# Algoritmia e Programação

# Trabalho Prático Nº1

## ***Sistema de Gestão para Clínicas***


Daniel Ventura Brito, Nº 32749

Docentes Orientadores:

David Verde

João Filipe Correia Pereira


2024/2025


Cofinanciado por:

NORTE 2020 PROGRAMA OPERACIONAL REGIONAL DO NORTE

PORTUGAL 2020

UNIÃO EUROPEIA Fundo Social Europeu

# Índice

# 1 Introdução

Foi desenvolvido, em Linguagem C, um sistema de gestão para clínicas, com o objetivo de controlar e gerir os dados dos seus clientes, médicos e consultas da clínica. O trabalho prático foi realizado no âmbito da unidade curricular de Algoritmia e Programação, durante o ano letivo 2024/2025, visando consolidar os conhecimentos e aplicar os conceitos adquiridos durante as aulas lecionadas.

O sistema de gestão para clínicas, foi concebido para obedecer as necessidades básicas de gestão da clínica, permitindo ao utilizador uma gestão completa dos dados dos clientes, médicos e das consultas, como: Inserção de novos clientes, alteração de dados de clientes, inserção de novos médicos, obter uma lista dos médicos por especialidade, agendamento de consultas e histórico de consultas por clientes.

No presente relatório, iremos abordar ao detalhe as estruturas de dados implementadas, as funções específicas para evitar erros ou más intenções por parte dos utilizadores e a organização do código por módulos.

A aplicação divide-se em três secções, sendo a gestão de clientes, a gestão de médicos e a gestão de consultas. Na gestão de clientes, é possível gerir e manipular os dados dos clientes, incluído um conjunto de dados relevantes, tratando-se, o número de identificação do cliente (ID), nome do cliente, informações da localidade do cliente e o seu estado de atividade. A respeito da gestão de médicos, as funcionalidades são idênticas, contendo um menor conjunto de dados, sendo eles, o nome do médico, especialidade e identificação única. Por último, a gestão de consultas, abrangendo um número de identificação único, o nome do médico,

nome e número de identificação do cliente e ainda data e hora da consulta.

# 2 Estruturas de Dados

As implementações de estruturas de dados desempenham uma importante função na organização e manipulação de dados da aplicação desenvolvida para a clínica. De modo a atender aos requisitos proposto no enunciado, foram criadas três estruturas principais, sendo os clientes, médicos e consultas.

## 2.1 Estrutura para Clientes

A estrutura nomeada de 'ST_CLIENTES', é utilizada para armazenar os dados dos clientes. É composto por número de identificação único, atribuindo automaticamente por uma função, o nome do cliente e o e-mail, ambos limitados a 50 caracteres, a atribuição de uma data de nascimento do tipo de dado de uma outra estrutura designada de 'ST_DATA', a morada do cliente também do tipo de dado de uma outra estrutura 'ST_MORADA', ainda a declaração de tipos de dados inteiros como número de identificação fiscal (NIF), o número de utente (SNS), e ainda o estado de atividade do cliente atribuído por valor inteiros representado o estado, sendo o valor 0 como inativo e o valor 1 como ativo.

```
typedef struct {
int ID;
char nome[50];
ST_MORADA morada;
char email [50];
ST_DATA data_nascimento;
int NIF;
int SNS;
int estado;
}ST_CLIENTES;
```

**Tabela 1: Representação da estrutura clientes**

| Campo | Tipo | Descrição |
|---|---|---|
| ID | Inteiro | Identificação única do cliente. |
| nome | String | Nome do cliente. |
| morada | ST_MORADA | Conjunto de dados relevantes a morada. |
| data_de_nascimento | ST_DATA | Conjunto de valores inteiros relacionados com datas. |
| NIF | Inteiro | Identificação fiscal do cliente. |
| SNS | Inteiro | Número de utente do cliente. |
| estado | Inteiro | Valores inteiros representado o estado de atividade. |

## 2.2 Estrutura para Médicos

De igual forma, foi implementada uma estrutura designada 'ST_MEDICOS', concebida para armazenar informações sobre os profissionais de saúde da clínica. A estrutura é composta por os seguintes campos, um número de identificação único para cada profissional, o nome do médico, um número referente a identificação profissional do médico, uma especialidade, e ainda o indicador do estado de atividade do médico, atribuído com valores inteiros como o valor 0 sendo não disponível e o valor de 1 sendo disponível.

```
typedef struct {
int ID;
char nome[50];
int identificacao;
char especialidade[20];
int estado;
}ST_MEDICOS;
```

**Tabela 2: Representação da estrutura médicos**

| Campo | Tipo | Descrição |
|---|---|---|
| ID | Inteiro | Identificação única do médico. |
| nome | String | Nome do médico. |
| identificacao | Inteiro | Número de identificação profissional. |
| especialidade | String | Especifica a área do médico. |
| estado | Inteiro | Referentes ao estado de atividade. |

## 2.3 Estrutura para Consultas

A estrutura de dados para as consultas, responsável pelo agendamento e interações entre os médicos e os clientes. Cada consulta é composta por um número de identificação único para facilitar o rastreio, o horário que pertence à estrutura 'ST_DATA', utilizando para verificação de sobreposição de horas, um número de identificação e nome do cliente, o nome do profissional de saúde, e o indicador do estado da consulta, podendo marcar como agendada, realizada ou cancelada.

```
typedef struct {
int ID;
char medico[50];
char cliente[50];
int id_cliente;
int estado;
ST_DATA horario;
}ST_CONSULTAS;
```

**Tabela 3: Representação da estrutura consultas**

| Campo | Tipo | Descrição |
|---|---|---|
| ID | Inteiro | Identificação única da consulta. |
| medico | String | Nome do médico. |
| cliente | String | Nome do cliente. |
| id_cliente | inteiro | Número de identificação do cliente. |
| estado | Inteiro | Indicador do estado da consulta. |
| horario | ST_DATA | Armazena informações do dia, mês e hora da consulta. |

## 2.4 Estruturas Auxiliares

Adicionalmente, foram implementadas outras estruturas de modo a facilitar e organizar o conjunto de dados, garantindo uma eficiência na legibilidade no código.

### *2.4.1 Estrutura para Morada*

A criação de uma estrutura auxiliar para uma representação organizada de informações de morada associadas aos clientes. A estrutura nomeada de 'ST_MORADA' é composta por o nome da rua onde o cliente reside, o número da porta, o número de código postal e ainda o nome da cidade.

```
typedef struct{
char rua[50];
int numero;
int codigo_postal;
char cidade[25];
}ST_MORADA;
```

**Tabela 4: Representação da estrutura morada**

| Campo | Tipo | Descrição |
|---|---|---|
| rua | String | Nome da rua onde o cliente reside. |
| numero | Inteiro | Número da porta do cliente. |
| codigo_postal | Inteiro | Número referente ao código postal. |
| cidade | String | Nome da cidade do cliente. |

### *2.4.2 Estrutura para Data*

Da mesma forma, foi desenvolvida uma estrutura composta com informações relacionadas a data, como data de nascimento dos clientes ou data e hora das consultas. Foi designada de 'ST_DATA' e é constituída pelo dia, mês, ano e hora.

```
typedef struct{
int dia;
int mes;
int ano;
int hora;
}ST_DATA;
```

**Tabela 5: Representação da estrutura data**

| Campo | Tipo | Descrição |
|---|---|---|
| dia | Inteiro | Número do dia. |
| mes | Inteiro | Número do mês. |
| ano | Inteiro | Número do ano. |
| hora | Inteiro | Número da hora. |

# 3 Estruturação do Código

No começo, a aplicação foi desenvolvida utilizando um único ficheiro, mas devido ao elevado número de funções e da dimensão do projeto, foi definido a divisão do projeto por módulos, adotando a técnica de programação modular. Essa decisão foi motivada pela falta de legibilidade e compreensão em que o desenvolvimento do projeto estava a apresentar. A organização por módulos oferece inúmeras vantagens, principalmente em projetos de grupo, sendo então este projeto separado da seguinte forma:

**Tabela 6: Módulos do Projeto**

| Ficheiro | Descrição |
|---|---|
| main.c | Declaração dos vetores das estruturas e as variáveis. |
| cliente.c | Implementação das funções relacionadas com os clientes. |
| cliente.h | Protótipos das funções relacionadas com os clientes. |
| medico.c | Implementação das funções relacionadas com os médicos. |
| medico.h | Protótipos das funções relacionadas com os médicos. |
| consulta.c | Implementação das funções relacionadas com as consultas. |
| consulta.h | Protótipos das funções relacionadas com as consultas. |
| menu.c | Implementação das funções dos menus. |
| menu.h | Protótipos das funções dos menus. |
| auxiliares.c | Implementação das funções auxiliares. |
| auxiliares.h | Protótipos das funções auxiliares. |
| structs.h | Implementação das estruturas e constantes. |

## 3.1 Ficheiros headers (.h)

Os conhecidos ficheiros para declarar as bibliotecas, foram utilizados para declaração de estruturas de dados, constantes e os protótipos das funções.

