva 13
Figura 13 - Função verificar Cliente 13
Figura 15 - Função gravar ficheiro Cliente 14
Figura 14 - Função editar Cliente 14
Figura 16 - Função carregar ficheiro Cliente 15
Figura 18 - Função listar Cliente 15
Figura 17 - Função Menu Gestor 16
Figura 19 - Função Menu Informacao 16
Figura 20 - Função fmensal 17
Figura 21 - Função Menu Faturas 17
Figura 22 -Função Menu Quarto 18
Figura 23 - Função addQuarto 19
Figura 24 - Função Verificar Quarto 19
Figura 25 - Função listar Quarto 19
Figura 26 - Função gravar ficheiro Quarto 20
Figura 27 - Função carregar ficheiro Quarto 20
Figura 28 - Função editar Quarto 21
Figura 29 - Função indisponível Quarto 21

Introdução

No âmbito da unidade curricular de Laboratório de Programação, foi realizado um trabalho com o objetivo de compreender e dominar os conhecimentos práticos e teóricos sobre a linguagem de programação em C, abordados durante a unidade curricular.

Os hotéis existem desde longa data e são vulgarmente conhecidos como estabelecimentos de repouso que são mais comuns usados pelos turistas. Estes estabelecimentos têm sofrido grandes alterações ao longo dos tempos no que concerne aos sistemas de gestão informáticos.

Assim sendo, no decorrer deste relatório, será explicado os passos necessários para a realização de um sistema automatizado.

Funcionalidades

Requeridas

- **Funcionário**

  -Criação de novos Clientes;

  -Editar as informações dos Clientes;

  -Introduzir as reservas no sistema;

  -Guardar as todas as informações;

  -Carregar Informações dos Clientes;

  -Carregar Informações das Reservas;

  -Carregar Informações dos Quartos;

- **Gestor**

  -Pode adicionar, editar, remover, altera o estado e lista Quartos;

  -Consultar toda as informações dos Clientes;

  - Consultar toda as informações dos Quartos;

  - Consultar toda as informações dos Reservas;

  - Consultar Faturas;

  -Carregar Informações dos Clientes;

  -Carregar Informações das Reservas;

  -Carregar Informações dos Quartos;

  -Remover Quartos;

- Memoria Dinâmica;
- Persistência de Dados;

Propostas

- O Funcionário pode criar Clientes, verificando automaticamente a existência de outro Cliente com o mesmo NIF;
- O Funcionário pode criar Reservas, verificando automaticamente a existência de outra Reserva com o mesmo número;
- O Funcionário pode editar todas as informações dos Clientes através do NIF;
- O Funcionário pode guardar para um ficheiro a parte essas mesmas informações;
- O Funcionário pode gravar as reservas num ficheiro;
- O Funcionário pode carregar o ficheiro de informações dos Clientes;

- O Diretor pode adicionar, editar, alterar estado e listar quartos através dos seus Números;
- O Diretor pode carregar as informações do ficheiro a parte dos Quartos;
- O Diretor pode carregar as informações do ficheiro a parte das Reservas;
- O Diretor pode carregar as informações do ficheiro a parte dos Clientes;
- O Diretor pode gravar os Quartos num ficheiro;
- O Diretor pode carregar as informações dos Clientes de um ficheiro;
- O Diretor pode listar os Clientes;
- O Diretor pode carregar as informações das Compras;
- O Diretor pode listar as mesmas informações das Compras;
- O Diretor pode carregar as informações das Reservas;
- O Diretor pode listar as mesmas informações das Reservas;
- Remover Quarto;
- Listar Quartos Ocupados;
- Listar Quartos em Manutenção;
- Listar Reserva a cima dos 200$;

Estrutura do projeto

O projeto foi realizado numa criação de vários ficheiros .c, onde cada um contem informações do programa sendo que o Cliente.c terá toda as funções do cliente, o Faturas.c das faturas, o Quarto.c dos quartos e a Reserva.c das reservas. Todas as estruturas utilizadas foram criadas num ficheiro .h a parte.

Estruturas

A Estrutura Cliente armazena toda a informação do Cliente, sendo ela o Nome, o NIF, o seu número de Cartão de Cidadão e para ver se é empresarial ou não.

A Estrutura Reserva contem um código de reserva, número de quartos, preço dos quartos, número de adultos, número de crianças, número de noites, número de quartos que pretende alugar, o tipo de pensão, o tipo de serviço, o preço total e a data.

