



SI203B/SI200 - Algoritmos e Programação de Computadores II

Marcelo Amaral Dotta 285716
Nicholas Cardoso Alencar 297207
Arthur Besseler Baldini 249420
Nícolas Ribeiro Luz 296731
Luca de Menezes Fraga 249375

Grupo 12 - Biblioteca Virtual

Projeto desenvolvido como exigência parcial para a aprovação na disciplina de Algoritmo e Programação de Computadores II, ministrada na Faculdade de Tecnologia da Universidade Estadual de Campinas, no curso de bacharelado em Sistemas de Informação

Professor: Profa. Dra. Thais Rocha.

Limeira
2025

Sumário

| | |
|-------------------------------------|-----------|
| 1. Introdução | 3 |
| 2. Análise de requisitos | 4 |
| 2.1. Requisitos funcionais | 4 |
| 2.2. Requisitos não funcionais | 4 |
| 3. Diagramas | 6 |
| 3.1. Diagrama de Caso de Uso | 6 |
| 3.2. Diagrama de Classes | 7 |
| 4. Implementação | 8 |
| 4.1. Arquivo sistema_ingressos.c | 8 |
| 4.2. Arquivo cinema.h | 10 |
| 4.3. Arquivo cinema.c | 11 |
| 4.4. Compilação e execução | 23 |
| 5. Protótipos de tela | 25 |
| 5.1. Tela inicial | 25 |
| 5.2. Tela de cadastro | 25 |
| 5.3. Tela de atualização | 27 |
| 5.4. Tela de consulta | 28 |
| 5.5. Tela de exclusão | 29 |
| 5.6. Tela de consulta após exclusão | 30 |
| 5.7. Tela de encerramento | 30 |
| 6. Conclusão | 31 |



1. Introdução

Este projeto prático consiste no desenvolvimento de um sistema de biblioteca digital na linguagem C, contendo documentação completa e detalhada, todos os seus requisitos, arquivos criados durante a elaboração do projeto, código fonte, tela e repositório para exposição do código e avaliação acadêmica

Os diagramas são um Diagrama de Caso de Uso do sistema, detalhando as funcionalidades do sistema e contendo o Cliente e o Administrador como atores, e um Fluxograma discorrendo sobre como o sistema se comporta. A tela, por sua vez, é uma representação gráfica e representativa de como é a interface do sistema implementado

O código do sistema em si é capaz de cadastrar clientes, validar dados e credenciais inseridos, fazer e desfazer login, excluir usuários, atualizar dados e credenciais, consultar cadastros, registrar livros, fazer um empréstimo e devolvê-lo com datas de devolução, consultar empréstimos e administrador criar e excluir usuários.



2. Análise de requisitos

2.1. Requisitos funcionais

RF001 - Cadastro de usuário: O sistema deve permitir o usuário realizar cadastro de uma nova conta

Entrada: nome de usuário, email e senha

Saída: Confirmação de dados cadastrais inseridos

RF002 - Validar dados cadastrais inseridos: O sistema deve checar se os dados inseridos no cadastro são válidos

Entrada: Nome, email e senha já inseridos pelo usuário

Saída: Confirmação de cadastro de usuário concluído

RF003 - Fazer login: O sistema deve permitir que o usuário entre com login e senha

Entrada: Login e senha válidos

Saída: Entrada na tela da biblioteca

RF004 - Desfazer login: O sistema deve permitir que o usuário saia com login dele

Entrada: deslogar

Saída: Entrada na tela de login

RF005 - Excluir usuários: O sistema deve permitir que o usuário exclua o seu cadastro se desejado

Entrada: Confirmação de exclusão de usuário

Saída: Confirmação de usuário excluído com sucesso

RF006 - Atualizar os dados cadastrais: O sistema deve permitir que o usuário edite seus dados de cadastro

Entrada: Dados a serem atualizados

Saída: Confirmação de atualização concluída com sucesso

RF007 - Consultar cadastros: O sistema deve permitir que o administrador consulte os usuários cadastrados

Entrada: Dados de administrador

Saída: Todos os usuários cadastrados

RF008 - Cadastrar livro : O sistema deve permitir que o administrador cadastre novos



livros com título, autor, gênero e quantidade disponível

Entrada: Título, autor, gênero e quantidade disponível

Saída: Confirmação de livro cadastrado

RF009 - Registrar empréstimo: O sistema deve permitir que o usuário realize o empréstimo de um livro desejado

Entrada: Número do livro no catálogo

Saída: Confirmação de empréstimo realizado com sucesso e informar a data limite para devolução

RF010 - Registrar devoluções: O sistema deve permitir que o usuário devolva um livro que está com ele

Entrada: Usuário e senha/número do livro na lista de empréstimos

Saída: Confirmação de devolução concluída com sucesso

RF011 - Consultar empréstimos de usuário: O sistema deve permitir que o usuário veja quais seus empréstimos ativos com datas de devolução

Entrada: usuário e senha

Saída: histórico de empréstimos ativos e antigos

RF012 - administrador cadastrar usuários: O sistema deve permitir que o administrador cadastre usuário direto do sistema

Entrada: id, nome, senha, email

Saída: usuário cadastrado

2.2. Requisitos não funcionais

RNF001 - Usabilidade: O sistema deve possuir uma interface simples e intuitiva, porém deve ser funcional.