### *3.1.1 structs.h*

O ficheiro ‘structs.h’ é composto pela declaração de todas as estruturas utilizadas, que representam os clientes, médicos e consultas. Foram ainda definidas no ficheiro, as macros do projeto, sendo utilizadas para definir valores constantes, como o número máximo de clientes, médicos e consultas. Ao invés de utilizar números diretamente, o uso de macros torna a utilização mais prática e organizada.

```
#ifndef STRUCTS_H
#define STRUCTS_H

// MACROS (CONSTANTES)
#define MAX_CLIENTES 1000
#define MAX_MEDICOS 100
#define MAX_CONSULTAS 5000

// STRUCTS
typedef struct{
int dia;
int mes;
int ano;
int hora;
}ST_DATA;

typedef struct{
char rua[50];
int numero;
int codigo_postal;
char cidade[25];
}ST_MORADA;

typedef struct {
int ID;
char nome[50];
ST_MORADA morada;
char email [50];
ST_DATA data_nascimento;
int NIF;
int SNS;
int estado;
}ST_CLIENTES;
```

```
typedef struct {
int ID;
char nome[50];
int identificacao;
char especialidade[20];
int estado;
}ST_MEDICOS;


typedef struct {
int ID;
char medico[50];
char cliente[50];
int id_cliente;
int estado;
ST_DATA horario;
}ST_CONSULTAS;

#endif
```

### *3.1.2 cliente.h*

Ficheiro header dos clientes, contém as declarações das funções relacionados com a gestão dos clientes da aplicação. O ficheiro serve para centralizar e organizar as funcionalidades específicas de gestão dos clientes, tornando fácil a manutenção do código.

```
#ifndef CLIENTES_H
#define CLIENTES_H

#include "structs.h"

void inserir_cliente(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
void alterar_dados_cliente(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
void ativar_desativar_cliente(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
void consultar_dados_cliente(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
void lista_clientes_ativos(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
```

```
        void clientes_nome(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
        int procurar_cliente(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
        int procurar_cliente_sns(ST_CLIENTES clientes[], int *n_clientes);
        int procurar_cliente_nif(ST_CLIENTES clientes[], int *n_clientes);
        int procurar_cliente_id(ST_CLIENTES clientes[], int *n_clientes);
        int procurar_cliente_nome(ST_CLIENTES clientes[], int *n_clientes);
        int atribuir_id_cliente(int *n_clientes);

        #endif
```

### *3.1.3 medico.h*

Na continuação dos ficheiros headers, o ficheiro 'medico.h' responsável pelas funções de gestão dos médicos da clínica, contém a declaração de todas as funções relacionadas com os profissionais.

```
        #ifndef MEDICO_H
        #define MEDICO_H

        #include "structs.h"


        void inserir_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
        int atribuir_id_medico(int *n_medicos);
        void alterar_dados_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
        int procurar_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
        int procurar_medico_id(ST_MEDICOS medicos[], int *n_medicos);
        int procurar_medico_nome(ST_MEDICOS medicos[], int *n_medicos);
        int procurar_medico_identificacao(ST_MEDICOS medicos[], int *n_medicos);
        int procurar_medico_especialidade(ST_MEDICOS medicos[], int *n_medicos);
        void ativar_desativar_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
        void consultar_dados_medicos(ST_CLIENTES clientes[], ST_MEDICOS
medicos[], ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int
*n_consultas);
        void lista_medicos(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
        void lista_medicos_disponivel(ST_CLIENTES clientes[], ST_MEDICOS
medicos[], ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int
*n_consultas);
```

```
    void lista_medicos_especialidade(ST_CLIENTES clientes[], ST_MEDICOS
medicos[], ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int
*n_consultas);
    void medicos_nome(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);


    #endif
```

### *3.1.4 consulta.h*

No projeto, o ficheiro 'consulta.h' é composto pela declaração das funções a respeito das consultas, mantendo a organização do código.

```
    #ifndef CONSULTA_H
    #define CONSULTA_H

    #include "structs.h"


    void agendar_consulta(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
    void desmarcar_consulta(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
    void marcar_consulta_realizada(ST_CLIENTES clientes[], ST_MEDICOS
medicos[], ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int
*n_consultas);
    void atualizar_consulta(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
    void listagem_dia_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
    void historico_consultas_clientes(ST_CLIENTES clientes[], ST_MEDICOS
medicos[], ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int
*n_consultas);
    void alterar_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
    void alterar_data(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
    int verificar_horario(ST_CONSULTAS consultas[], int *n_consultas, char
medico[], int dia, int mes, int hora);
    int verificar_medico(ST_MEDICOS medicos[], int *n_medicos, char
medico[50]);
    int verificar_cliente(ST_CLIENTES clientes[], int *n_clientes, char cliente[50], int
id_cliente);
    int atribuir_id_consulta(int *n_consultas);
```

```
#endif
```

### *3.1.5 menu.h*

O ficheiro ‘menu.h’ é composto pela definição do menu principal da aplicação, e ainda dos restantes menus de gestão dos clientes, médicos e consultas.

```
#ifndef MENU_H
#define MENU_H

#include "structs.h"


void menu_principal(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
void gestao_clientes(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
void gestao_medicos(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);
void gestao_consultas(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas);


#endif
```

### *3.1.6 auxiliares.h*

Por último o ficheiro header relacionado com as funções auxiliares utilizadas no projeto, composto pela declaração dessas funções, que desempenham papéis específicos. A implementação de funções como associar um código postal a uma cidade no distrito de Viana do Castelo, a obtenção dos valores respetivos a data e hora atuais do sistema, e a verificação da especialidade do médico fornecida.

```
#ifndef AUXILIARES_H
#define AUXILIARES_H

#include "structs.h"

int cp_cidade(int codigo_postal, char *cidade);
void data_atual(ST_DATA *data_hora_atual);
int verificar_especialidade(char *especialidade);
```

```
#endif
```

## 3.2 Ficheiros em C (.c)

Na organização do projeto, foram criados ficheiros de implementação, seguindo o princípio da programação por módulos. Cada ficheiro é formado pela implementação das suas respetivas funções, sendo a declaração realizada nos ficheiros headers .

### *3.2.1 main.c*

O ficheiro ‘main.c’ do projeto, é o ponto inicial do código, contém a função main, a primeira função executada. É respetivamente na função main que são invocadas outras funções para que o fluxo do programa funcione conforme planeado, é responsável por exibir o menu principal da gestão da clínica. Além, da declaração dos vetores das respetivas estruturas com um limite definido nas constantes, a declaração das variáveis responsáveis pela contagem do número atual de clientes, médicos e consultas, de modo a controlar o limite máximo predefinido no código.

```
// BIBLIOTECAS
#include <stdio.h>
#include "structs.h"
#include "menu.h"

// MAIN
int main(){
int n_clientes = 0, n_medicos = 0, n_consultas = 0;
ST_CLIENTES clientes[MAX_CLIENTES];
ST_MEDICOS medicos[MAX_MEDICOS];
ST_CONSULTAS consultas[MAX_CONSULTAS];
menu_principal(clientes, medicos, consultas, &n_clientes, &n_medicos,
&n_consultas);
return 0;
}
```