A Estrutura Quarto contem o código do quarto, o estado do quarto e o tipo de quarto.

```c
#ifndef STRUCT_H
#define STRUCT_H
#define MAX 3
#define DEC 1

#ifdef __cplusplus
extern "C" {
#endif

    struct Cliente{
        char nome[30];
        int nif;
        int ncc;
        char empresa[3];
    };

    struct Reserva{
        int codReserva;
        int quartos;
        float precoQuarto;
        int numAdultos;
        int numCriancas;
        int numNoites;
        int numQuarto;
        char pensao[1];
        int tpservico;
        float precoTotal;
        int data;

    };

    struct Quarto{
        int codQuarto;
        char ativo[2];
        char tipoQuarto[3];
    };

       typedef struct tm tm;

#ifdef __cplusplus
}
#endif

#endif /* STRUCT_H */
```

Figura 1 - Estruturas

Funcionalidades Implementadas

Este ficheiro apresenta a função principal, onde é possível escolher entre 2 opções. A primeira, permite fazer “Login” no programa, no qual se entrar com “F” ou “f” ira entrar como Funcionário sem necessitar de uma palavra-passe para autenticar, também podemos entrar com Diretor Clínico sem necessitar de uma palavra-passe com “G” ou “g”. Na segunda opção o programa termina. Quando se autenticar com um dos dois ira entrar no “menuFuncionario” ou “menuGestor”.

```
int main(int argc, char** argv) {
    struct Cliente *Clientes;
    struct Reserva *Reservas;
    struct Quarto *Quartos;

    char utilizador[1];
    int n_clientes = 0, n_reservas = 0, n_quartos = 0;
    int opcao, n_max = 0, n_maxR = 0, n_maxQ = 0;
    Clientes = (struct Cliente *) malloc (sizeof (struct Cliente) * MAX);
    Reservas = (struct Reserva *) malloc (sizeof (struct Reserva) * MAX);
    Quartos = (struct Quarto *) malloc (sizeof (struct Quarto) * MAX);

    n_max = MAX;
    n_maxR = MAX;
    n_maxQ = MAX;

    if ((Clientes == NULL) || (Quartos == NULL) || (Reservas == NULL)){
        printf("Erro \n");
        exit(0);
    }

    do {
        printf("*-------------------------* \n");
        printf("          Menu             \n");
        printf("*-------------------------* \n");
        printf("1 - Login \n");
        printf("0 - Sair \n");
        do {
            printf("Escolha uma opcao >> ");
            scanf("%d", &opcao);
        }
        while (opcao < 0 || opcao > 1);
        switch (opcao) {
            case 0: printf("A sair... \n");
                break;

            case 1: printf("Utilizador: ");
                    scanf("%s", &utilizador);
                    if((strcmp(utilizador, "F") == 0) || (strcmp(utilizador, "f") == 0)) {
                        carregarfc(&Clientes, &n_clientes, &n_max);
                        menuFuncionario(&Clientes, &n_clientes, &Reservas, &n_reservas, &Quartos, &n_quartos, &n_max , &n_maxQ, &n_maxR);
                    }
                    else if((strcmp(utilizador, "G") == 0) || (strcmp(utilizador, "g") == 0)){
                        carregarfc(&Clientes, &n_clientes, &n_max);
                        carregarfq(&Quartos, &n_quartos, &n_maxQ);
                        carregarfr(&Reservas, &n_reservas, &n_maxR);
                        menuGestor(&Clientes, &n_clientes, &Reservas, &n_reservas,&Quartos, &n_quartos, &n_max , &n_maxQ, &n_maxR);
                    }
                    else {
                        printf("Utilizador errado! \n");
                    }
                   break;
```

Figura 3 - Função Main

```
                }
                break;

            default: puts("A opcao e invalida! \n");
        }
    }
    while (opcao != 0);
    free(Clientes);
    free(Reservas);
    free(Quartos);
    return (EXIT_SUCCESS);
}
```

Figura 2 - Função Main 2

No caso de autenticar como Funcionário ira abrir outro menu com 4 opções sendo que a primeira, permite criar um Reserva, a segunda opção ira criar um novo cliente, a terceira opção vai guardar as reservas e os clientes num ficheiro.

```c
void menuFuncionario(struct Cliente **Clientes, int *n_clientes, struct Reserva **Reservas,  int *n_reservas, struct Quarto **Quartos, int *n_quartos, int *n_max, int *n_maxR){

    int opcao;

    do {
        printf("*------------------------* \n");
        printf("       Funcionario         \n");
        printf("*------------------------* \n");
        printf("1 - Nova Reserva \n");
        printf("2 - Addicionar \n");
        printf("3 - Guardar \n");
        printf("4 - Editar \n");
        printf("0 - Recuar \n");
        do {
            printf("Escolha uma opcao >> ");
            scanf("%d", &opcao);
        }
        while (opcao < 0 || opcao > 4);
        switch (opcao) {
            case 0:
                break;

            case 1: addReserva(Reservas, n_reservas, Quartos, n_quartos, n_maxR);
                break;

            case 2: addClient(Clientes, n_clientes, n_max);
                break;

            case 3: gravarfc(*Clientes, *n_clientes);
                    gravarfr(*Reservas, *n_reservas);
                break;

            case 4: editarClient(*Clientes, *n_clientes);
                break;

            default: puts("A opcao e invalida! \n");
        }
    }
    while (opcao != 0);
}
```