RNF002 - Desempenho: O código deve ser otimizado para que o usuário tenha uma resposta em ao menos 3 segundos.

RNF003 - Confiabilidade: O sistema deve ser confiável, ou seja, não se pode permitir que 2 usuários façam o empréstimo do mesmo exemplar, além de manter atualizado as quantidades e disponibilidades de livros.

RNF004 - Segurança: Apenas usuários cadastrados e logados podem realizar empréstimos e devoluções.

RNF005 - Manutenibilidade: O código deve ser modular e comentado, facilitando a manutenção do mesmo.

RNF006 - Disponibilidade: O sistema deve estar disponível 24h por dia.

3. Diagramas



3.1. Diagrama de Caso de Uso

https://drive.google.com/file/d/1mpM1H2EZKVbVwQLFNjK_72D9DEaWyROg/view?usp=drive_link

3.2. Fluxograma

https://drive.google.com/file/d/15cNLa4dhZZL0FWEfaABCKkjO2Z3ihaG1/view?usp=drive_link

4. Implementação

A implementação do projeto está estruturada em 3 arquivos. O arquivo main.c, o biblioteca.h e o biblioteca.c. Cada um deles tem um trabalho que será melhor comentado adiante no projeto. Além da utilização do github para facilitar o trabalho em equipe.

Acesso ao [projeto](#) no Github

4.1. Arquivo main.c

Possui a função “main”, responsável pelo início do programa, chamando a função start(), a qual roda o programa. Além de criar as variáveis globais utilizadas no programa.

C/C++

```
#include "biblioteca.h"
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <windows.h>

int contagemPassagemEmail = 0;
int conatgemPassagemUserName = 0;
int idUsuarioLogado;

char nomeUsuario[100];
char email[100];
char senha[50];
int numeroId;

int main()
{
    start();

    return 0;
}
```

4.2. Arquivo biblioteca.h

Contém o cabeçalho das funções utilizadas e as structs.

C/C++

```
#ifndef BIBLIOTECA_H_INCLUDED
#define BIBLIOTECA_H_INCLUDED

extern int contagemPassagemEmail;
extern int conatgemPassagemUserName;
extern int idUsuarioLogado;
extern char nomeUsuario[100];
extern char email[100];
extern char senha[50];
extern int numeroId;

typedef struct {
    char titulo[50];
    char autor[50];
    char genero[50];
    char data[50];
    int ano;
    int quantidade;
} Catalogo;

void limpar_Tela();
void continuar();
int encerrar_Codigo();
int encerrar_Codigo();
void validacao_Nome_Usuario();
void criarConta();
void fazer_Login();
int Ler_Opcoes();
void opcoes_Menu_Biblioteca();
void opcoes_Menu_Biblioteca_Adm();
int menu_Principal();
void emprestimo(int id);
void consultarCatalogo(int id);
void apagarRegistro();
```

```
C/C++  
void registrarLivro();  
void consultarEmprestimo(int id);  
void atualizar_Dados_Cadastrais();  
void deslogar();  
void usuario_Excluir_Usuario();  
void adm_Excluir_Usuarios(int id);  
void consultar_Cadastros(int id);  
int validacao_Id();  
void adm_Criar_Contra(int id);  
void opcoes_Menu_Contra();  
void opcoes_Menu_Contra_Adm();  
void menu_Contra();  
void menu_Biblioteca(int id);  
void start();  
  
#endif // BIBLIOTECA_H_INCLUDED
```

4.3. Arquivo biblioteca.c

Foi dividido em alguns blocos para melhor visualização.

4.3.1. Definições básicas

4.3.2. Função criarConta()

A função chama 2 outras funções as quais recebem o nome e email do cadastro, após isso a própria função recebe a senha do usuário. Então ele armazena tudo em um arquivo mestre, além de gerar um ID para esse usuário, arquivando também no arquivo mestre. Além de contar a quantidade de pessoas cadastradas em um novo arquivo txt, para que possamos usar esse número como ID dos próximos usuários, evitando que várias pessoas tenham o mesmo ID.

```
C/C++  
void criarConta() {  
    FILE* contada = fopen("BD/contador.txt", "w");  
    FILE* mestre = fopen("BD/arquivoMestre.txt", "a+");  
    FILE *usuario;  
    char nomeArquivoUsuario[20];  
  
    validacao_Nome_Usuario();  
  
    validacao_Email();  
  
    printf("Digite a senha: ");  
    scanf("%s", senha);  
  
    sprintf(nomeArquivoUsuario, "BD/usuarios/%d.txt", numeroId);  
    usuario = fopen(nomeArquivoUsuario, "w");  
  
    fprintf(mestre, "\n%d %d %s %s", 0, numeroId, nomeUsuario, senha, email);  
  
    numeroId++;  
    fprintf(contada, "%d", numeroId);  
  
    fclose(contada);  
    fclose(usuario);  
    fclose(mestre);  
}
```