### *3.2.2 menu.c*

O ficheiro ‘menu.c’ contém a exibição do menu principal, e dos menus para gestão de clientes, médicos e consultas. Proporciona uma interação organizada e intuitiva com o utilizador, no menu principal o utilizador seleciona entre as opções de gestão de cliente, gestão de médicos ou gestão de consultas, entrando no menu selecionado. Cada menu de gestão permite o controlo de cada estrutura, invocando as suas funções específicas.

```
#include <stdio.h>
#include <string.h>
#include "menu.h"
#include "structs.h"
#include "cliente.h"
#include "medico.h"
#include "consulta.h"

// MENU PRINCIPAL
void menu_principal(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
int opcao;
printf("\n============================== CLÍNICA
==============================\n");
printf(" [1] - Gestão de Clientes\n");
printf(" [2] - Gestão de Médicos\n");
printf(" [3] - Gestão de Consultas\n");
printf(" [0] - Sair\n");
printf("\nDigita a opção: ");
scanf("%d", &opcao);
getchar();
printf("\n");
switch (opcao){
case 1:
gestao_clientes(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
break;
case 2:
gestao_medicos(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
break;
case 3:
gestao_consultas(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
break;
```

```
        case 0:
        printf("A encerrar programa");
        for (int i = 0; i < 5; i++){
        printf(".");
        }
        printf("\n");
        break;
        default:
        printf("A opção não é valida.\n");
        }
        }


        // MENU - GESTÃO DE CLIENTES
        void gestao_clientes(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int opcao;
        printf("\n============================== GESTÃO DE CLIENTES
==============================\n");
        printf(" [1] - Inserir novos clientes\n");
        printf(" [2] - Alterar dados de um cliente\n");
        printf(" [3] - Ativar ou desativar um cliente\n");
        printf(" [4] - Consultar dados de um cliente\n");
        printf(" [5] - Obter uma lista de clientes ativos\n");
        printf(" [6] - Procurar um cliente pelo nome\n");
        printf(" [0] - Sair\n");
        printf("\nDigita a opção: ");
        scanf("%d", &opcao);
        getchar();
        printf("\n");
        switch (opcao){
        case 1:
        inserir_cliente(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 2:
        alterar_dados_cliente(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 3:
        ativar_desativar_cliente(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 4:
        consultar_dados_cliente(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
```

```
        break;
        case 5:
        lista_clientes_ativos(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 6:
        clientes_nome(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 0:
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        default:
        printf("A opção não é valida.\n");
        }
        }


        // MENU - GESTÃO DE MEDICOS
        void gestao_medicos(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int opcao;
        printf("\n============================== GESTÃO DE MÉDICOS
==============================\n");
        printf(" [1] - Inserir novo médico\n");
        printf(" [2] - Alterar dados de um médico\n");
        printf(" [3] - Marcar um médico como disponível ou indisponível\n");
        printf(" [4] - Consultar dados de um médico\n");
        printf(" [5] - Obter lista de todos os médicos\n");
        printf(" [6] - Obter lista de médicos disponíveis\n");
        printf(" [7] - Obter lista de médicos por especialidade\n");
        printf(" [8] - Procurar um médico pelo nome\n");
        printf(" [0] - Sair\n");
        printf("\nDigita a opção: ");
        scanf("%d", &opcao);
        getchar();
        printf("\n");
        switch (opcao){
        case 1:
        inserir_medico(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 2:
        alterar_dados_medico(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
```

```
        break;
        case 3:
        ativar_desativar_medico(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 4:
        consultar_dados_medicos(clientes, medicos, consultas, n_clientes,
n_medicos, n_consultas);
        break;
        case 5:
        lista_medicos(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 6:
        lista_medicos_disponivel(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 7:
        lista_medicos_especialidade(clientes, medicos, consultas, n_clientes,
n_medicos, n_consultas);
        break;
        case 8:
        medicos_nome(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 0:
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        default:
        printf("A opção não é valida.\n");
        }
        }


        // MENU - GESTÃO DE CONSULTAS
        void gestao_consultas(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int opcao;
        printf("\n============================== GESTÃO DE CONSULTAS
==============================\n");
        printf(" [1] - Agendar uma consulta\n");
        printf(" [2] - Desmarcar uma consulta\n");
        printf(" [3] - Marcar consulta como realizada\n");
        printf(" [4] - Atualizar uma consulta\n");
        printf(" [5] - Obter lista de consultas para o dia atual por médico\n");
```

```
        printf(" [6] - Obter histórico de consultas por cliente\n");
        printf(" [0] - Sair\n");
        printf("\nDigita a opção: ");
        scanf("%d", &opcao);
        getchar();
        printf("\n");
        switch (opcao){
        case 1:
        agendar_consulta(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 2:
        desmarcar_consulta(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 3:
        marcar_consulta_realizada(clientes, medicos, consultas, n_clientes,
n_medicos, n_consultas);
        break;
        case 4:
        atualizar_consulta(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 5:
        listagem_dia_medico(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 6:
        historico_consultas_clientes(clientes, medicos, consultas, n_clientes,
n_medicos, n_consultas);
        break;
        case 0:
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        default:
        printf("A opção não é valida.\n");
        }
        }
```

### *3.2.3 cliente.c*

Este ficheiro é composto pelas funções responsáveis pela gestão dos clientes, as funções anteriormente declaradas no ficheiro header, são agora implementadas neste ficheiro.

A função ‘inserir_cliente’ é responsável por adicionar um novo cliente, faz a verificação se o número máximo de clientes foi atingido e permite a ‘inserção de dados’ por parte do utilizador.

A função ‘alterar_dados_cliente’ permite modificar dados de um cliente já inserido, o utilizador pode inserir o ID do cliente ou procurar o cliente por outras opções, é solicitado depois a inserção dos novos dados.

A função ‘ativar_desativar_cliente’ faz a alteração do estado de um cliente, procurando um cliente pelo ID ou por outras opções, depois é realizada a atualização do estado do cliente.