Figura 4 – Menu Funcionário

Ao abrir o adicionar Reserva ira pedir o número de reserva caso existe essa reserva vai dizer q já existe senão vai pedir quantos quartos queremos o número do quarto, o número de pessoas, número de crianças, a pensão quer pretendemos, o número de noites e o preço total.

```c
void addReserva(struct Reserva **Reservas, int *n_reservas, struct Quarto **Quartos, int *n_quartos, int *n_maxR){
    struct Reserva * temp;
    int numReserva, existe, numCriancas, numQuarto, existeQuarto, existeverificiar, refecoes, i, nquartos, numAdultos;

    time_t atual;
    time(&atual);

    if(*n_reservas == *n_maxR){
        temp = (struct Reserva*) realloc (*Reservas, sizeof(struct Reserva)*((*n_reservas) + MAX));

        if (temp == NULL){
            printf("Erro na realocacao! \n");
        }
        else{
            *Reservas = temp;
            *n_maxR = (*n_reservas) + MAX;
            printf("Numero da Reserva: ");
            scanf("%d",&numReserva);
            existe = verificarReserva(*Reservas,*n_reservas,numReserva);

            if(existe == -1){
                (*Reservas)[*n_reservas].codReserva = numReserva;

                printf("Quantos quartos: ");
                scanf("%d",&nquartos);

                (*Reservas)[*n_reservas].quartos = nquartos;

                listarQuarto(Quartos, n_quartos);

                for (i=0; i < nquartos; i++){

                printf("Escolha um quarto: ");
                scanf("%d", &numQuarto);
                existeQuarto = verificarQuarto(*Quartos,*n_quartos,numQuarto);

                if (existeQuarto != -1){

                    strcpy((*Quartos)[*n_quartos].ativo, "O");
                    existeverificiar = verificartipo(*Quartos, *n_quartos, numQuarto);

                    if (existeverificiar != 50){
                        (*Reservas)[*n_reservas].precoQuarto = 75;
                    }
                    else{
                        (*Reservas)[*n_reservas].precoQuarto = 50;
                    }
                        printf("Numero de Adultos: ");
                        scanf("%d",&numAdultos);
```

Figura 5 – Funcao add Reserva

```
if (numAdultos <= 2){
    (*Reservas)[*n_reservas].numAdultos = numAdultos;
}
else{
    printf("Demasiados adultos para um quarto! \n");
}

printf("Numero de Criancas: ");
scanf("%d",&numCriancas);
if (numCriancas <= 2){
    (*Reservas)[*n_reservas].numCriancas = numCriancas;
}
else{
    printf("Demasiadas criancas para um quarto! \n");
}

printf("Tipo de Pensao (B:Base M:Meia C:Completa): ");
scanf("%s",(*Reservas)[*n_reservas].pensao);
    if ((strcmp((*Reservas)[*n_reservas].pensao, "B") == 0) || (strcmp((*Reservas)[*n_reservas].pensao, "b") == 0)){
          printf("Quantas refecoes ira fazer: ");
          scanf("%d",refecoes);
          (*Reservas)[*n_reservas].tpservico = ((8 * (*Reservas)[*n_reservas].numAdultos) + (4 * (*Reservas)[*n_reservas].numCriancas)) * refecoes;
    }
    else if ((strcmp((*Reservas)[*n_reservas].pensao, "M") == 0) || (strcmp((*Reservas)[*n_reservas].pensao, "m") == 0)){
          (*Reservas)[*n_reservas].tpservico = ((8 * (*Reservas)[*n_reservas].numAdultos) + (4 * (*Reservas)[*n_reservas].numCriancas)) * 2;
    }
    else if ((strcmp((*Reservas)[*n_reservas].pensao, "C") == 0) || (strcmp((*Reservas)[*n_reservas].pensao, "c") == 0)){
          (*Reservas)[*n_reservas].tpservico = ((8 * (*Reservas)[*n_reservas].numAdultos) + (4 * (*Reservas)[*n_reservas].numCriancas)) * 3;
    }

printf("Numero de Noites: ");
scanf("%d",&(*Reservas)[*n_reservas].numNoites);

if (existeverificiar == 50){
    (*Reservas)[*n_reservas].precoTotal = (*Reservas)[*n_reservas].precoQuarto + (*Reservas)[*n_reservas].tpservico  + (15 * (*Reservas)[*n_reservas].numCriancas) * (*Reservas)[*n_reservas].numNoites;
}
else{
    (*Reservas)[*n_reservas].precoTotal = (*Reservas)[*n_reservas].precoQuarto + (*Reservas)[*n_reservas].tpservico  + (15 * (*Reservas)[*n_reservas].numCriancas) * (*Reservas)[*n_reservas].numNoites;
}

tm *myTime = localtime(&atual);
printf("Data: %i/%i/%i \n", myTime->tm_mday, myTime->tm_mon + 1, myTime->tm_year + 1900);

printf("Preco Total: %.2f€ \n",(*Reservas)[*n_reservas].precoTotal );

printf("'------------------' \n");


(*n_reservas)++;
```

Figura 6 - Função add Reserva 2

```
                    (*n_reservas)++;
                }
                else{
                    printf("Quarto nao existe! \n");
                    }
                }
            }
            else{
                printf("Essa Reserva ja existe! \n");
                }
        }
    }
```

