CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE LINS PROF. ANTÔNIO SEABRA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ALINE ADRIANA SOARES

PORTFÓLIO ACADÊMICO

LINS/SP
2º SEMESTRE/2023

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE LINS PROF. ANTÔNIO SEABRA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ALINE ADRIANA SOARES

PORTFÓLIO ACADÊMICO

Trabalho de Conclusão de Curso (TCC) apresentado à Faculdade de Tecnologia de Lins para a obtenção do título de Tecnóloga em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Dr. Anderson Pazin.

LINS/SP
2º SEMESTRE/2023

Soares, Aline Adriana

S676p PORTFÓLIO ACADÊMICO / Aline Adriana Soares. — Lins, 2023.

65f.

Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) — Faculdade de Tecnologia de Lins Professor Antonio Seabra: Lins, 2023.

Orientador(a): Dr. Anderson Pazin

1. Portfólio acadêmico. 2. Programação. 3. Desenvolvimento. 4. JavaScript. 5. CSS. I. Pazin, Anderson. II. Faculdade de Tecnologia de Lins Professor Antonio Seabra. III. Título.

CDD 004.21

Gerada automaticamente pelo módulo web de ficha catalografica da FATEC Lins mediante dados fornecidos pelo(a) autor(a).

ALINE ADRIANA SOARES

PORTFÓLIO ACADÊMICO

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Lins como parte dos requisitos para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas sob orientação do Prof. Dr. Anderson Pazin.

Data de aprovação: ___/___/____

Dr. Anderson Pazin

Me. Júlio Fernando Lieira

Dr. Fernando Augusto Garcia Muzzi

À memória de Djenifer Thais dos Santos, considerada uma irmã e amiga incrível que através da dor de sua perda, me deu forças para transformar tristeza em motivação. Hoje não pode comemorar minhas vitórias, mas sei que onde estiver é a pessoa que mais deixei orgulhosa

Aline Adriana Soares

AGRADECIMENTOS

Agradeço profundamente a todos os professores que estiveram ao meu lado nesta jornada acadêmica, reservando uma gratidão especial ao meu orientador, Dr. Anderson Pazin, pelo tempo dedicado e pelas experiências compartilhadas. À professora Dra. Adriana de Bortoli, reconheço sua sensibilidade ao perceber minhas dificuldades em socializar, proporcionando-me a oportunidade de integrar um grupo no qual conheci pessoas incríveis que se tornaram companheiras durante todo o curso e que levo para a vida.

Expresso meu reconhecimento à minha família e aos meus colegas de sala, cuja parceria e cuidado foram fundamentais durante essa jornada, em especial à Priscila da Silva Batista e Letícia Santos, que, em diversas ocasiões, foram fontes de inspiração e força. Meus agradecimentos se estendem também a pessoas incríveis fora da instituição, como Wagner Williams e Keylla Santos, pelos momentos de apoio e pelas palavras que fizeram toda a diferença.

Quero expressar minha imensa gratidão a minha amiga Julia Manfio, que, em momentos nos quais fatores externos me fizeram duvidar de minha capacidade e merecimento, foi a luz que mostrou o contrário. Mesmo sem perceber, é responsável por influenciar positivamente várias decisões corretas que tomei. Obrigada a todos por fazerem parte dessa jornada significativa em minha vida e contribuírem para o meu crescimento pessoal e acadêmico

Aline Adriana Soares

RESUMO

O portfólio acadêmico consiste na compilação de trabalhos e projetos realizados ao longo do curso, disponibilizados de forma organizada. Foram escolhidas cinco matérias, incluindo linguagem de programação, estrutura de dados, programação avançada orientada a objetos, programação mobile e tópicos especiais em informática, uma para cada semestre do curso com exceção do primeiro. Cada uma delas representa um marco importante do desenvolvimento de conhecimento e habilidades técnicas. Foram utilizadas linguagens como C, Arduino, JavaScript e PHP para a programação, com o Visual Studio Code como editor de código que proporcionou um ambiente de desenvolvimento eficiente. As disciplinas selecionadas são detalhadas, incluindo objetivos, processos com imagens, explicação de código e resultados alcançados e sua seleção representa uma abordagem abrangente das áreas de programação e informática. O portfólio digital foi criado em HTML, CSS e JavaScript, refletindo a evolução do desenvolvedor e servindo como uma ferramenta valiosa para demonstrar competência.