4.3.3. Função validacao_Email()

A função recebe o email do usuário e verifica se possui o caractere '@', caso o possua, o programa guarda o email e continua a normalmente o programa. Porém caso o email seja invalido, chamamos novamente a função para que o usuário coloque um novo e-mail válido. Isso demonstra a recursividade aplicada no nosso projeto.

```
C/C++  
void validacao_Email(){  
    contagemPassagemEmail++;  
    printf("Digite o email: ");  
    scanf("%s", email);  
  
    for (int i = 0; i < strlen(email); i++){  
        if (email[i] == '@'){  
            contagemPassagemEmail = -26;  
        }  
    }  
  
    if (contagemPassagemEmail != -26){  
        limpar_Tela();  
        printf("Email invalido!\n");  
        validacao_Email();  
    }  
}
```

4.3.4. Procedimento validacao_Nome_Usuario()

Função na qual recebe o nome do usuário que está se cadastrando, o mesmo é

validado, ou seja, o arquivo mestre é percorrido e verifica se já existe um nome de usuário igual ao inserido, caso não exista o programa recebe o nome e continua normalmente. Porém caso o nome exista ele retorna uma mensagem de erro e pede um novo nome.

C/C++

```
void validacao_Nome_Usuario(){
    FILE* mestre = fopen("BD/arquivoMestre.txt", "r");

    int nomeValido = 0, idPreencher, statusPreencher;
    char nomeUsuarioValidacao[100], senhaPreencher[50], emailPreencher[100];

    while (!nomeValido) {
        printf("Digite o nome de usuario(sem espacos e numeros): ");
        scanf("%s", nomeUsuario);

        nomeValido = 1;
        rewind(mestre);

        while (fscanf(mestre, "%d %d %s %s", &statusPreencher, &idPreencher,
        &nomeUsuarioValidacao, &senhaPreencher, &emailPreencher) != EOF){
            if (strcmp(nomeUsuario, nomeUsuarioValidacao) == 0){
                limpar_Tela();
                printf("Esse nome de usuario ja existe! Insira outro\n");
                nomeValido = 0;
                validacao_Nome_Usuario();
                break;
            }
        }
        fclose(mestre);
    }
}
```

4.3.5. Procedimento fazer_Login()

Esse procedimento recebe um nome de usuário e um a senha e percorre todo o arquivo mestre, caso ele reconheça que existe um usuário correspondente ao nome e a senha ele chama a função menu Biblioteca, mandando o ID do usuário logado para diferenciar contas normais e de adm, dando acesso ao sistema, caso os nomes estejam incorretos ele chama novamente a função fazer_Login.

C/C++

```
void fazer_Login(){
    FILE* mestre = fopen("BD/arquivoMestre.txt", "r");
    char nomeUsuarioInserido[100], senhaUsuario[100];
    int entrou = 0, statusUsuario;

    printf("Digite o seu userName: ");
    scanf("%s", nomeUsuarioInserido);

    printf("Digite a senha: ");
    scanf("%s", senhaUsuario);

    rewind(mestre);

    while (fscanf(mestre, "%d %d %s %s %s", &statusUsuario, &idUsuarioLogado,
    &nomeUsuario, &senha, &email) != EOF){
        if (strcmp(nomeUsuarioInserido, nomeUsuario) == 0 && strcmp(senhaUsuario, senha)
        == 0 && statusUsuario == 0){
            fclose(mestre);
            menu_Biblioteca(idUsuarioLogado);
            entrou++;
        }
    }
    if (entrou == 0){
        limpar_Tela();
        printf("Usuario ou senhas incorretos!\n");
        fclose(mestre);
        fazer_Login();
    }
}
```

4.3.6. Procedimento menu_Biblioteca()

Função a qual chama uma outra função que apresenta as opções e após recebe um número do usuário que definirá o que o sistema fará em sequência através da função switch. Além de responder “Opção inexistente” caso não exista a opção.

C/C++

```
void menu_Biblioteca(int id){  
    limpar_Tela();  
  
    int opcao;  
    if (id < 0) {  
        opcoes_Menu_Biblioteca_Adm();  
        opcao = Ler_Opcoes();  
    } else {  
        opcoes_Menu_Biblioteca();  
        opcao = Ler_Opcoes();  
    }  
  
    switch (opcao) {  
    case 1:  
        consultarCatalogo(id);  
        break;  
    case 2:  
        emprestimo(id);  
        break;  
    case 3:  
        devolucao(id);  
        break;  
    case 4:  
        consultarEmprestimo(id);  
        break;  
    case 5:  
        menu_Conta(id);  
        break;  
    case 6:  
        limpar_Tela();  
        encerrar_Codigo();  
        break;  
    case 1113:  
        registrarLivro(id);  
        break;  
    case 1114:  
        apagarregistro(id);  
        break;  
  
    default:  
        printf("Opção inexistente!!");  
        continuar();  
        menu_Biblioteca(id);  
        break;  
    }  
}
```