Figura 7 - Função add Reserva 3

Dentro da função addReserva, a função ira chamar outras funções para ajudar na verificação dos dados introduzidos, sendo que vai verificar o código da reserva e o tipo de reserva.

```
int verificarReserva(struct Reserva *Reservas, int n_reservas, int numReserva){
    int i, e;

    e= -1;

     for (i=0; i < n_reservas; i++){
        if (Reservas[i].codReserva == numReserva){
            e = i;
        }
     }
 return e;
}

/**
 * Funcao para verificar o tipo de quarto para atribuir um valor
 * @param Quartos, apontador para a estrutura de quartos
 * @param n_quartos, numero de quartos
 * @param numQuarto, variavel que vai verificar o tipo de quarto
 * @return
 */
int verificartipo(struct Quarto *Quartos, int n_quartos, int numQuarto){
    int i, e;

    for (i = 0; i<n_quartos;i++){
        if (Quartos[i].codQuarto == numQuarto){
            if ((strcmp(Quartos[i].tipoQuarto, "SR") == 0) || (strcmp(Quartos[i].tipoQuarto, "sr") == 0)){
                e = 50;
            }
            else{
                e = 75;
            }
        }
    }
    return e;
}
```

Figura 8 - Função Listar Compras

Na função gravar ficheiro reserva, as reservas vão ser gravadas num ficheiro chamado “reserva.txt”.

```
void gravarfr(struct Reserva *Reservas, int n_reservas){
    FILE * f;

    int i;

    f = fopen ("reservas.txt","w");

    if (f == NULL){
        printf("Nao foi possivel abrir o ficheiro! \n");
    }
    else{
        fwrite(&n_reservas,sizeof(int),1,f);
        i=fwrite(Reservas, sizeof(struct Reserva),n_reservas,f);
        printf("Escreveu no ficheiro %d reserva(s). \n", i);
        fclose(f);
    }
}
```

Figura 9 - Função gravar ficheiro Reserva

A função carregarfr, serve para carregar as reservas que foram gravadas no ficheiro “reservas.txt”

```
void carregarfr(struct Reserva **Reservas, int *n_reservas, int *n_maxR){
    FILE * f;
    int i, nreservas;
    struct Reserva * reservaTemp;

    f = fopen ("reservas.txt", "r");

    if (f == NULL){
        printf("Nao foi possivel carregar o ficheiro! \n");
    }
    else{
        nreservas= fread (n_reservas,sizeof(int),1,f);
        if (nreservas == 0){
            printf("Erro na leitura do ficheiro! \n");
    }
        else{
            reservaTemp = (struct Reserva*) realloc (*Reservas,sizeof(struct Reserva) * (*n_reservas + MAX));

                if (reservaTemp==NULL){
                    printf("Erro ao carregar o ficheiro! \n");
                }
                else{
                    *Reservas = reservaTemp;
                    *n_maxR = *n_reservas + MAX;
                    i = fread(*Reservas, sizeof(struct Reserva),*n_reservas,f);
                    printf(" %d Reservas carregados com sucesso. \n", i);
                    *n_reservas = i;
                    fclose(f);
                }
            }
    }
}
```

Figura 10 - Função carregar ficheiro

Na segunda opção do menuFuncionario pode se criar um cliente no qual ira preguntar se vai ser cliente de empresa ou não caso seja de empresa apenas pergunta pelo NIF e nome da empresa.

```
void addClient(struct Cliente **Clientes, int *n_clientes, int *n_max){
    struct Cliente * temp;
    int e_nif, existe;
    char empresa[3];

    if(*n_clientes == *n_max){
        temp=(struct Cliente*) realloc (*Clientes, sizeof(struct Cliente)*((*n_clientes) + MAX));

        if (temp == NULL){
            printf("Erro na realocacao! \n");
        }
        else{
            *Clientes = temp;
            *n_max = (*n_clientes) + MAX;
            printf("NIF do Cliente: ");
            scanf("%d",&e_nif);
            existe = verificarClient(*Clientes,*n_clientes,e_nif);
            if(existe == -1){
                (*Clientes)[*n_clientes].nif = e_nif;
                printf("Empresa: ");
                scanf("%s", empresa);
                if ((strcmp(empresa, "Sim") == 0) || (strcmp(empresa, "sim") == 0)){
                    strcpy((*Clientes)[*n_clientes].empresa, empresa);
                    printf("Nome da Empresa: ");
                    scanf("%s",&(*Clientes)[*n_clientes].nome);
                }
                else{
                    strcpy((*Clientes)[*n_clientes].empresa, empresa);
                    printf("Nome do Cliente: ");
                    scanf("%s",&(*Clientes)[*n_clientes].nome);
                    printf("Numero CC do Cliente: ");
                    scanf("%d",&(*Clientes)[*n_clientes].ncc);
                }
                (*n_clientes)++;
            }
            else{
                printf("Esse NIF ja existe! \n");
            }
        }
    }
```

Figura 11 - Função add Cliente

A função listarReserva, vai listar todas as reservas que foram criados dos respetivos códigos de reservas.

```
void listarReservas(struct Reserva *Reservas, int n_reservas){
    int i, numReserva, existe;

    printf("Numero da Reserva: ");
    scanf("%d",&numReserva);
    existe = verificarReserva(Reservas,n_reservas,numReserva);

    if(existe != -1){
        for (i = 0; i<n_reservas;i++){
            printf("-------------\n");
            printf("Numero da Reserva: %d \n",Reservas[i].codReserva);
            printf("Adultos: %d \n",Reservas[i].numAdultos);
            printf("Criancas: %d \n",Reservas[i].numCriancas);
            printf("Quarto: %d \n",Reservas[i].numQuarto);
            printf("Preco do Quarto: %.2f \n",Reservas[i].precoQuarto);
            printf("Tipo de Servico: %s \n",Reservas[i].pensao);
            printf("Noites: %d \n",Reservas[i].numNoites);
            printf("Preco Total: %.2f \n",Reservas[i].precoTotal);
        }
    }
    printf("-------------\n");
}
```