Palavras–chave: Portfólio acadêmico. Programação. Linguagem de programação. Desenvolvimento de habilidades. HTML. CSS. JavaScript.

ABSTRACT

The academic portfolio consists of the compilation of works and projects carried out throughout the course, organized and made available in an organized manner. Five subjects were chosen, including programming languages, data structures, advanced object-oriented programming, mobile programming, and special topics in computer science, one for each semester of the course, except for the first. Each of them represents an important milestone in the development of knowledge and technical skills. Languages such as C, Arduino, JavaScript, and PHP were used for programming, with Visual Studio Code as the code editor. Visual Studio Code provided an efficient development environment. The selected subjects are detailed, including objectives, processes with images, code explanation, and achieved results. The selection of subjects represents a comprehensive approach to the fields of programming and computer science. The digital portfolio was created in HTML, CSS, and JavaScript, reflecting the developer's evolution and serving as a valuable tool to demonstrate competence.

Keywords: Academic portfolio. Programming. Programming languages. Skill development. HTML. CSS. JavaScript.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

IDE –	Integrated Development Environment
HTML-	HyperText Markup Language
CSS-	*Cascading Style Sheets*
BCPL-	Basic Combined Programming Language
ANSI-	American National Standardization Institute
SPI-	Serial Peripheral Interface
OLED-	Organic Light-Emitting Diode
DOM-	Document Object Model
PHP-	Hypertext Preprocessor
SQL-	Structured Query Language
DDL-	Data Definition Language
DML-	Data Manipulation Language
DQL-	Data Query Language
DCL-	Data Control Language
TCL-	Transaction Control Language

LISTA DE SÍMBOLOS

+	Soma
-	Subtração
*****	Multiplicação
/	Divisão
=	Igualdade
>	Maior
<	Menor

SUMÁRIO

1 INTRODUÇÃO

O Portfólio foi a opção escolhida para trabalho de graduação pois me permite mostrar habilidades e experiências, assim como acompanhar o meu desenvolvimento no decorrer do curso superior e mantê-lo atualizado com projetos pessoais ao longo da carreira. Quando bem elaborada ajuda na credibilidade e é um diferencial fundamental para futuros empregadores.

O analista desenvolvedor de sistemas tem como atividade a elaboração, construções de sistemas, análises de dados e soluções especificas em sistemas informatizados. O profissional precisa se manter atualizado em relação ao mercado digital, linguagens e ambientes de programação.

Como representação da minha trajetória no curso de Análise e Desenvolvimento de Sistemas, será apresentado como trabalho de graduação, a evolução acadêmica via trabalhos realizados ao longo dos semestres.

A proposta do portfólio, além da representação da evolução, é também apresentar meu currículo digital e as ferramentas técnicas utilizadas no desenvolvimento. As atividades do curso que foram escolhidas são descritas em forma de artigo. Como atividade do segundo semestre foi escolhida a matéria de Linguagem de programação onde em um programa em C, organizei várias atividades realizadas em sala. Estrutura de dados foi a matéria escolhida para representar o terceiro semestre, sendo apresentado um programa de gerenciamento de uma livraria, desenvolvido em linguagem C. Para o quarto semestre, a matéria escolhida foi Programação avançada orientada a objetos, o projeto é um mini humidificador com sensor de umidade e temperatura programado utilizando um microcontrolador Arduino. Programação para dispositivos moveis é a matéria do quinto semestre, onde foi implementado funções adicionar/editar/excluir/listar e procurar em um site como avaliação do primeiro bimestre.

As tecnologias utilizadas no desenvolvimento do portfólio digital foram *Hypertext Markup Language (*HTML), *Cascading Style Sheets (*CSS*)* JavaScript. Como ambiente de desenvolvimento escolhido foi o Visual Studio Code e o Codepen.