4.3.7. Procedimento consultarCatalogo()

Esse procedimento abre o arquivo.txt chamado catálogo, onde estão guardados todos os livros cadastrados e coloca-os na tela do usuário, sendo apenas uma tela de consulta, ou seja, após receber um char do usuário, volta ao menu da biblioteca normalmente.

```
C/C++  
  
void consultarCatalogo(int id)  
{  
    limpar_Tela();  
    char lixo;  
    int linha = 1;  
    Catalogo livro;  
    FILE *arq = fopen("BD/catalogo.txt","r");  
  
    printf("Livros disponíveis:\n\n");  
    while(fscanf(arq, " %[^\n] %c %[^\n] %c %d %c %d", &livro.titulo, &lixo,  
    livro.autor, &lixo, livro.genero, &lixo, &livro.ano, &lixo, &livro.quantidade) != EOF)  
    {  
        printf("%d| %s| %s| Ano de lançamento: %d | Quantidade Disponível: %d\n",  
        linha, livro.titulo, livro.autor, livro.genero, livro.ano, livro.quantidade);  
        linha++;  
    }  
  
    fclose(arq);  
  
    printf("\nPressione qualquer tecla para continuar\n");  
    while ((lixo = getchar()) != '\n' && lixo != EOF);  
    getchar();  
    limpar_Tela();  
    menu_Biblioteca(id);  
}
```

4.3.8. Procedimento Emprestimo()

O procedimento de Empréstimo, primeiramente ele abre 3 arquivos, o Catálogo, um arquivo temporário, e o registro. O código percorre o catálogo e mostra para o usuário as opções de livros disponíveis. Então ele permite que o usuário digite um número para escolher o livro desejado. Caso ele selecione um livro que está disponível o sistema calcula a data de devolução (1 semana), e registra o empréstimo no registro pessoal da pessoa logada, ou seja, cada usuário tem um arquivo.txt em que estão guardados os empréstimos deste usuário. Caso a opção escolhida de livro possua 0 unidades, o sistema retorna “livro indisponível”, além de mandar uma mensagem de erro caso seja uma opção inexistente. Após a escolha do usuário reescrevemos todo o catálogo no arquivo temporário mudando a quantidade disponível do livro escolhido, ao fim removemos o catálogo e transformamos o arquivo temp no catálogo.

```

C/C++
void emprestimo(int id)
{
    limpar_Tela();
    char lixo;
    char nomeArquivo[10];
    int sel = 0;
    int linha = 1;
    Catalogo livro;
    FILE *arq = fopen("BD/catalogo.txt","r");
    FILE *fp = fopen("BD/temp.txt","w");
    FILE *reg;

    //Mostra o catálogo para o usuário
    printf("Livros disponíveis:\n\n");
    while(fscanf(arq,"%[^ ] %c %[^ ] %c %d %c %d", &livro.titulo, &lixo,
    livro.autor, &lixo, livro.genero, &lixo, &livro.ano, &lixo, &livro.quantidade) != EOF)
    {
        printf("%d. %s| %s| %s| Ano de lançamento: %d | Quantidade Disponível: %d\n",
        linha, livro.titulo, livro.autor, livro.genero, livro.ano, livro.quantidade);
        linha++;
    }
    rewind(arq);
    int contador = 1;
    linha--;

    printf("\nDigite a opção do livro que você quer emprestar\n");
    scanf("%d", &sel);

    //Menu pra escolha de livro do usuário e Salva a nova quantidade de livros
    //disponível
    while(fscanf(arq,"%[^ ] %c %[^ ] %c %d %c %d", &livro.titulo, &lixo,
    livro.autor, &lixo, livro.genero, &lixo, &livro.ano, &lixo, &livro.quantidade) != EOF)
    {
        if(sel == contador && livro.quantidade == 0){
            printf("\nLivro indisponível");
            continuar();
        }

        if(sel == contador && livro.quantidade > 0){
            livro.quantidade--;
            //Calcula o tempo daqui 1 semana e retorna para o usuário como data de
            devolução
            time_t tempo_atual = time(NULL);
            time_t tempo_futuro = tempo_atual + (7 * 24 * 60 * 60);
            struct tm *info_tempo_futuro = localtime(&tempo_futuro);
            char tempofuturo[50];
            strftime(tempofuturo, sizeof(tempofuturo), "%d/%m/%Y", info_tempo_futuro);

            printf("\nEmpréstimo realizado com sucesso\nData para devolução
            %s",tempofuturo);
            while ((lixo = getchar()) != '\n' && lixo != EOF);
            getchar();
            sprintf(nomeArquivo, "BD/usuarios/%d.txt", id);
            reg = fopen(nomeArquivo,"a");
            fprintf(reg,"%s| %s| %s| %d | Data de devolução: %s", livro.titulo,
            livro.autor, livro.genero, livro.ano, tempofuturo);
            fclose(reg);
        }
    }

C/C++
    //Reescreve o catálogo com a alteração de quantidade em um arquivo temporário
    fprintf(fp,"%s|%s|%s|%d|%d", livro.titulo, livro.autor, livro.genero, livro.ano,
    livro.quantidade);
    contador++;
}

fclose(arq);
fclose(fp);

    //Transforma o arquivo temporário com a alteração no novo catálogo e apaga o antigo
    arquivo
remove("BD/catalogo.txt");
rename("BD/temp.txt", "BD/catalogo.txt");

if(sel > linha || sel <= 0){
    printf("\nOpção inválida\n\n");
    while ((lixo = getchar()) != '\n' && lixo != EOF);
    getchar();
    emprestimo(id);
}

menu_Biblioteca(id);
}