Figura 12 - Função listar Reserva

Esta função verificarClient ira verificar se o nif introduzido já existe caso já existe não ira deixar criar um novo cliente.

```
int verificarClient(struct Cliente *Clientes, int n_clientes, int e_nif){
    int i, e;

    e= -1;

     for (i=0; i < n_clientes; i++){
        if (Clientes[i].nif == e_nif){
            e = i;
        }
        if (e == -1){
        }
     }
return e;
}
```

Figura 13 - Função verificar Cliente

Na função para editar o cliente, editarCliente, iremos poder editar o nome da empresa caso o cliente for da empresa e o seu nif, se não for empresarial ira deixar alterar todos os dados.

```
void editarClient(struct Cliente *Clientes, int n_clientes){
    int i, e_nif;

    printf("Introduza o NIF do Cliente: ");
    scanf("%d",&e_nif);

    i = verificarClient(Clientes, n_clientes, e_nif);

    if (i != -1){
        printf("-------------\n");
        if ((strcmp(Clientes[i].empresa, "Sim") == 0) || (strcmp(Clientes[i].empresa, "sim") == 0)){
            printf("Nome da Empresa: ");
            scanf("%s",&Clientes[i].nome);
            printf("NIF: ");
            scanf("%d",&Clientes[i].nif);
        }
        else{
            printf("Nome: ");
            scanf("%s",&Clientes[i].nome);
            printf("NIF: ");
            scanf("%d",&Clientes[i].nif);
            printf("CC: ");
            scanf("%d",&Clientes[i].ncc);
        }
        printf("-------------\n");
    }
}
```

Figura 15 - Função editar Cliente

Na função gravar ficheiro do cliente, podemos gravar todas as informações dos clientes no ficheiro “informações.txt”.

```
void gravarfc(struct Cliente *Clientes, int n_clientes){
    FILE * f;

    int i;

    f = fopen ("informacoes.txt","w");

    if (f == NULL){
        printf("Nao foi possivel abrir o ficheiro! \n");
    }
    else{
        fwrite(&n_clientes,sizeof(int),1,f);
        i=fwrite(Clientes, sizeof(struct Cliente),n_clientes,f);
        printf("Escreveu no ficheiro %d cliente(s). \n", i);
        fclose(f);
    }
}
```

Figura 14 - Função gravar ficheiro Cliente

Para carregarmos os ficheiros dos clientes usamos a função carrregarfc, que ira buscar todas as informações do ficheiro “informações.txt”.

```
void carregarfc(struct Cliente **Clientes, int *n_clientes, int *n_max){
    FILE * f;
    int i, nclientes;
    struct Cliente * clientTemp;

    f = fopen ("informacoes.txt", "r");

    if (f == NULL){
        printf("Nao foi possivel carregar o ficheiro! \n");
    }
    else{
        nclientes= fread (n_clientes,sizeof(int),1,f);
        if (nclientes == 0){
            printf("Erro na leitura do ficheiro! \n");
    }
        else{
            clientTemp = (struct Cliente*) realloc (*Clientes,sizeof(struct Cliente) * (*n_clientes + MAX));

                if (clientTemp==NULL){
                    printf("Erro ao carregar o ficheiro! \n");
                }
                else{
                    *Clientes = clientTemp;
                    *n_max = *n_clientes + MAX;
                    i = fread(*Clientes, sizeof(struct Cliente),*n_clientes,f);
                    printf(" %d Clientes carregados com sucesso. \n", i);
                    *n_clientes = i;
                    fclose(f);
                }
            }
    }
}
```

Figura 16 - Função carregar ficheiro Cliente

Na função listar cliente iremos listar todos os clientes.