2 DISCIPLINA PROFISSIONAL DO 2º SEMESTRE: LINGUAGEM DE PROGRAMAÇÃO

Para a elaboração do programa, foi utilizada a linguagem C e a ferramenta Dev C++. As atividades escolhidas foram desenvolvidas na disciplina de Linguagem de Programação, todas envolvidas em um só programa para maior visualização do aprendizado do semestre em questão e é possível fazer o download do código no portfólio digital.

2.1 LINGUAGEM C

Criada em 1972, de alto nível e considerada de tipagem fraca, C é a linguagem “mãe” por ser responsável por hoje utilizarmos linguagens mais modernas, e, apesar do tempo é muito utilizada. Sua criação foi direcionada para compiladores e Sistemas Operacionais (S.O). É a linguagem principal na construção do S.O UNIX, assim como MC OS, Linux e Android.

A linguagem C é derivada de duas linguagens do final da década de 60: Algol68 (1968) e B(1970). A linguagem B, por sua vez, foi derivada de BCPL(1969). Posteriormente vieram C++ (1986) e ANSI C(1989).

2.2 BIBLIOTECAS

Existem bibliotecas padrões de linguagem C, utilizadas por funções. A função de cada biblioteca é descrita em um header-file(arquivo-interface), e contém o nome da biblioteca seguido por “.h”. Para ter acesso à biblioteca, o programa deve incluir uma copia do header-file como no exemplo: #include <”nome da biblioteca”.h>.

Foram utilizadas algumas das bibliotecas padrão C:

stdio.h: seu nome vem de Standart imput-output header, traduzindo significa cabeçalho padrão de entrada e saída, ou seja, é possível realizar leitura e escrita por meio de variáveis e funções de entrada e saída disponibilizadas pela máquina.

locale.h: utilizável para a implementação e localização de programas. Junto com a função setlocale(), foi possível configurar para que “ç” e acentuações fossem exibidos no meu programa.

stdlib.h: estão localizadas as funções de alocação e desalocação da memória, assim como converter números representados em string para um tipo de dado que represente um número, por exemplo, double e funções de algoritmo de ordenação.

math.h: biblioteca que contêm várias funções matemáticas básicas como, calculo de raiz quadrada, valor absoluto, trigonometria. As funções dessa biblioteca retornam valores double.

stdbool.h: define o tipo booleano que ocupa apenas 1 byte e pode somente assumir os valores das constantes inteiras true, que possui valor 1 e false de valor 0.

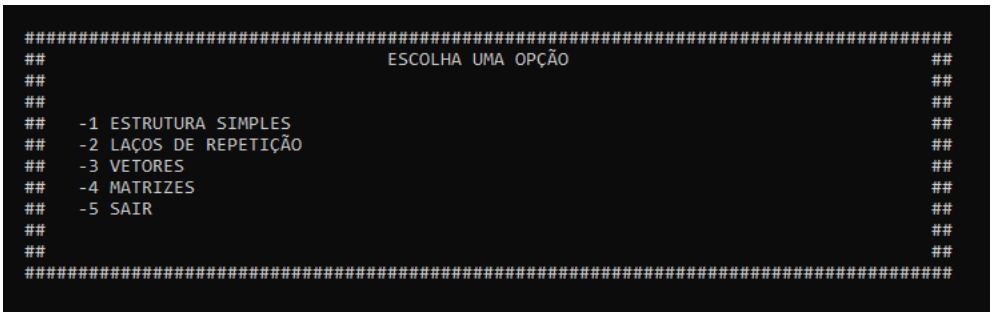
2.3 FUNÇÕES

São um conjunto de comandos realizando uma tarefa. Muito comum na programação estruturada para modularizar o programa. É possível dividir o programa em partes, cada parte realizando uma tarefa. Como a função é montada:

```
tipo_de_retorno nome_da_funcao (parametros)
  {
instruções;
retorno_da_função;
  }
```

Cada exercício foi transformado em uma função, onde o usuário escolhe por interação através de menus. A função principal Main() tem como objetivo apresentar o primeiro menu com cinco opções disponíveis como vemos abaixo:

Figura 2.1 – Menu de opções de exercícios



Fonte: Elaborado pela autora, 2023.