A função ‘consultar_dados_cliente’ faz a exibição das informações de um cliente específico, podendo ser procurado pelo ID ou por outras opções de pesquisa.

```
#include <stdio.h>
#include <string.h>
#include "cliente.h"
#include "menu.h"
#include "auxiliares.h"

void inserir_cliente(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
int NIF, SNS, dia, mes, ano, numero, opcao, codigo_postal, estado, id;
char nome[50], email[50], rua[50], cidade[25];


if (*n_clientes >= MAX_CLIENTES){
printf("Número máximo de clientes registados!\n");
menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
}else {
printf("============== INSERIR CLIENTE ==============\n");
id = atribuir_id_cliente(n_clientes);
printf("Número do indentificador do cliente: %d\n", id);
```

```
printf("\nNome do cliente: ");
fgets(nome, sizeof(nome), stdin);
nome[strcspn(nome, "\n")] = '\0';


printf("\nMorada do cliente:");
printf("\nRua: ");
fgets(rua, sizeof(rua), stdin);
rua[strcspn(rua, "\n")] = '\0';


printf("Número da porta: ");
scanf("%d", &numero);
getchar();


printf("Código Postal (xxxxxxx): ");
scanf("%d", &codigo_postal);
getchar();


if (cp_cidade(codigo_postal, cidade) == 0){
printf("Cidade: ");
fgets(cidade, sizeof(cidade), stdin);
cidade[strcspn(cidade, "\n")] = '\0';
} else {
printf("Cidade: %s\n", cidade);
}


printf("\nE-mail: ");
fgets(email, sizeof(email), stdin);
email[strcspn(email, "\n")] = '\0';


printf("\nData de Nascimento (dd-mm-aaaa): ");
scanf("%d-%d-%d", &dia, &mes, &ano);


printf("\nNIF: ");
scanf("%d", &NIF);


printf("\nSNS: ");
scanf("%d", &SNS);
getchar();


printf("\n");
printf("Estado do cliente:\n [0] - Inativo\n [1] - Ativo\n");
scanf("%d", &estado);
```

```
        getchar();


        printf("\n\nDeseja confirmar a inserção do cliente? 1 para SIM ou 0 para NÃO:
");
        scanf("%d", &opcao);
        getchar();
        printf("\n");
        if(opcao == 1){
        strcpy(clientes[*n_clientes].nome, nome);
        clientes[*n_clientes].ID = id;
        strcpy(clientes[*n_clientes].morada.rua, rua);
        clientes[*n_clientes].morada.codigo_postal = codigo_postal;
        strcpy(clientes[*n_clientes].morada.cidade, cidade);
        strcpy(clientes[*n_clientes].email, email);
        clientes[*n_clientes].data_nascimento.dia = dia;
        clientes[*n_clientes].data_nascimento.mes = mes;
        clientes[*n_clientes].data_nascimento.ano = ano;
        clientes[*n_clientes].NIF = NIF;
        clientes[*n_clientes].SNS = SNS;
        clientes[*n_clientes].estado = estado;
        printf("\nCliente inserido com sucesso.\n");
        (*n_clientes)++;
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else if (opcao == 0){
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        printf("\nA opção não é valida.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }
        }


        void alterar_dados_cliente(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int opcao, indice, NIF, SNS, dia, mes, ano, numero, codigo_postal, estado;
        char nome[50], email[50], rua[50], cidade[25];


        printf("============== ALTERAR DADOS DE UM CLIENTE
==============\n");
        printf("[1] - Inserir ID\n");
        printf("[2] - Procurar um cliente\n");
```

```
        printf("[0] - SAIR\n");
        printf("\nDigita a opção: ");
        scanf("%d", &opcao);
        getchar();
        printf("\n");


        switch(opcao){
        case 1:
        printf("Inserir ID do cliente: ");
        scanf("%d", &indice);
        indice = indice - 10000;
        getchar();
        break;
        case 2:
        indice = procurar_cliente(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 0:
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        default:
        break;
        }
        if (indice < 0){
        printf("ID não é válido.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        opcao = 0;


        printf("\nNome do cliente: ");
        fgets(nome, sizeof(nome), stdin);
        nome[strcspn(nome, "\n")] = '\0';


        printf("\nMorada do cliente:");
        printf("\nRua: ");
        fgets(rua, sizeof(rua), stdin);
        rua[strcspn(rua, "\n")] = '\0';


        printf("Número da porta: ");
        scanf("%d", &numero);
        getchar();


        printf("Código Postal (xxxxxxx): ");
```

```
        scanf("%d", &codigo_postal);
        getchar();

        if (cp_cidade(codigo_postal, cidade) == 0){
        printf("Cidade: ");
        fgets(cidade, sizeof(cidade), stdin);
        cidade[strcspn(cidade, "\n")] = '\0';
        } else {
        printf("Cidade: %s\n", cidade);
        }

        printf("\nE-mail: ");
        fgets(email, sizeof(email), stdin);
        email[strcspn(email, "\n")] = '\0';

        printf("\nData de Nascimento (dd-mm-aaaa): ");
        scanf("%d-%d-%d", &dia, &mes, &ano);

        printf("\nNIF: ");
        scanf("%d", &NIF);

        printf("\nSNS: ");
        scanf("%d", &SNS);
        getchar();

        printf("\nEstado do cliente:\n [0] - Inativo\n [1] - Ativo\n");
        scanf("%d", &estado);
        getchar();

        printf("\n\nDeseja confirmar a alteração do cliente? 1 para SIM ou 0 para NÃO:
");
        scanf("%d", &opcao);
        getchar();
        printf("\n");

        if(opcao == 1){
        strcpy(clientes[indice].nome, nome);
        strcpy(clientes[indice].morada.rua, rua);
        clientes[indice].morada.codigo_postal = codigo_postal;
        strcpy(clientes[indice].morada.cidade, cidade);
        strcpy(clientes[indice].email, email);
        clientes[indice].data_nascimento.dia = dia;
        clientes[indice].data_nascimento.mes = mes;
```

```
        clientes[indice].data_nascimento.ano = ano;
        clientes[indice].NIF = NIF;
        clientes[indice].SNS = SNS;
        clientes[indice].estado = estado;
        printf("\nDados alterados com sucesso.");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else if (opcao == 0){
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        printf("\nA opção não é valida.");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }
        }


        void ativar_desativar_cliente(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int opcao, indice;
        printf("============== ATIVAR OU DESATIVAR CLIENTE ==============\n");
        printf("[1] - Inserir ID\n");
        printf("[2] - Procurar um cliente\n");
        printf("[0] - SAIR\n");
        printf("\nDigita a opção: ");
        scanf("%d", &opcao);
        getchar();


        switch(opcao){
        case 1:
        printf("Inserir ID do cliente: ");
        scanf("%d", &indice);
        printf("\n");
        indice = indice - 10000;
        getchar();
        break;
        case 2:
        indice = procurar_cliente(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 0:
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        default:
```

```
        break;
        }
        if (indice < 0){
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        opcao = 0;
        printf("Nome: %s\n", clientes[indice].nome);
        printf("ID: %d\n", clientes[indice].ID);
        if (clientes[indice].estado == 0){
        printf("Estado: Inativo\n");
        }else if(clientes[indice].estado == 1){
        printf("Estado: Ativo\n");
        }
        printf("\nDeseja alterar o Estado do cliente? 1 para SIM e 0 para NÃO: ");
        scanf("%d", &opcao);
        getchar();
        if (opcao == 0){
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else if (opcao == 1){
        printf("\nEstado do cliente:\n [0] - Inativo\n [1] - Ativo\n");
        scanf("%d", &clientes[indice].estado);
        getchar();
        printf("Cliente alterado com sucesso.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        printf("Opção não é válida.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }
        }


        void consultar_dados_cliente(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int opcao, indice;
        printf("============== CONSULTAR DADOS DE UM CLIENTE
==============\n");
        printf("[1] - Inserir ID\n");
        printf("[2] - Procurar um cliente\n");
        printf("[0] - SAIR\n");
        printf("\nDigita a opção: ");
        scanf("%d", &opcao);
```

```
        getchar();
        switch(opcao){
        case 1:
        printf("Inserir ID do cliente: ");
        scanf("%d", &indice);
        indice = indice - 10000;
        getchar();
        break;
        case 2:
        indice = procurar_cliente(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 0:
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        default:
        break;
        }
        if (indice < 0){
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        opcao = 0;
        printf("Nome: %s\n", clientes[indice].nome);
        printf("ID: %d\n", clientes[indice].ID);
        printf("\nMorada:\n");
        printf("Rua: %s\n", clientes[indice].morada.rua);
        printf("Código Postal: %d\n", clientes[indice].morada.codigo_postal);
        printf("Cidade: %s\n", clientes[indice].morada.cidade);
        printf("E-Mail: %s\n", clientes[indice].email);
        printf("Data de Nascimente : %d-%d-%d\n",
clientes[indice].data_nascimento.dia, clientes[indice].data_nascimento.mes,
clientes[indice].data_nascimento.ano);
        printf("NIF: %d\n", clientes[indice].NIF);
        printf("SNS: %d\n", clientes[indice].SNS);
        if (clientes[indice].estado == 0){
        printf("Estado: Inativo\n");
        }else if (clientes[indice].estado == 1){
        printf("Estado: Ativo\n");
        }
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }
```

```
	void lista_clientes_ativos(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
	printf("============== LISTA DE CLIENTES ATIVOS ==============\n");
	for (int i = 0; i < *n_clientes; i++){
	if (clientes[i].estado == 1){
	printf("\nID: %d - NOME: %s\n", clientes[i].ID, clientes[i].nome);
	}
	}
	menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
	}


	void clientes_nome(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
	int encontrados = 0, opcao;
	char string[50];
	printf("============== PROCURAR UM CLIENTE ==============\n");
	printf("Introduzir nome: ");
	fgets(string, sizeof(string), stdin);
	string[strcspn(string, "\n")] = '\0';


	for (int i = 0; i < *n_clientes; i++){
	if(strcspn(clientes[i].nome, string) == 0){
	encontrados++;
	printf("Cliente encontrado %d:\n ID: %d - NOME: %s\n\n", encontrados,
clientes[i].ID, clientes[i].nome);
	}
	}
	if (encontrados == 0){
	printf("Nenhum cliente encontrado com o nome.\n");
	}else if (encontrados > 0){
	printf("Foram encontrados %d clientes com esse nome.\n", encontrados);
	}
	printf("Deseja realizar uma nova procura? 1 para SIM e 0 para NÃO: ");
	scanf("%d", &opcao);
	getchar();
	if (opcao == 1){
	clientes_nome(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
	}else if (opcao == 0){
	menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
	}else {
	printf("Opção não é válida.\n");
```

```
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }


        int procurar_cliente(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int opcao, indice;
        printf("============== PROCURAR UM CLIENTE ==============\n");
        printf("[1] - Procurar por ID\n");
        printf("[2] - Procurar por Nome\n");
        printf("[3] - Procurar por NIF\n");
        printf("[4] - Procurar por SNS\n");
        printf("[0] - SAIR\n");
        printf("\nDigita a opção: ");
        scanf("%d", &opcao);
        getchar();
        printf("\n");


        switch(opcao){
        case 1:
        return indice = procurar_cliente_id(clientes, n_clientes);
        break;
        case 2:
        return indice = procurar_cliente_nome(clientes, n_clientes);
        break;
        case 3:
        return indice = procurar_cliente_nif(clientes, n_clientes);
        break;
        case 4:
        return indice = procurar_cliente_sns(clientes, n_clientes);
        break;
        case 0:
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        default:
        break;
        }
        }


        int procurar_cliente_sns(ST_CLIENTES clientes[], int *n_clientes){
        int sns, id, encontrados = 0;
        printf("============== PROCURAR CLIENTE POR SNS ==============\n");
        printf("Inserir SNS do cliente: ");
```

```
        scanf("%d", &sns);
        getchar();


        for (int i = 0; i < *n_clientes; i++){
        if(clientes[i].SNS == sns){
        encontrados++;
        printf("Cliente Encontrado: ID: %d - NOME: %s\n", clientes[i].ID,
clientes[i].nome);
        }
        }
        if(encontrados == 0){
        printf("Nenhum cliente encontrado com o SNS.\n");
        return -1;
        }else {
        printf("Selecionar cliente encontrado por ID: ");
        scanf("%d", &id);
        getchar();
        return (id-10000);
        }
        }



        int procurar_cliente_nif(ST_CLIENTES clientes[], int *n_clientes){
        int nif, id, encontrados = 0;
        printf("============== PROCURAR CLIENTE POR NIF ==============\n");
        printf("Inserir NIF do cliente: ");
        scanf("%d", &nif);
        getchar();


        for (int i = 0; i < *n_clientes; i++){
        if (clientes[i].NIF == nif){
        encontrados++;
        printf("Cliente Encontrado: ID: %d - NOME: %s\n", clientes[i].ID,
clientes[i].nome);
        }
        }
        if(encontrados == 0){
        printf("Nenhum cliente encontrado com o NIF.\n");
        return -1;
        }else {
        printf("Selecionar cliente encontrado por ID: ");
        scanf("%d", &id);
        getchar();


        return (id-10000);
```

```
        }
        }


        int procurar_cliente_id(ST_CLIENTES clientes[], int *n_clientes){
        int id, encontrados = 0;
        printf("============== PROCURAR CLIENTE POR ID ==============\n");
        printf("Inserir ID do cliente: ");
        scanf("%d", &id);
        getchar();
        for (int i = 0; i < *n_clientes; i++){
        if (clientes[i].ID == id){
        encontrados++;
        printf("Cliente Encontrado: ID: %d - NOME: %s\n", clientes[i].ID,
clientes[i].nome);
        }
        }
        if (encontrados == 0){
        printf("Nenhum cliente encontrado com o ID.\n");
        return -1;
        }else {
        return (id-10000);
        }
        }


        int procurar_cliente_nome(ST_CLIENTES clientes[], int *n_clientes){
        char nome[50];
        int encontrados = 0;
        int id;


        printf("============== PROCURAR CLIENTE POR NOME ==============\n");
        printf("Inserir nome do cliente: ");
        fgets(nome, sizeof(nome), stdin);
        nome[strcspn(nome, "\n")] = '\0';
        for (int i = 0; i < *n_clientes; i++){
        if (strcmp(clientes[i].nome, nome) == 0){
        encontrados++;
        printf("\nCliente Encontrado %d: ID: %d - NOME: %s\n", encontrados,
clientes[i].ID, clientes[i].nome);
        }
        }
        if (encontrados == 0){
        printf("Nenhum cliente encontrado com o nome %s. \n", nome);
        return -1;
        }else {
```

```
    printf("Selecionar cliente encontrado por ID: ");
    scanf("%d", &id);
    getchar();


    return (id-10000);
    }
    }


    int atribuir_id_cliente(int *n_clientes){
    int id = 10000 + (*n_clientes);
    return id;
    }
```