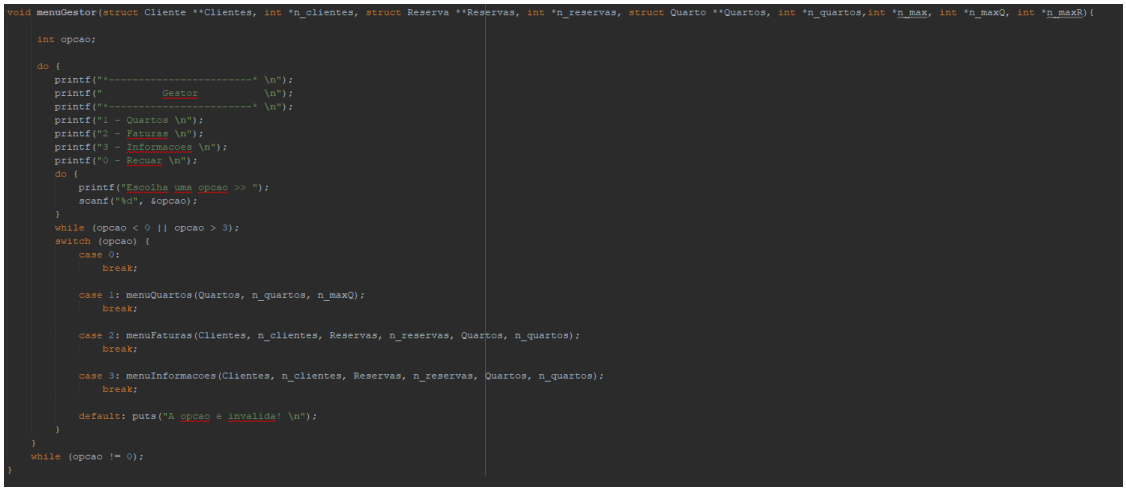
```
void listarClient(struct Cliente *Clientes, int n_clientes){
    int i;

    for (i = 0; i<n_clientes;i++){
        printf("-------------\n");
        if ((strcmp(Clientes[i].empresa, "Sim") == 0) || (strcmp(Clientes[i].empresa, "sim") == 0)){
            printf("Empresa: %s \n",Clientes[i].nome);
            printf("NIF: %d \n",Clientes[i].nif);
        }
        else{
            printf("Nome: %s \n",Clientes[i].nome);
            printf("NIF: %d \n",Clientes[i].nif);
            printf("CC: %d \n",Clientes[i].ncc);
        }

    }
    printf("-------------\n");
}
```

Figura 17 - Função listar Cliente

No menu Gestor podemos escolher outros menus como Quartos, Faturas ou Informacoes.



```
void menuGestor(struct Cliente **Clientes, int *n_clientes, struct Reserva **Reservas, int *n_reservas, struct Quarto **Quartos, int *n_quartos,int *n_max, int *n_maxQ, int *n_maxR){

    int opcao;

    do {
       printf("*------------------------* \n");
       printf("          Gestor          \n");
       printf("*------------------------* \n");
       printf("1 - Quartos \n");
       printf("2 - Faturas \n");
       printf("3 - Informacoes \n");
       printf("0 - Recuar \n");
       do {
           printf("Escolha uma opcao >> ");
           scanf("%d", &opcao);
       }
       while (opcao < 0 || opcao > 3);
       switch (opcao) {
           case 0:
               break;

           case 1: menuQuartos(Quartos, n_quartos, n_maxQ);
               break;

           case 2: menuFaturas(Clientes, n_clientes, Reservas, n_reservas, Quartos, n_quartos);
               break;

           case 3: menuInformacoes(Clientes, n_clientes, Reservas, n_reservas, Quartos, n_quartos);
               break;

           default: puts("A opcao e invalida! \n");
       }
   }
   while (opcao != 0);
}
```

Figura 18 - Função Menu Gestor

Neste menu vamos chamar todas as funções de listar.

```
void menuInformacoes(struct Cliente **Clientes, int *n_clientes,struct Reserva **Reservas, int *n_reservas,struct Quarto **Quartos, int *n_quartos){

    int opcao;

    do {
       printf("*------------------------* \n");
       printf("       Informacoes        \n");
       printf("*------------------------* \n");
       printf("1 - Consultar Quartos \n");
       printf("2 - Consultar Clientes \n");
       printf("3 - Consultar Reservas \n");
       printf("0 - Recuar \n");
       do {
           printf("Escolha uma opção >> ");
           scanf("%d", &opcao);
       }
       while (opcao < 0 || opcao > 3);
       switch (opcao) {
           case 0:
               break;

           case 1: listarQuarto(*Quartos, *n_quartos);
               break;

           case 2: listarClient(*Clientes, *n_clientes);
               break;

           case 3: listarReservas(*Reservas, *n_reservas);
               break;

           default: puts("A opcao e invalida! \n");
       }
   }
   while (opcao != 0);
}
```

Figura 19 - Função Menu Informacao

No Menu das Faturas iremos imprimir as faturas.

```c
void menuFaturas(struct Cliente **Clientes, int *n_clientes,struct Reserva **Reservas, int *n_reservas){

    int opcao;

    do {
       printf("*------------------------* \n");
       printf("          Faturas          \n");
       printf("*------------------------* \n");
       printf("1 - Mensal \n");
       printf("0 - Recuar \n");
       do {
           printf("Escolha uma opcao >> ");
           scanf("%d", &opcao);
       }
       while (opcao < 0 || opcao > 2);
       switch (opcao) {
           case 0:
               break;

           case 1: fmensal(*Reservas, *n_reservas, *Clientes, *n_clientes);
               break;

           default: puts("A opcao e invalida! \n");
       }
    }
    while (opcao != 0);
}
```