A escolha do usuário é verificada pela condicional IF para verificar se esta enquadrada nas opções exibidas em tela, sendo verdadeira a condição o programa entra em um switch para determinar a ação de acordo com a opção escolhida. Caso

o usuário escolha as opções de 1 à 4, serão chamadas as funções correspondentes, e estas serão explicadas adiante. Se o usuário digitar 5, ou seja, se o mesmo desejar sair do programa, será encerrado através da função exit(0). Utilizando o laço de repetição Do While, o programa exibirá o menu principal enquanto o usuário digitar um número diferente das opções exibidas em tela.

As opções apresentadas na imagem, nos levam para outro menu com uma lista de exercícios a serem escolhidos. Em todos os menus as mesmas condições são aplicadas, somente acrescentando a opção de retornar ao menu principal, onde se escolhida a opção, o programa inicia a função main() novamente. No final de cada exercício são exibidas duas opções, uma que chama a função main() e retornamos ao menu inicial, e a outra para sair do programa, chamando a função exit(0). Assim como no menu principal também foi validada a resposta do usuário, onde o menu se repete enquanto a resposta for diferente das opções exibidas.

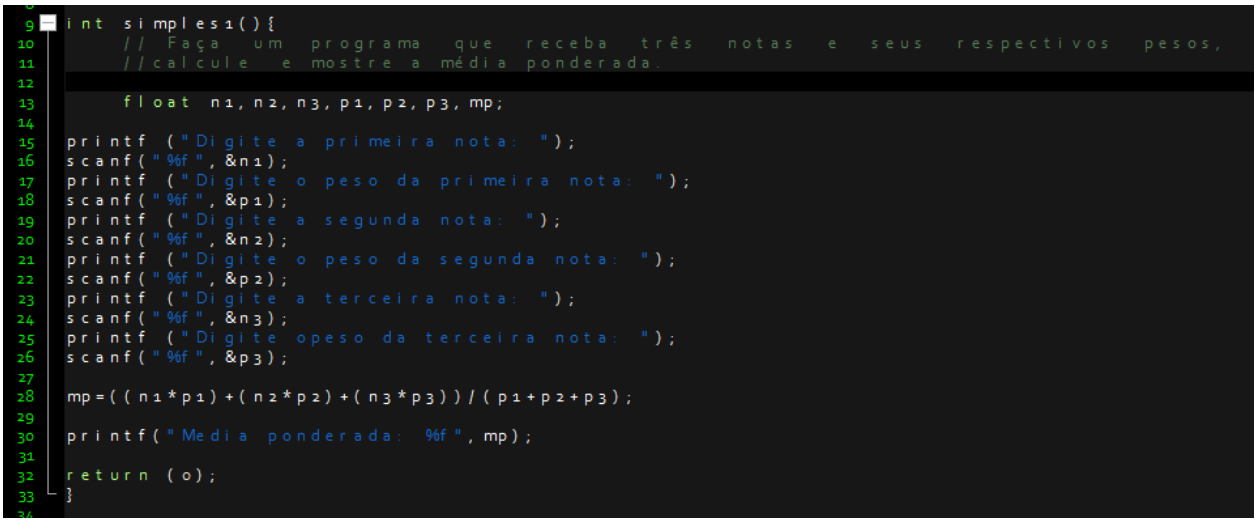
2.4 EXERCÍCIOS DE ESTRUTURA SIMPLES

O primeiro exercício de estrutura simples é iniciado através da função simples1(). A proposta do exercício dado em sala era elaborar um programa que receba três notas e seus respectivos pesos para que seja calculada a média ponderada. O programa inicia declarando as variáveis que serão utilizadas: n1, n2 e n3 para os valores das três notas, p1, p2 e p3 para os respectivos pesos e mp para o valor da média ponderada.

Foi utilizada a formula mp=((n1*p1)+(n2*p2)+ (n3*p3))/(p1+p2+p3). Para maior entendimento do que significa o “peso”, precisamos imaginar que na escola cada prova tem um peso, e esse peso é multiplicado no valor da nota, por exemplo, no caso do programa serão três notas: supondo que a primeira prova o aluno tirou 8 em uma prova de peso 6, na segunda prova o aluno tirou 7 na prova com peso 2 e na terceira nota ele tirou 5 em uma prova de peso 2.