### *3.2.4 medico.c*

O ficheiro ‘medico.c’ é responsável pela gestão dos médicos, funções como inserção de novos médicos, alteração de dados de médicos, marcação de disponibilidade e consulta de informação referente aos médicos.

A função ‘inserir_medico’ faz a inserção de novos médicos, verifica o valor na variável ‘n_medicos’ e solicita dados do novo médico.

A função ‘atribuir_id_medico’ gera um ID único para o médico, tem como base o número atual de médicos inseridos no sistema, sendo incrementado a medida que novos médicos são inseridos.

A função ‘consultar_dados_medicos’ permite a consulta dos dados de um medico com base no ID ou após uma procura do médico por outras opções.

A função ‘ativar_desativar_medico’ permite a alteração do estado de um médico, sendo disponível ou não disponível.

```
    #include <stdio.h>
    #include <string.h>
    #include "medico.h"
    #include "menu.h"
    #include "auxiliares.h"

    void inserir_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
```

```
        int id, identificacao, estado, opcao;
        char nome[50], especialidade[20];

        if (*n_medicos >= MAX_MEDICOS){
        printf("Número máximo de médicos registados!\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        printf("============== INSERIR NOVOS MÉDICOS ==============\n");
        id = atribuir_id_medico(n_medicos);
        printf("Número do indentificador do médico: %d\n", id);

        printf("\nNome do médico: ");
        fgets(nome, sizeof(nome), stdin);
        nome[strcspn(nome, "\n")] = '\0';
        printf("\n");

        printf("Identificação profissional: ");
        scanf("%d", &identificacao);
        getchar();
        printf("\n");

        printf("Especialidade:\n Cardiologia - Dermatologia - Ortopedia - Pediatria\n
Psiquiatria - Radiologia - Neurologia - Urologia\n");
        fgets(especialidade, sizeof(especialidade), stdin);
        especialidade[strcspn(especialidade, "\n")] = '\0';
        if(verificar_especialidade(especialidade) == 0){
        printf("Especialidade inválida.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        printf("\n");

        printf("Estado do médico:\n[0] - Não Disponível\n[1] - Disponível\n");
        scanf("%d", &estado);
        getchar();

        printf("\n\nDeseja confirmar a inserção do médico? 1 para SIM ou 0 para NÃO:
");
        scanf("%d", &opcao);
        getchar();
        printf("\n");
```

```
        if(opcao == 1){
        strcpy(medicos[*n_medicos].nome, nome);
        medicos[*n_medicos].ID = id;
        medicos[*n_medicos].identificacao = identificacao;
        strcpy(medicos[*n_medicos].especialidade, especialidade);
        medicos[*n_medicos].estado = estado;
        printf("\nMédico inserido com sucesso.\n");
        (*n_medicos)++;
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else if (opcao == 0){
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        printf("\nA opção não é valida.");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }
        }



        int atribuir_id_medico(int *n_medicos){
        int id = 1000 + (*n_medicos);
        return id;
        }



        void alterar_dados_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int opcao, indice, identificacao, estado;
        char nome[50], especialidade[20];
        printf("============================== ALTERAR DADOS DE UM MÉDICO
==============================\n");
        printf("[1] - Inserir Identificador do Médico\n");
        printf("[2] - Procurar um Médico\n");
        printf("[0] - SAIR\n");
        printf("\nDigita a opção: ");
        scanf("%d", &opcao);
        getchar();
        printf("\n");


        switch(opcao){
        case 1:
        printf("Inserir Identificador do médico: ");
        scanf("%d", &indice);
```

```
        indice = indice - 1000;
        getchar();
        break;
        case 2:
        indice = procurar_medico(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 0:
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        default:
        break;
        }
        if (indice < 0){
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        opcao = 0;


        printf("\nNome do médico: ");
        fgets(nome, sizeof(nome), stdin);
        nome[strcspn(nome, "\n")] = '\0';


        printf("\nIdentificação profissional: ");
        scanf("%d", &identificacao);
        getchar();


        printf("Especialidade:\n Cardiologia - Dermatologia - Ortopedia - Pediatria\n
Psiquiatria - Radiologia - Neurologia - Urologia\n");
        fgets(especialidade, sizeof(especialidade), stdin);
        especialidade[strcspn(especialidade, "\n")] = '\0';
        if(verificar_especialidade(especialidade) == 0){
        printf("Especialidade inválida.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        printf("\n");


        printf("Estado do médico:\n [0] - Não Disponível\n [1] - Disponível\n");
        scanf("%d", &estado);
        getchar();


        printf("\n\nDeseja confirmar a alteração do médico? 1 para SIM ou 0 para
NÃO: ");
```

```
        scanf("%d", &opcao);
        getchar();
        printf("\n");


        if(opcao == 1){
        strcpy(medicos[indice].nome, nome);
        medicos[indice].identificacao = identificacao;
        strcpy(medicos[indice].especialidade, especialidade);
        medicos[indice].estado = estado;
        printf("\nDados alterados com sucesso.");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else if (opcao == 0){
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        printf("\nA opção não é valida.");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }
        }



        void ativar_desativar_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int opcao, indice;
        printf("============================== ALTERAR DISPONIBILIDADE DE
MÉDICO ==============================\n");
        printf("[1] - Inserir ID\n");
        printf("[2] - Procurar um médico\n");
        printf("[0] - SAIR\n");
        printf("\nDigita a opção: ");
        scanf("%d", &opcao);
        getchar();


        switch(opcao){
        case 1:
        printf("Inserir ID do médico: ");
        scanf("%d", &indice);
        indice = indice - 1000;
        getchar();
        break;
        case 2:
```

```
        indice = procurar_medico(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 0:
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        default:
        break;
        }
        if (indice < 0){
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        opcao = 0;
        printf("Nome: %s\n", medicos[indice].nome);
        printf("ID: %d\n", medicos[indice].ID);
        if (medicos[indice].estado == 0){
        printf("Estado: Inativo\n");
        }else if(medicos[indice].estado == 1){
        printf("Estado: Ativo\n");
        }
        printf("\nDeseja alterar disponibilidade do médico? 1 para SIM e 0 para NÃO:
");
        scanf("%d", &opcao);
        getchar();
        if (opcao == 0){
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else if (opcao == 1){
        printf("\nEstado do médico:\n [0] - Indisponível\n [1] - Disponibilidade\n");
        scanf("%d", &medicos[indice].estado);
        getchar();
        printf("Estado do médico alterado com sucesso.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        printf("Opção não é válida.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }
        }
```

```
	void consultar_dados_medicos(ST_CLIENTES clientes[], ST_MEDICOS
medicos[], ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int
*n_consultas){
	int opcao, indice;
	printf("============================== CONSULTAR DADOS DE UM
MÉDICO ==============================\n");
	printf("[1] - Inserir ID\n");
	printf("[2] - Procurar um médico\n");
	printf("[0] - SAIR\n");
	printf("\nDigita a opção: ");
	scanf("%d", &opcao);
	getchar();
	switch(opcao){
	case 1:
	printf("Inserir ID do médico: ");
	scanf("%d", &indice);
	indice = indice - 1000;
	getchar();
	break;
	case 2:
	indice = procurar_medico(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
	break;
	case 0:
	menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
	default:
	break;
	}
	if (indice < 0){
	menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
	}else {
	opcao = 0;
	printf("Nome: %s\n", medicos[indice].nome);
	printf("ID: %d\n", medicos[indice].ID);
	printf("Especialidade: %s\n", medicos[indice].especialidade);