```



4.3.9. Procedimento devolução()

O procedimento de Devolução, primeiramente ele abre 1 arquivo, o registro do usuário logado, caso ele não possua empréstimos o sistema manda uma mensagem e retorna para o menu da biblioteca. Caso ele possua empréstimos, são abertos mais 1 arquivo, o catálogo e 2 arquivos temporários. Primeiramente o usuário recebe em sua tela todos os seus empréstimos e suas datas de devolução, nesse momento ele pode escolher qual livro devolver, ao escolher uma opção válida o sistema reescreve o registro do usuário, retirando o livro devolvido, em um arquivo temporário, e reescreve o catálogo com a quantidade aumentada do livro devolvido no outro arquivo temporário. Ao fim, remove os arquivos originais e transforma os temp em originais.

```

C/C++
void devolução(int id)
{
    limpar_Tela();
    char save[50];
    char lixo;
    char nomeArquivo[30];
    int sel = 0;
    int linha = 1;
    int tamanho;
    Catalogo livro;
    sprintf(nomeArquivo, "BD/usuarios/%d.txt",id);
    FILE *reg = fopen(nomeArquivo,"r");
    fseek(reg, 0, SEEK_END);
    tamanho = ftell(reg);

    if(tamanho == 0){
        printf("Você não possui empréstimos\nAperte qualquer botão para continuar\n");
        while ((lixo = getchar()) != '\n' && lixo != EOF);
        fgetchar();
        fclose(reg);
    }
    else{
        FILE *arq = fopen("BD/catalogo.txt","r");
        FILE *fp = fopen("BD/temp.txt","w");
        FILE *temp = fopen("BD/usuarios/temp2.txt","w");

        rewind(reg);
        printf("Livro(s) Emprestado(s):\n\n");
        while(fscanf(reg,"%[^ ] %c %[^ ] %c %d %c %[^ ]", livro.titulo,
&lixo, livro.autor, &lixo, livro.genero, &lixo, &livro.ano, &lixo, livro.data) != EOF)
        {
            printf("%d. %s| %s| %s| Ano de lançamento: %d | %s\n", linha, livro.titulo,
livro.autor, livro.genero, livro.ano, livro.data);
            linha++;
        }

        rewind(reg);
        int contador = 1;
        linha--;
        printf("\nDigite a opção do livro que você quer devolver\n");
        scanf("%d", &sel);

        while(fscanf(reg,"%[^ ] %c %[^ ] %c %d %c %[^ ]", livro.titulo, &lixo,
livro.autor, &lixo, livro.genero, &lixo, &livro.ano, &lixo, livro.data) != EOF)
        {
            if(sel == contador){
                sprintf(save,"%s",livro.titulo);
                printf("\nDevolução realizada com sucesso");
            }
            if(sel != contador){
                fprintf(temp,"%s| %s| %s| %d | %s", livro.titulo, livro.autor, livro.genero,
livro.ano, livro.data);}

            contador++;
        }

        C/C++
        while(fscanf(arq,"%[^ ] %c %[^ ] %c %d %c %d", livro.titulo, &lixo, livro.autor, &lixo, livro.genero,
&lixo, &livro.ano, &lixo, &livro.quantidade) != EOF)
        {
            if(strcmp(save, livro.titulo) == 0){
                livro.quantidade++;}
            fprintf(fp,"%s| %s| %s| %d| %d", livro.titulo, livro.autor, livro.genero, livro.ano, livro.quantidade);
        }

        fclose(arq);
        fclose(fp);
        fclose(reg);
        fclose(temp);

        remove("BD/catalogo.txt");
        rename("BD/temp.txt", "BD/catalogo.txt");
        remove(nomeArquivo);
        rename("BD/usuarios/temp2.txt", nomeArquivo);

        if(sel > linha || sel <= 0){
            printf("\nOpção invalida\n\n");
            while ((lixo = getchar()) != '\n' && lixo != EOF);
            getchar();
            devolução(id);}
    }

    menu_Biblioteca(id);
}

```

4.3.10. Procedimento registrarLivro()

Esse procedimento recebe Título, Autor, Gênero, Ano de lançamento e quantidade disponível de um livro e os arquiva no catálogo.

C/C++

```
void registrarLivro(int id) {
    limpar_Tela();

    if (id >= 0) {
        printf("Opção inexistente!!!");
        continuar();
        menu_Biblioteca(id);
    } else {
        FILE *fp = fopen("BD/catalogo.txt", "a"); // Abrir o arquivo para leitura

        typedef struct
        {
            char titulo[100], autor[100], genero[100];
            int ano, quantidade;
        } catalogo;
        catalogo livro;

        printf("Informe os dados do livro para o catálogo:\n");
        printf("Título: ");
        scanf(" %[^\n]", livro.titulo);
        printf("Autor: ");
        scanf(" %[^\n]", livro.autor);
        printf("Gênero: ");
        scanf(" %[^\n]", livro.genero);
        printf("Ano de lançamento: ");
        scanf("%d", &livro.ano);
        printf("Quantidade disponível: ");
        scanf("%d", &livro.quantidade);

        fprintf(fp, "\n%s | %s | %s | %d | %d", livro.titulo, livro.autor, livro.genero,
                livro.ano, livro.quantidade);

        fclose(fp);
        menu_Biblioteca(id);
    }
}
```

4.3.11. Procedimento consultarEmprestimo()

O procedimento de consultar Empréstimo, primeiramente abre o arquivo de registro do usuário logado e verifica se existem empréstimos, caso não exista ele retorna uma mensagem e continua para o menu. Se ele possuir empréstimos, colocamos na tela do usuário juntamente de suas datas de devolução e aguarda que o usuário aperte ENTER para voltar ao menu.