Figura 21 - Função Menu Faturas

A função fmensal ira listar as faturas de cada mês.

```c
void fmensal(struct Reserva *Reservas, int n_reservas, struct Cliente *Clientes, int n_clientes){

    int i, e_nif, existe, numReserva, existeR;
    int x;

        printf("NIF do Cliente: ");
        scanf("%d",&e_nif);
        existe = verificarClient(*Clientes,n_clientes,e_nif);

    if (existe != 1){
        for (i=0; i < n_clientes ; i++){
            printf("*-----FATURA---*\n");
            printf("Nome: %s \n", Clientes[i].nome);
            printf("NIF: %d \n", Clientes[i].nif);

            printf("Numero da Reserva: ");
            scanf("%d",&numReserva);
            existeR = verificarReserva(*Reservas,n_reservas,numReserva);

            if(existeR != -1){
            for (x=0; x < n_reservas; x++){
                printf("Numero do quarto:%d \n", Reservas[x].numQuarto);
                printf("Numero de adultos:%d \n", Reservas[x].numAdultos);
                printf("Numero de crianças:%d \n", Reservas[x].numCriancas);
                printf("Numero de noites:   %d              \n", Reservas[x].numNoites);
                if ((strcmp(Reservas[x].pensao, "b")==0) || (strcmp(Reservas[x].pensao, "B")==0)){
                    printf("Pensao: Base \n");
                } else if ((strcmp(Reservas[x].pensao, "m")==0) || (strcmp(Reservas[x].pensao, "M")==0)){
                    printf("Pensao: Meia \n");
                } else if ((strcmp(Reservas[x].pensao, "c")==0) || (strcmp(Reservas[x].pensao, "C")==0)){
                    printf("Pensao: Completa \n");
                }
            }

            printf("Preco Total: %.2f \n",Reservas[x].precoTotal);
            printf("*------------*\n");
            }
        }

    } else {
        printf("NIF ou Reserva invalido! \n");
    }
}
```

Figura 20 - Função fmensal

No menuQuarto podemos criar, mudar a disponibilidade dos quartos, remove , gravar e editar os quartos.

```
void menuQuartos(struct Quarto **Quartos, int *n_quartos,int *n_maxQ){

    int opcao;

    do {
       printf("*------------------------* \n");
       printf("          Quarto           \n");
       printf("*------------------------* \n");
       printf("1 - Adicionar \n");
       printf("2 - Tornar Indesponivel \n");
       printf("3 - Remover \n");
       printf("4 - Guardar \n");
       printf("5 - Editar \n");
       printf("0 - Recuar \n");
       do {
           printf("Escolha uma opcao >> ");
           scanf("%d", &opcao);
       }
       while (opcao < 0 || opcao > 5);
       switch (opcao) {
           case 0:
               break;

           case 1: addQuarto(Quartos, n_quartos, n_maxQ);
               break;

           case 2: indesponivelQuarto(*Quartos, *n_quartos);
               break;

           case 3: //removerQuarto();
               break;

           case 4: gravarfq(*Quartos, *n_quartos);
               break;

           case 5: editarQuarto(*Quartos, *n_quartos);
               break;

           default: puts("A opcao e invalida! \n");
       }
    }
    while (opcao != 0);
}
```

Misspelled Word

Figura 22 -Função Menu Quarto

A função “addQuarto” ira permitir criarmos um quarto sendo que não permite a criação de um quarto existente.

```
void addQuarto(struct Quarto **Quartos, int *n_quartos, int *n_maxQ){
    struct Quarto * temp;
    int numQuarto, existe;
    char tpquarto[2];

    if(*n_quartos == *n_maxQ){
        temp=(struct Quarto*) realloc (*Quartos, sizeof(struct Quarto)*((*n_quartos) + MAX));

        if (temp == NULL){
            printf("Erro na realocacao! \n");
        }
        else{
            *Quartos = temp;
            *n_maxQ = (*n_quartos) + MAX;
            printf("Numero do Quarto: ");
            scanf("%d",&numQuarto);
            existe = verificarQuarto(*Quartos,*n_quartos,numQuarto);

            if(existe == -1){
                (*Quartos)[*n_quartos].codQuarto = numQuarto;

                printf("Qual e o Tipo de Quarto (SR - DR): ");
                scanf("%s",tpquarto);

                if((strcmp(tpquarto, "SR") == 0) || (strcmp(tpquarto, "sr") == 0) || (strcmp(tpquarto, "DR") == 0) || (strcmp(tpquarto, "dr") == 0)){
                    strcpy((*Quartos)[*n_quartos].tipoQuarto, tpquarto);
                }

                strcpy((*Quartos)[*n_quartos].ativo, "L");
                (*n_quartos)++;
            }
            else{
                printf("Esse quarto ja existe! \n");
            }
        }
    }
```

Figura 23 - Função addQuarto

Função para verificar se o quarto existe.

```
int verificarQuarto(struct Quarto *Quartos, int n_quartos, int numQuarto){
    int i, e;

    e= -1;

     for (i=0; i < n_quartos; i++){
        if (Quartos[i].codQuarto == numQuarto){
            e = i;
        }
     }
return e;
}
```

Figura 24 - Função Verificar Quarto

Função para listar todos os quarto, sendo que se tiver Ocupado ou em Manutenção ira dizer as respetivas informações.

```
void listarQuarto(struct Quarto *Quartos, int n_quartos){
    int i;

    for (i = 0; i<n_quartos;i++){
        if ((strcmp(Quartos[i].ativo, "L") == 0) || (strcmp(Quartos[i].ativo, "l") == 0)){
            printf("-------------\n");
            printf("Numero do Quarto: %d \n",Quartos[i].codQuarto);
            printf("Tipo de Quarto: %s \n",Quartos[i].tipoQuarto);

        }
        else if ((strcmp(Quartos[i].ativo, "M") == 0) || (strcmp(Quartos[i].ativo, "m") == 0)){
            printf("Quarto para Manutencao \n");
        }
        else{
            printf("Quarto Ocupado \n");
        }
    printf("-------------\n");
    }
}
```