	printf("Identificação profissional: %d\n", medicos[indice].identificacao);
	if (medicos[indice].estado == 0){
	printf("Estado: Indisponível\n");
	}else if (medicos[indice].estado == 1){
	printf("Estado: Disponível\n");
	}
	menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
	}
	}
```

```
	void lista_medicos(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
	printf("============================== LISTA DE MÉDICOS
==============================\n");
	for (int i = 0; i < *n_medicos; i++){
	printf("\nID: %d - NOME: %s - ESPECIALIDADE: %s\n", medicos[i].ID,
medicos[i].nome, medicos[i].especialidade);
	}
	menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
	}


	void lista_medicos_disponivel(ST_CLIENTES clientes[], ST_MEDICOS
medicos[], ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int
*n_consultas){
	printf("============================== LISTA DE MÉDICOS DISPONÍVEIS
==============================\n");
	for (int i = 0; i < *n_medicos; i++){
	if (medicos[i].estado == 1){
	printf("\nID: %d - NOME: %s - ESPECIALIDADE: %s\n", medicos[i].ID,
medicos[i].nome, medicos[i].especialidade);
	}
	}
	menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
	}


	void lista_medicos_especialidade(ST_CLIENTES clientes[], ST_MEDICOS
medicos[], ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int
*n_consultas){
	char especialidade[20];
	printf("============================== LISTA DE MÉDICOS POR
ESPECIALIDADE ==============================\n");
	printf("Inserir especialidade de médico: ");
	fgets(especialidade, sizeof(especialidade), stdin);
	especialidade[strcspn(especialidade, "\n")] = '\0';
	for (int i = 0; i < *n_medicos; i++){
	if (strcspn(medicos[i].especialidade, especialidade) == 0){
	printf("\nID: %d - NOME: %s - ESPECIALIDADE: %s\n", medicos[i].ID,
medicos[i].nome, medicos[i].especialidade);
	}
	}
```

```
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }


        void medicos_nome(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int encontrados = 0, opcao;
        char string[50];
        printf("============================== PROCURAR UM MÉDICO
==============================\n");
        printf("Introduzir nome: ");
        fgets(string, sizeof(string), stdin);
        string[strcspn(string, "\n")] = '\0';


        for (int i = 0; i < *n_medicos; i++){
        if(strcspn(medicos[i].nome, string) == 0){
        encontrados++;
        printf("Médico encontrado %d:\n ID: %d - NOME: %s -
ESPECIALIDADE: %s\n\n", encontrados, medicos[i].ID, medicos[i].nome,
medicos[i].especialidade);
        }
        }
        if (encontrados == 0){
        printf("Nenhum médico encontrado com o nome.\n");
        }else if (encontrados > 0){
        printf("Foram encontrados %d médicos com esse nome.\n\n", encontrados);
        }
        printf("Deseja realizar uma nova procura? 1 para SIM e 0 para NÃO: ");
        scanf("%d", &opcao);
        getchar();
        if (opcao == 1){
        medicos_nome(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else if (opcao == 0){
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        printf("Opção não é válida.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }
```

```
	int procurar_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
	int opcao, indice;
	printf("============================== PROCURAR UM MÉDICO
==============================\n");
	printf("[1] - Procurar por ID\n");
	printf("[2] - Procurar por Nome\n");
	printf("[3] - Procurar por Identificação Profissional\n");
	printf("[4] - Procurar por Especialidade\n");
	printf("[0] - SAIR\n");
	printf("\nDigita a opção: ");
	scanf("%d", &opcao);
	getchar();
	printf("\n");


	switch(opcao){
	case 1:
	indice = procurar_medico_id(medicos, n_medicos);
	break;
	case 2:
	indice = procurar_medico_nome(medicos, n_medicos);
	break;
	case 3:
	indice = procurar_medico_identificacao(medicos, n_medicos);
	break;
	case 4:
	indice = procurar_medico_especialidade(medicos, n_medicos);
	break;
	case 0:
	menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
	default:
	break;
	}
	return indice;
	}



	int procurar_medico_id(ST_MEDICOS medicos[], int *n_medicos){
	int id, encontrados = 0;
	printf("============== PROCURAR MÉDICO POR ID ==============\n");
	printf("Inserir ID do médico: ");
	scanf("%d", &id);
	getchar();
	for (int i = 0; i < *n_medicos; i++){
	if (medicos[i].ID == id){
```

```
        encontrados++;
        printf("Médico Encontrado: ID: %d - NOME: %s\n", medicos[i].ID,
medicos[i].nome);
        }
        }
        if (encontrados == 0){
        printf("Nenhum médico encontrado com o ID.\n");
        return -1;
        }else {
        return (id-1000);
        }
        }


        int procurar_medico_nome(ST_MEDICOS medicos[], int *n_medicos){
        char nome[50];
        int encontrados = 0;
        int id;

        printf("============================== PROCURAR MÉDICOS POR NOME
==============================\n");
        printf("Inserir nome do médico: ");
        fgets(nome, sizeof(nome), stdin);
        nome[strcspn(nome, "\n")] = '\0';
        for (int i = 0; i < *n_medicos; i++){
        if (strcmp(medicos[i].nome, nome) == 0){
        encontrados++;
        printf("\nMédico Encontrado %d: ID: %d - NOME: %s\n", encontrados,
medicos[i].ID, medicos[i].nome);
        }
        }
        if (encontrados == 0){
        printf("Nenhum médico encontrado com o nome %s. \n", nome);
        return -1;
        }else {
        printf("Selecionar médico encontrado por ID: ");
        scanf("%d", &id);
        getchar();

        return (id-1000);
        }
        }


        int procurar_medico_identificacao(ST_MEDICOS medicos[], int *n_medicos){
```

```
        int identificacao, encontrados = 0;
        int id;
        printf("============================== PROCURAR MÉDICO POR
IDENTIFICAÇÃO PROFISSIONAL ==============================\n");
        printf("Inserir Identificação Profissional do médico: ");
        scanf("%d", &identificacao);
        getchar();
        for (int i = 0; i < *n_medicos; i++){
        if (medicos[i].identificacao == identificacao){
        encontrados++;
        printf("Médico Encontrado: ID: %d - NOME: %s\n", medicos[i].ID,
medicos[i].nome);
        }
        }
        if (encontrados == 0){
        printf("Nenhum médico encontrado com a Identificação.\n");
        return -1;
        }else {
        printf("Selecionar médico encontrado por ID: ");
        scanf("%d", &id);
        getchar();


        return(id-1000);
        }
        }


        int procurar_medico_especialidade(ST_MEDICOS medicos[], int *n_medicos){
        char especialidade[20];
        int encontrados = 0;
        int id;


        printf("============================== PROCURAR MÉDICOS POR
ESPECIALIDADE ==============================\n");
        printf("Inserir especialidade do médico: ");
        fgets(especialidade, sizeof(especialidade), stdin);
        especialidade[strcspn(especialidade, "\n")] = '\0';
        for (int i = 0; i < *n_medicos; i++){
        if (strcmp(medicos[i].especialidade, especialidade) == 0){
        encontrados++;
        printf("\nMédico Encontrado %d: ID: %d - NOME: %s\n", encontrados,
medicos[i].ID, medicos[i].nome);
        }
        }
        if (encontrados == 0){
```

```
        printf("Nenhum médico encontrado com a especialidade %s. \n",
especialidade);
        return -1;
        }else {
        printf("Selecionar médico encontrado por ID: ");
        scanf("%d", &id);
        getchar();


        return (id-1000);
        }
        }
```

### *3.2.5 consulta.c*

O ficheiro 'consulta.c' contém a implementação das funções para agendamento, alteração e cancelamento de consultas, além de outras funções relacionados com o histórico de consultas. Foram ainda implementadas algumas funções de verificação de horários, de modo, a controlar a sobreposição de consultas em médicos e clientes, além da implementação do horário de funcionamento da clínica, garantido o agendamento de consultas apenas entre as 8h às 18h.

A função 'agendar_consulta' realiza o agendamento de uma consulta para um cliente com um médico e com o horário específicos, verifica o número máximo de consultas, se o médico e o cliente existem e se o horário está disponível.

A função 'desmarcar_consulta' altera o estado de uma consulta já existente, alterado o seu estado para cancelada (-1).

A função 'atualizar_consulta' permite atualizar uma consulta já agendada, com as opções de alterar a data ou o médico.