C/C++

```
void consultarEmprestimo(int id)
{
    limpar_Tela();
    char lixo;
    char nomeArquivo[30];
    int linha = 1;
    long tamanho;
    Catalogo livro;
    sprintf(nomeArquivo,"BD/usuarios/%d.txt",id);
    FILE *arq = fopen(nomeArquivo,"r");

    fseek(arq, 0, SEEK_END);
    tamanho = ftell(arq);

    if(tamanho == 0){
        printf("Você não possui empréstimos\nAperte qualquer botão para continuar\n");
        while ((lixo = getchar()) != '\n' && lixo != EOF);
        fgetchar();
        fclose(arq);
    }
    else{
        rewind(arq);
        printf("Livros emprestados:\n\n");
        while(fscanf(arq,"%[^ ] %c %[^ ] %c %[^ ] %c %d %c %[^ ]", &livro.titulo, &lixo,
        &livro.autor, &lixo, &livro.genero, &lixo, &livro.ano, &lixo, &livro.data) != EOF)
        {
            printf("%d. %s| %s| %s| Ano de lançamento: %d | %s\n", linha, livro.titulo,
            livro.autor, livro.genero, livro.ano, livro.data);
            linha++;
        }
        fclose(arq);

        printf("\nPressione qualquer tecla para continuar\n");
        while ((lixo = getchar()) != '\n' && lixo != EOF);
        getchar();
        limpar_Tela();
    }
    menu_Biblioteca(id);
}
```

4.3.12. Procedimento atualizar_Dados_Cadastrais()

O procedimento atualizar dados cadastrais serve para que usuário possa atualizar os seus dados cadastrais. Ela funciona pedindo qual dado o usuário quer cadastrar se é nome, senha ou email, depois ele usa um switch case para selecionar a opção e pergunta se o usuário realmente tem certeza que quer alterar o dado, se o usuário confirmar ela vai começar escrever no arquivo temporário os dados já existentes no arquivo mestre enquanto o id do usuário logado for diferente dos escritos no arquivo mestre e se o id do usuário logado for o mesmo que apareceu na hora de transcrever no arquivo temporário ele vai inserir os dados alterados depois remove o arquivo mestre original e deixa o arquivo mestre temporário como o original e informa que o usuário foi atualizado com sucesso caso tenha dado erro na troca ele informa erro e volta ao menu de contas. Caso a autorização seja negada ele volta ao menu de contas.

C/C++

```

void atualizar_Dados_Cadastrais(){
    FILE *mestre = fopen("BD/arquivoMestre.txt", "r"), *mestreTemp;
    mestreTemp = fopen("BD/mestreTemp.txt", "w");

    char nomeTemp[100], senhaTemp[50], emailTemp[100], teste[21] ="BD/arquivoMestre.txt";
    int autorizacao, idTemp, statusUsuario, opcao;

    printf("O que voce deseja alterar? <nome = 0 / senha = 1 / email = 2>: ");
    opcao = Ler_Opcoes();
    switch (opcao) {
        case 0:
            validacao_Nome_Usuario();
            break;
        case 1:
            printf("Insira a nova senha: ");
            scanf("%s", senha);
            break;
        case 2:
            printf("Insira o novo email: ");
            scanf("%s", email);
            break;
        default:
            printf("Opcao inexistente!!!");
            continuar();
            atualizar_Dados_Cadastrais();
            break;
    }

    printf("\nTem certeza que voce deseja alterar o usuario?<sim = 0/ nao = 1\n");
    scanf("%d", &autorizacao);

    if (autorizacao == 0){
        while (fscanf(mestre, "%d %d %s %s %s", &statusUsuario, &idTemp, &nomeTemp,
        &senhaTemp, &emailTemp) != EOF){
            if(idUsuarioLogado != idTemp){
                fprintf(mestreTemp, "%d %d %s %s %s\n", statusUsuario, idTemp, nomeTemp,
                senhaTemp, emailTemp);
            } else {
                fprintf(mestreTemp, "%d %d %s %s %s\n", statusUsuario, idTemp,
                nomeUsuario, senha, email);
            }
        }

        fclose(mestre);
        fclose(mestreTemp);

        if (remove("BD/arquivoMestre.txt") == 0 && rename("BD/mestreTemp.txt",
        "BD/arquivoMestre.txt") == 0) {
            printf("O usuario foi atualizado com sucesso!");
            continuar();
            menu_Conta();
        } else {
            printf("Erro ao substituir o arquivo mestre!\n");
            continuar();
            menu_Conta();
        }
    } else{
        menu_Conta();
    }
}

```

4.3.13. Procedimento consultar_Cadastros()

Primeiramente o sistema verifica se o usuário logado é um administrador, caso não seja ele retorna que a opção é invalidada e retorna para o menu. Caso o usuário logado seja um administrador, o sistema abre o arquivo mestre, onde estão todos os cadastros, e coloca na tela o status, id, nome e email para o administrador. O sistema não mostra a senha para que seja mantida a segurança.