Figura 25 - Função listar Quarto

Nesta função podemos gravar todas as informações dos quarto num ficheiro chamado "quartos.txt".

```
void gravarfq(struct Quarto *Quartos, int n_quartos){
    FILE * f;

    int i;

    f = fopen ("quarto.txt","w");

    if (f == NULL){
        printf("Nao foi possivel abrir o ficheiro! \n");
    }
    else{
        fwrite(&n_quartos,sizeof(int),1,f);
        i=fwrite(Quartos, sizeof(struct Quarto),n_quartos,f);
        printf("Escreveu no ficheiro %d quarto(s). \n", i);
        fclose(f);
    }
}
```

Figura 26 - Função gravar ficheiro Quarto

A função carregarfq serve para carregar todas as informações do ficheiro "quartos.txt".

```
void carregarfq(struct Quarto **Quartos, int *n_quartos, int *n_maxQ){
    FILE * f;
    int i, nquartos;
    struct Quarto * quartoTemp;

    f = fopen ("quarto.txt", "r");

    if (f == NULL){
        printf("Nao foi possivel carregar o ficheiro! \n");
    }
    else{
        nquartos= fread (n_quartos,sizeof(int),1,f);
        if (nquartos == 0){
            printf("Erro na leitura do ficheiro! \n");
    }
        else{
            quartoTemp = (struct Quarto*) realloc (*Quartos,sizeof(struct Quarto) * (*n_quartos + MAX));

                if (quartoTemp==NULL){
                    printf("Erro ao carregar o ficheiro! \n");
                }
                else{
                    *Quartos = quartoTemp;
                    *n_maxQ = *n_quartos + MAX;
                    i = fread(*Quartos, sizeof(struct Quarto),*n_quartos,f);
                    printf(" %d Quartos carregados com sucesso. \n", i);
                    *n_quartos = i;
                    fclose(f);
                }
            }
    }
}
```

Figura 27 - Função carregar ficheiro Quarto

Na função editar quarto podemos editar as informações dos quartos sendo elas numero e o tipo do quarto.

```
void editarQuarto(struct Quarto *Quartos, int n_quartos){
    int i, numQuarto;

    printf("Introduza o numero do Quarto: ");
    scanf("%d",&numQuarto);

    i = verificarQuarto(Quartos, n_quartos, numQuarto);

    if (i != -1){
        printf("--------------\n");
        printf("Numero: ");
        scanf("%d",&Quartos[i].codQuarto);
        printf("Tipo de Quarto: ");
        scanf("%s",&Quartos[i].tipoQuarto);
        printf("--------------\n");

    }
}
```

Figura 28 - Função editar Quarto

Na ultima função nesta caso função indespoivelQuarto podemos alterar o estado dos quarto que pretendemos.

```
void indesponivelQuarto(struct Quarto *Quartos, int n_quartos){
    int i, numQuarto;
    char opcao[3], opcaoE[1];

    printf("Introduza o numero do Quarto: ");
    scanf("%d",&numQuarto);

    i = verificarQuarto(Quartos, n_quartos, numQuarto);

    if (i != -1){
        printf("Deseja por o quarto em Manutencao, Ocupado ou livre? Sim ou nao? ");
        scanf("%s", opcao);
            if ((strcmp(opcao, "Sim") == 0) || (strcmp(opcao, "sim") == 0)){
                printf("Escolha a opcao (M,O,L): ");
                scanf("%s", opcaoE);
                    if ((strcmp(opcaoE, "M") == 0) || (strcmp(opcaoE, "m") == 0)){
                        strcpy(&Quartos[i].ativo, "M");
                        printf("Alterado com sucesso \n");
                    }
                    else if ((strcmp(opcaoE, "O") == 0) || (strcmp(opcaoE, "o") == 0)){
                        strcpy(&Quartos[i].ativo, "O");
                        printf("Alterado com sucesso \n");
                    }
                    else if ((strcmp(opcaoE, "L") == 0) || (strcmp(opcaoE, "l") == 0)){
                        strcpy(&Quartos[i].ativo, "L");
                        printf("Alterado com sucesso \n");
                    }
            }
    }
}
```

Figura 29 - Função indisponível Quarto

Conclusão

Concluindo este trabalho, não foi possível realizar todas as tarefas propostas, devido a algumas dificuldades, mas foram concluídas a maior parte das funcionalidades propostas.

Foram feitas várias tarefas neste trabalho, de foram a realizar o objetivo pretendido, ou seja, a criação de um sistema de operações logísticas de uma farmácia.

Foi utilizado o Apache NetBeans para o desenvolvimento do código e o Doxygen para a documentação do projeto.

Ao longo do trabalho foram encontrados vários bloqueios, mas a principal sendo a memoria dinâmica e outras dificuldades que através de pesquisas foram resolvidas.