A função 'listagem_dia_medico' faz a verificação do dia atual e exibe as consultas para o dia.

```
        #include <stdio.h>
        #include <string.h>
        #include "consulta.h"
        #include "menu.h"
        #include "auxiliares.h"

        void agendar_consulta(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int opcao, id, id_cliente, dia, mes, hora;
```

```
        char medico[50], cliente[50];
        if (*n_consultas >= MAX_CONSULTAS){
        printf("Número máximo de consultas registadas!\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        printf("============================== AGENDAR UMA CONSULTA
==============================\n");
        id = atribuir_id_consulta(n_consultas);
        printf("\nNúmero de indentificação da consulta : %d\n", id);


        printf("\nNome do médico: ");
        fgets(medico, sizeof(medico), stdin);
        medico[strcspn(medico, "\n")] = '\0';
        printf("\n");
        opcao = verificar_medico(medicos, n_medicos, medico);
        if (opcao == -1){
        printf("\nNenhum médico encontrado.\n");
        agendar_consulta(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        printf("Nome do cliente: ");
        fgets(cliente, sizeof(cliente), stdin);
        cliente[strcspn(cliente, "\n")] = '\0';
        printf("\n");


        printf("ID do cliente: ");
        scanf("%d", &id_cliente);
        getchar();
        if (verificar_cliente(clientes, n_clientes, cliente, id_cliente) == -1){
        printf("Nenhum cliente encontrado.\n");
        agendar_consulta(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }


        printf("\nHorário:\n");
        printf("Dia: ");
        scanf("%d", &dia);
        printf("Mês: ");
        scanf("%d", &mes);
        printf("Hora: ");
        scanf("%d", &hora);
        getchar();
        opcao = verificar_horario(consultas, n_consultas, medico, dia, mes, hora);
        if (opcao == -1){
```

```
        printf("Horário não disponível.\n");
        printf("Deseja inserir novo horário? 1 para SIM ou 0 para NÃO: ");
        scanf("%d", &opcao);
        getchar();
        if (opcao == 1){
        agendar_consulta(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else{
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }else if (opcao == 1){
        printf("\n\nDeseja confirmar o agendamento da consulta? 1 para SIM ou 0
para NÃO: ");
        scanf("%d", &opcao);
        getchar();
        printf("\n");
        if (opcao == 1){
        strcpy(consultas[*n_consultas].medico, medico);
        consultas[*n_consultas].ID = id;
        strcpy(consultas[*n_consultas].cliente, cliente);
        consultas[*n_consultas].id_cliente = id_cliente;
        consultas[*n_consultas].horario.dia = dia;
        consultas[*n_consultas].horario.mes = mes;
        consultas[*n_consultas].horario.hora = hora;
        consultas[*n_consultas].estado = 1;
        printf("\nConsulta agendada com sucesso:\nCliente: %s - Médico: %s -
Horário: %d/%d - %d:00h até às %d:00h\n", consultas[*n_consultas].cliente,
consultas[*n_consultas].medico, consultas[*n_consultas].horario.mes,
consultas[*n_consultas].horario.dia, consultas[*n_consultas].horario.hora,
consultas[*n_consultas].horario.hora+1);
        (*n_consultas)++;
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }
        }
        }


        void desmarcar_consulta(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int id, indice;
        printf("============================== DESMARCAR UMA CONSULTA
==============================\n");
        printf("Número de indentificação da consulta: ");
```

```
        scanf("%d", &id);
        getchar();
        indice = id - 10000;
        if (indice >= 0){
        consultas[indice].estado = -1; // 1 : Agendada; 0 : Realizada; -1 : Cancelada
        printf("Consulta cancelada com sucesso.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        printf("Consulta não encontrada.\n");
        desmarcar_consulta(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }


        void marcar_consulta_realizada(ST_CLIENTES clientes[], ST_MEDICOS
medicos[], ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int
*n_consultas){
        int id, indice;
        printf("============================== MARCAR CONSULTA COMO
REALIZADA ==============================\n");
        printf("Número de indentificação da consulta: ");
        scanf("%d", &id);
        getchar();
        indice = id - 10000;
        if (indice >= 0){
        consultas[indice].estado = 0; // 1 : Agendada; 0 : Realizada; -1 : Cancelada
        printf("Consulta foi marcada como realizada.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }else {
        printf("Consulta não encontrada.\n");
        desmarcar_consulta(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        }


        void atualizar_consulta(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int opcao;
        printf("============================== ATUALIZAR CONSULTA
==============================\n");
        printf("[1] - Alterar médico\n");
        printf("[2] - Alterar data\n");
```

```
        printf("[0] - SAIR\n");
        printf("\nDigita a opção: ");
        scanf("%d", &opcao);
        switch(opcao){
        case 1:
        alterar_medico(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        case 2:
        alterar_data(clientes, medicos, consultas, n_clientes, n_medicos, n_consultas);
        break;
        case 0:
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        default:
        printf("Opção não é valida.\n");
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        break;
        }
        }


        void alterar_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int id, indice, opcao;
        char medico[50];
        printf("Número de indentificação da consulta: ");
        scanf("%d", &id);
        getchar();


        indice = id - 10000;
        if (indice >= 0){
        printf("Nome do novo médico: ");
        fgets(medico, sizeof(medico), stdin);
        medico[strcspn(medico, "\n")] = '\0';
        printf("\n");
        opcao = verificar_medico(medicos, n_medicos, medico);
        if (opcao == -1){
        printf("Nenhum médico encontrado.\n");
        alterar_medico(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        strcpy(consultas[*n_consultas].medico, medico);
```

```
        printf("\nMédico alterado com sucesso:\nCliente: %s - Médico: %s -
Horário: %d/%d - %d:00h até às %d:00h\n", consultas[indice].cliente,
consultas[indice].medico, consultas[indice].horario.mes,
consultas[indice].horario.dia, consultas[indice].horario.hora,
consultas[indice].horario.hora+1);
        }else {
        printf("Consulta não encontrada.\n");
        alterar_medico(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }


        void alterar_data(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        int id, indice, opcao, dia, mes, hora;
        char medico[50];
        printf("Número de indentificação da consulta: ");
        scanf("%d", &id);


        indice = id - 10000;
        if (indice >= 0){
        strcpy(medico, consultas[indice].medico);
        printf("Médico: %s\n", medico);
        printf("Horário:\n");
        printf("Dia: ");
        scanf("%d", &dia);
        printf("Mês: ");
        scanf("%d", &mes);
        printf("\nHora: ");
        scanf("%d", &hora);
        getchar();
        opcao = verificar_horario(consultas, n_consultas, medico, dia, mes, hora);
        if (opcao == -1){
        printf("Horário não disponível.\n");
        printf("Deseja inserir novo horário? 1 para SIM ou 0 para NÃO: ");
        scanf("%d", &opcao);
        getchar();
        if (opcao == 1){
        alterar_data(clientes, medicos, consultas, n_clientes, n_medicos, n_consultas);
        }else{
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
```

```
        }else if (opcao == 1){
        consultas[indice].horario.dia = dia;
        consultas[indice].horario.mes = mes;
        consultas[indice].horario.hora = hora;
        printf("\nData alterada com sucesso:\nCliente: %s - Médico: %s -
Horário: %d/%d - %d:00h até às %d:00h\n", consultas[indice].cliente,
consultas[indice].medico, consultas[indice].horario.mes,
consultas[indice].horario.dia, consultas[indice].horario.hora,
consultas[indice].horario.hora+1);
        }
        }else {
        printf("Consulta não encontrada.\n");
        alterar_data(clientes, medicos, consultas, n_clientes, n_medicos, n_consultas);
        }
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }


        void listagem_dia_medico(ST_CLIENTES clientes[], ST_MEDICOS medicos[],
ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int *n_consultas){
        ST_DATA dataAtual;
        data_atual(&dataAtual);
        int encontrados = 0;
        printf("============================== LISTAGEM DE CONSULTAS
==============================\n");
        printf("Data: %d/%d - Hora: %d:00h\n", dataAtual.dia, dataAtual.mes,
dataAtual.hora);
        printf("\n");

        for(int i = 0; i < *n_consultas; i++){
        if(consultas[i].horario.dia == dataAtual.dia && consultas[i].horario.mes ==
dataAtual.mes){
        printf("Médico: %s - Cliente: %s - Hora: %d:00h até às %d:00h\n",
consultas[i].medico, consultas[i].cliente, consultas[i].horario.hora,
consultas[i].horario.hora+1);
        encontrados++;
        }
        }
        if (encontrados == 0){
        printf("Nenhuma consulta para dia de hoje.\n");
        }
        menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
        }
```

```
	void historico_consultas_clientes(ST_CLIENTES clientes[], ST_MEDICOS
medicos[], ST_CONSULTAS consultas[], int *n_clientes, int *n_medicos, int
*n_consultas){
	char cliente[50];
	int id_cliente, encontrados = 0;
	printf("============================== HISTÓRICO DE CONSULTAS POR
CLIENTE ==============================\n");
	printf("Nome do cliente: ");
	fgets(cliente, sizeof(cliente), stdin);
	cliente[strcspn(cliente, "\n")] = '\0';


	printf("\nID do cliente: ");
	scanf("%d", &id_cliente);
	getchar();


	for(int i = 0; i < *n_consultas; i++){
	if (strcmp(consultas[i].cliente, cliente) == 0 && consultas[i].id_cliente ==
id_cliente){
	printf("\nConsulta ID: %d\n", consultas[i].ID);
	printf("Médico: %s\n", consultas[i].medico);
	printf("Data: %d/%d - %d:00h até às %d:00h\n", consultas[i].horario.dia,
consultas[i].horario.mes, consultas[i].horario.hora, consultas[i].horario.hora+1);
	if(consultas[i].estado == 0){
	printf("Estado: Realizada\n");
	}else if(consultas[i].estado == 1){
	printf("Estado: Agendada\n");
	}else if(consultas[i].estado == -1){
	printf("Estado: Cancelada\n");
	}
	encontrados++;
	}
	}if(encontrados == 0){
	printf("Nenhuma consulta encontrada.\n");
	}
	menu_principal(clientes, medicos, consultas, n_clientes, n_medicos,
n_consultas);
	}


	int verificar_medico(ST_MEDICOS medicos[], int *n_medicos, char
medico[50]){
	int encontrados = 0;
	for (int i = 0; i < *n_medicos; i++){
	if (strcmp(medicos[i].nome, medico) == 0){
	encontrados++;
```

```
        }
        }
        if (encontrados > 0){
        return 1;
        }else {
        return -1;
        }
        }


        int verificar_cliente(ST_CLIENTES clientes[], int *n_clientes, char cliente[50], int
id_cliente){
        for (int i = 0; i < *n_clientes; i++){
        if (strcmp(clientes[i].nome, cliente) == 0 && clientes[i].ID == id_cliente){
        return 1;
        }
        }
        return -1;
        }


        int verificar_horario(ST_CONSULTAS consultas[], int *n_consultas, char
medico[], int dia, int mes, int hora){
        ST_DATA dataAtual;
        data_atual(&dataAtual);

        if (mes < (dataAtual.mes) || (mes == (dataAtual.mes) && dia < dataAtual.dia) ||
(mes == (dataAtual.mes) && dia == dataAtual.dia && hora <= dataAtual.hora)) {
        return -1;
        }
        // Como exemplo: Clínica abre às 8:00h e fecha as 18:00h
        if (hora < 8 || hora >= 18){
        return -1;
        }
        for (int i = 0; i < *n_consultas; i++){
        if(strcmp(consultas[i].medico, medico) == 0 && consultas[i].horario.dia == dia
&& consultas[i].horario.mes == mes && consultas[i].horario.hora == hora){
        return -1;
        }
        }
        return 1;
        }


        int atribuir_id_consulta(int *n_consultas){
```

```
int id = 10000 + (*n_consultas);
return id;
}
```