C/C++

```
void consultar_Cadastros(int id){
    if (id >= 0) {
        printf("Opção inexistente!!!");
        continuar();
        menu_Cuenta();
    } else {
        FILE *mestre = fopen("BD/arquivoMestre.txt", "r");
        char nomeTemp[100], senhaTemp[50], emailTemp[100];
        int idTemp, statusUsuarioTemp;

        if (mestre == NULL)
            printf("Erro ao abrir o arquivo de cadastros ");

        limpar_Tela();

        if(id < 0){
            while(fscanf(mestre, "%d %d %s %s %s ", &statusUsuarioTemp, &idTemp, nomeTemp,
senhaTemp, emailTemp) != EOF){
                printf("\nStatus: %d | ID: %d | Nome: %s | Email: %s", statusUsuarioTemp,
idTemp, nomeTemp, emailTemp);
            }
            fclose(mestre);
            continuar();
            menu_Cuenta();
        }
        else {
            printf("Opção inexistente!!!");
            continuar();
            menu_Cuenta();
        }
    }
}
```

4.4. Compilação e execução

Inicialmente, para rodar o programa (considerando um usuário Linux, mas os passos no Windows e IOs são similares), é necessário acessar o terminal do computador, na pasta em que estão inseridos os programas (pode-se utilizar os comandos de ‘cd’ e ‘ls’), após acessar a pasta, inserimos a linha de comando de compilação dos códigos:

Unset

```
gcc biblioteca.c main.c -o main.exe
```

Posteriormente, para executar esse programa, utilizamos o comando:

Unset

```
./main
```



Resultado no terminal:

```
PS C:\Users\nicol> cd C:\Users\nicol\Desktop\Biblioteca-Virtual-\codigo
PS C:\Users\nicol\Desktop\Biblioteca-Virtual-\codigo> ls

Diretório: C:\Users\nicol\Desktop\Biblioteca-Virtual-\codigo

Mode                LastWriteTime       Length Name
----                -----          ---- -
d-----        12/11/2025      19:37            BD
-a----        12/11/2025      18:48        26264 biblioteca.c
-a----        10/11/2025      19:23         1199 biblioteca.h
-a----        10/11/2025      19:23          348 main.c
-a----        13/11/2025      08:15      164819 main.exe

PS C:\Users\nicol\Desktop\Biblioteca-Virtual-\codigo> gcc biblioteca.c main.c -o main.exe
PS C:\Users\nicol\Desktop\Biblioteca-Virtual-\codigo> ./main
```



5. Protótipos de tela

5.1. Tela de Login/Cadastro

Aguardando a escolha do usuário.

```
====Biblioteca Virtual====  
1. Fazer login  
2. Fazer cadastro  
0. Sair
```

Cadastro

```
Digite o nome de usuario(sem espacos e números): Teste  
Digite o email: teste@gmail.com  
Digite a senha: teste123
```

Login

```
Digite o seu userName: teste  
Digite a senha: teste123|
```

5.2. Menu

```
====Biblioteca Virtual====  
1. Visualizar catalogo  
2. Realizar emprestimo  
3. Realizar devolucao  
4. Consultar emprestimos  
5. Conta  
6. Sair  
|
```

Menu administrativo

```
====Biblioteca Virtual====  
1. Visualizar catalogo  
2. Realizar emprestimo  
3. Realizar devolucao  
4. Consultar emprestimos  
5. Conta  
6. Sair  
1113. Cadastrar livro  
1114. Apagar cadastro de livro
```



5.3. Tela de Registro de Livro

Informe os dados do livro para o catálogo:

Título: Jogos Vorazes

Autor: Suzanne Collins

Gênero: Ficção

Ano de lançamento: 2008

Quantidade disponível: 5

5.4. Tela de Apagar Livro

Livros no catálogo:

1. A vida Não é Util | Ailton Krenak | Filosofia | Ano de lançamento: 2020 | Quantidade Disponível: 6
2. Capitães de Areia | Jorge Amado | Romance | Ano de lançamento: 1937 | Quantidade Disponível: 10
3. Diario de um Banana | Jeff Kinney | Comédia | Ano de lançamento: 2007 | Quantidade Disponível: 1
4. Metamorfose | Franz Kafka | Ficção | Ano de lançamento: 1915 | Quantidade Disponível: 9
5. Hobbit | Tolkien | Ficção | Ano de lançamento: 1937 | Quantidade Disponível: 10
6. O Pequeno Príncipe | Antoine | Ficção | Ano de lançamento: 1943 | Quantidade Disponível: 2
7. Zac Power | HI Larry | Ficção | Ano de lançamento: 2014 | Quantidade Disponível: 23
8. A Revolução dos Bichos | George Orwell | Ficção | Ano de lançamento: 1945 | Quantidade Disponível: 10
9. Jogos Vorazes | Suzanne Collins | Ficção | Ano de lançamento: 2008 | Quantidade Disponível: 5