### *3.2.6 auxiliares.c*

O ficheiro ‘auxiliares.c’ é composto por funções auxiliares que implementam funcionalidades na utilização da aplicação, incluindo a atribuição automática da cidade com base no código postal, a obtenção da data e hora atual do sistema, e a verificação das especialidades do médico.

A função ‘cp_cidade’ atribui automaticamente o nome da cidade com base no código postal fornecido, recebe como parâmetros um valor inteiro referente ao código postal e um apontador de um array de caracteres referente ao nome da cidade. Devido a elevados números de códigos postais em Portugal, apenas foi implementado no distrito de Viana do Castelo.

A função ‘data_atual’ utiliza a biblioteca ‘time.h’ para obter a data e hora atual preenchendo os campos da estrutura ‘ST_DATA’ como o dia, mês e hora.

A função ‘verifica_especialidade’ recebe uma string referente ao nome da especialidade e verifica se corresponde as opções correspondentes previamente definidas (Cardiologia, Dermatologia, Neurologia, Pediatria, Ortopedia, Radiologia, Psiquiatria e Urologia). Caso a especialidade for válida, é retornado o valor de 1, caso contrário, retorna o valor de 0.

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "auxiliares.h"


int cp_cidade(int codigo_postal, char *cidade){
if (codigo_postal >= 4900000 && codigo_postal <= 4999999){
if (codigo_postal >=4900000 && codigo_postal <= 4909999){
strcpy(cidade, "Viana do Castelo");
return 1;
}else if(codigo_postal >= 4970000 && codigo_postal<= 4979999){
```

```
strcpy(cidade, "Arcos de Valdevez");
return 1;
}else if(codigo_postal >= 4990000 && codigo_postal <= 4999999){
strcpy(cidade, "Ponte de Lima");
return 1;
}else if(codigo_postal >= 4910000 && codigo_postal <= 4919999){
strcpy(cidade, "Caminha");
return 1;
}else if (codigo_postal >= 4920000 && codigo_postal <= 4929999){
strcpy(cidade, "Vila Nova de Cerveira");
return 1;
}else if (codigo_postal >= 4930000 && codigo_postal <= 4939999){
strcpy(cidade, "Valença");
return 1;
}else if(codigo_postal >= 4940000 && codigo_postal <= 4949999){
strcpy(cidade, "Paredes de Coura");
return 1;
}else if(codigo_postal >= 4950000 && codigo_postal <= 4959999){
strcpy(cidade, "Monção");
return 1;
}else if(codigo_postal >= 4960000 && codigo_postal <= 4969999){
strcpy(cidade, "Melgaço");
return 1;
}else if(codigo_postal >= 4980000 && codigo_postal <= 4989999){
strcpy(cidade, "Ponte de Barca");
return 1;
}
}
return 0;
}


void data_atual(ST_DATA *data_hora_atual){
time_t t = time(NULL);
struct tm tm = *localtime(&t);
data_hora_atual->dia = tm.tm_mday;
data_hora_atual->mes = tm.tm_mon + 1;
data_hora_atual->hora = tm.tm_hour;
}


int verificar_especialidade(char *especialidade){
if (strcmp(especialidade, "Cardiologia") == 0){
return 1;
}else if(strcmp(especialidade, "Dermatologia") == 0){
return 1;
```

```
}else if(strcmp(especialidade, "Neurologia") == 0){
return 1;
}else if(strcmp(especialidade, "Pediatria") == 0){
return 1;
}else if(strcmp(especialidade, "Ortopedia") == 0){
return 1;
}else if(strcmp(especialidade, "Radiologia") == 0){
return 1;
}else if(strcmp(especialidade, "Psiquiatria") == 0){
return 1;
}else if(strcmp(especialidade, "Urologia") == 0){
return 1;
}
return 0;
}
```

## 3.3 Diretivas de Compilação

Concluído o capítulo da estruturação do código, a utilização de diretivas de compilação como ‘#ifndef’, ‘#define’ e ‘endif’ é essencial para garantir eficiência e segurança do código, principalmente na utilização de vários ficheiros headers. As diretivas são instruções que o compilador interpreta durante a fase de pré-processamento do código, permitindo o controlo do comportamento da compilação e evitando problemas como inclusão múltipla do mesmo ficheiro.

A diretiva ‘#ifndef’ faz a verificação de uma macro, se ainda não foi definida, a diretiva ‘#define’ faz a declaração da macro e o código entre ‘#ifndef’ e o ‘#endif’ é incluído pelo compilador. Sendo assim é evitado que os mesmos ficheiros headers sejam processados novamente. A aplicação e a compreensão das diretivas de compilação foram fundamentais para organização e segurança do projeto, sendo fudamentais em grandes projetos, onde é essencial o trabalho em grupo.

# 4 Conclusão

O projeto da aplicação de gestão para clínicas, desenvolvido em linguagem C, foi concluído atendendo aos requisitos propostos no enunciado. Uma solução de abrangente para gerenciar eficientemente os recursos de uma clínica, permitindo a gestão dos clientes, médicos e consultas, oferecendo uma interface simples e funcional.

A implementação em linguagem C, utilizando estruturas, vetores e funções, fornece uma implementação organizada e de fácil acesso aos dados. Além da utilização de programação por módulos, dividindo o projeto em diversos módulos, cada um responsável por funções específicas, facilitando a processo de testes e correção de erros.

Além disso, a verificação e restrição das entradas do utilizador, são uma boa prática na programação, contribuindo para confiabilidade e reduzindo as possíveis más intenções dos utilizadores.

Embora a aplicação seja funcional para o seu uso, a implementação de melhorias adicionais trairia vantagens ao projeto, a utilização de estruturas de dados como conceitos de fila, pilha ou árvores binárias, e a utilização de memória dinâmica para o gerenciamento de memória conforme necessário, oferecem um bom caminho para expandir o conhecimento e otimizar o projeto.

Em resumo, o projeto de gestão de clientes, médicos e consultas, além de funcional e simples, serve como uma boa base para futuros projetos, evidenciando a organização do código como essencial para o desenvolvimento e uma possível adição futura de funcionalidades avançadas.

# 5 Bibliografia

Damas, L. (2015). Linguagem C (24th ed.). Lisboa: FCA.