Digite a opção do livro que você quer apagar:

9|

5.5. Tela de Consulta

Livros disponíveis:

1. A vida Não é Util | Ailton Krenak | Filosofia | Ano de lançamento: 2020 | Quantidade Disponível: 6
2. Capitães de Areia | Jorge Amado | Romance | Ano de lançamento: 1937 | Quantidade Disponível: 10
3. Diário de um Banana | Jeff Kinney | Comédia | Ano de lançamento: 2007 | Quantidade Disponível: 1
4. Metamorfose | Franz Kafka | Ficção | Ano de lançamento: 1915 | Quantidade Disponível: 9
5. Hobbit | Tolkien | Ficção | Ano de lançamento: 1937 | Quantidade Disponível: 10
6. O Pequeno Príncipe | Antoine | Ficção | Ano de lançamento: 1943 | Quantidade Disponível: 2
7. Zac Power | HI Larry | Ficção | Ano de lançamento: 2014 | Quantidade Disponível: 23
8. A Revolução dos Bichos | George Orwell | Ficção | Ano de lançamento: 1945 | Quantidade Disponível: 10

Pressione qualquer tecla para continuar

|

5.6. Tela de Empréstimo

Livros disponíveis:

1. A vida Não é Util | Ailton Krenak | Filosofia | Ano de lançamento: 2020 | Quantidade Disponível: 6
2. Capitães de Areia | Jorge Amado | Romance | Ano de lançamento: 1937 | Quantidade Disponível: 10
3. Diário de um Banana | Jeff Kinney | Comédia | Ano de lançamento: 2007 | Quantidade Disponível: 1
4. Metamorfose | Franz Kafka | Ficção | Ano de lançamento: 1915 | Quantidade Disponível: 9
5. Hobbit | Tolkien | Ficção | Ano de lançamento: 1937 | Quantidade Disponível: 10
6. O Pequeno Príncipe | Antoine | Ficção | Ano de lançamento: 1943 | Quantidade Disponível: 2
7. Zac Power | HI Larry | Ficção | Ano de lançamento: 2014 | Quantidade Disponível: 23
8. A Revolução dos Bichos | George Orwell | Ficção | Ano de lançamento: 1945 | Quantidade Disponível: 10

Digite a opção do livro que você quer emprestar

4

Empréstimo realizado com sucesso

Data para devolução 19/11/2025|

5.7. Tela de Devolução



Livro(s) Emprestado(s):

1. Metamorfose | Franz Kafka | Ficção | Ano de lançamento: 1915 | Data de devolução: 19/11/2025

Digite a opção do livro que você quer devolver

1

Devolução realizada com sucesso

Pressione para continuar!

5.8. Tela de Conta

====Menu Contas====

1. Atualizar dados cadastrais
2. Sair da conta
3. Excluir meu usuario
4. Voltar

Tela de Conta administrativa

====Menu Contas====

1. Atualizar dados cadastrais
 2. Sair da conta
 3. Excluir meu usuario
 4. Voltar
6822. Excluir usuarios
6823. Consultar cadastros
6824. Criar conta

5.9. Tela de Atualização

====Menu Contas====

1. Atualizar dados cadastrais
2. Sair da conta
3. Excluir meu usuario
4. Voltar

1

O que voce deseja alterar? <nome = 0 / senha = 1 / email = 2>: 0
Digite o nome de usuario(sem espacos e números): Teste123

Tem certeza que voce deseja alterar o usuario?<sim = 0/ nao = 1
0

O usuario foi atualizado com sucesso!
Pressione para continuar!

5.10. Tela de exclusão de usuário

```
==Menu Contas==  
1. Atualizar dados cadastrais  
2. Sair da conta  
3. Excluir meu usuario  
4. Voltar  
3
```

```
Tem certeza que voce deseja excluir o usuario?<sim = 0/ nao = 1>: 0  
0 usuario foi excluido com sucesso!  
Pressione para continuar!
```

5.11. Tela de encerramento

```
Programa Finalizado
```

```
Process returned 0 (0x0) execution time : 6.765 s  
Press any key to continue.
```



6. Conclusão

Conclui-se que, após a finalização da programação do

A partir da elaboração do projeto a equipe pôde aprofundar o aprendizado na linguagem C e na utilização de manipulação de arquivos, código limpo, modularização e alocação de memória.

Durante o trabalho foram códigos revisionados, erros corrigidos, dúvidas respondidas e ideias implementadas, uma atividade que exercita a nossa visão e conduta como programadores.

Foi possível perceber a importância do trabalho em equipe, de se manter uma comunicação clara e de enxergar resoluções de pontos de vistas diferentes que no fim nos levou a concluir o projeto de maneira satisfatória, não só pelo resultado, mas também pelo desenvolvimento que além de construir um software, nos construiu como profissionais.