Substituindo as variáveis da equação pelos valores informados acima, teremos o seguinte: mp=((8*6)+(7*2)+(5*2))/(6+2+2), ou seja mp=7,2.

Figura 2.2 – Programa de média ponderada

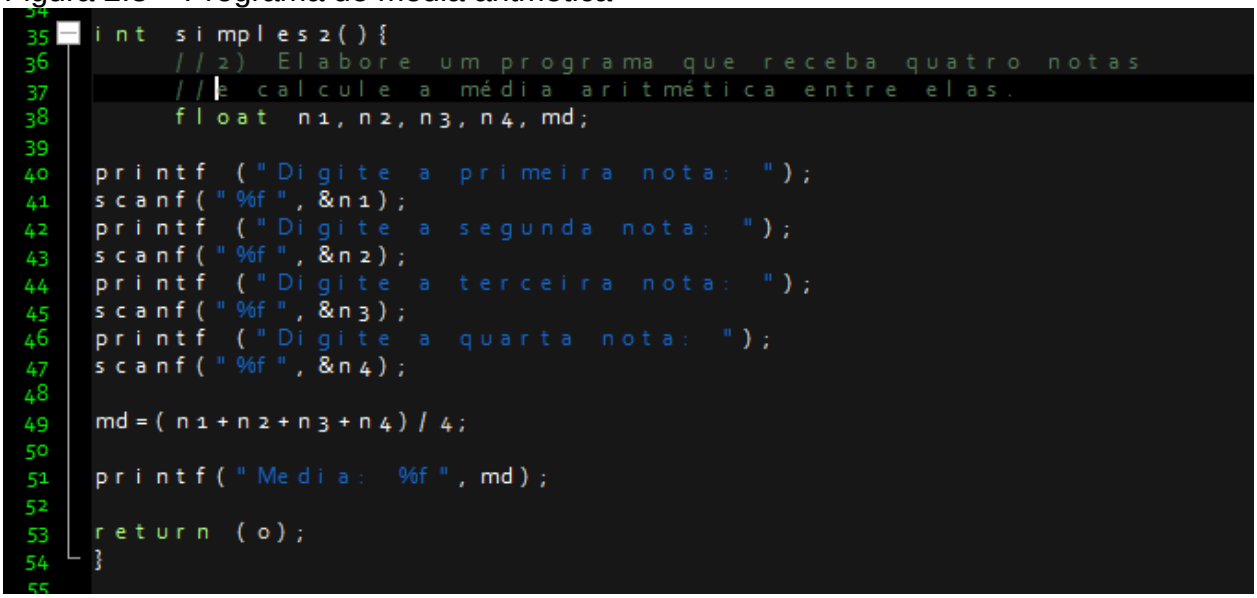


```
int simples1(){
    // Faça um programa que receba três notas e seus respectivos pesos,
    //calcule e mostre a média ponderada.

    float n1, n2, n3, p1, p2, p3, mp;

printf ("Digite a primeira nota: ");
scanf("%f", &n1);
printf ("Digite o peso da primeira nota: ");
scanf("%f", &p1);
printf ("Digite a segunda nota: ");
scanf("%f", &n2);
printf ("Digite o peso da segunda nota: ");
scanf("%f", &p2);
printf ("Digite a terceira nota: ");
scanf("%f", &n3);
printf ("Digite opeso da terceira nota: ");
scanf("%f", &p3);

mp=((n1*p1)+(n2*p2)+(n3*p3))/(p1+p2+p3);

printf("Media ponderada: %f", mp);

return (0);
}
```

Fonte: Elaborado pela autora, 2023.

O segundo exercício de estrutura simples é iniciado pela função simples2(). O objetivo do programa é calcular a média aritmética recebendo quatro notas. A função inicia declarando 5 variáveis do tipo float onde 4 são para guardar as notas e uma para guardar a média. O programa solicita as 4 notas ao usuário e as mesmas são armazenadas nas variáveis n1,n2,n3 e n4. A formula do cálculo da media aritmética é a soma das notas dividido pela quantidade de notas, no caso : md=(n1+n2+n3+n4)/4

Figura 2.3 – Programa de média aritmética



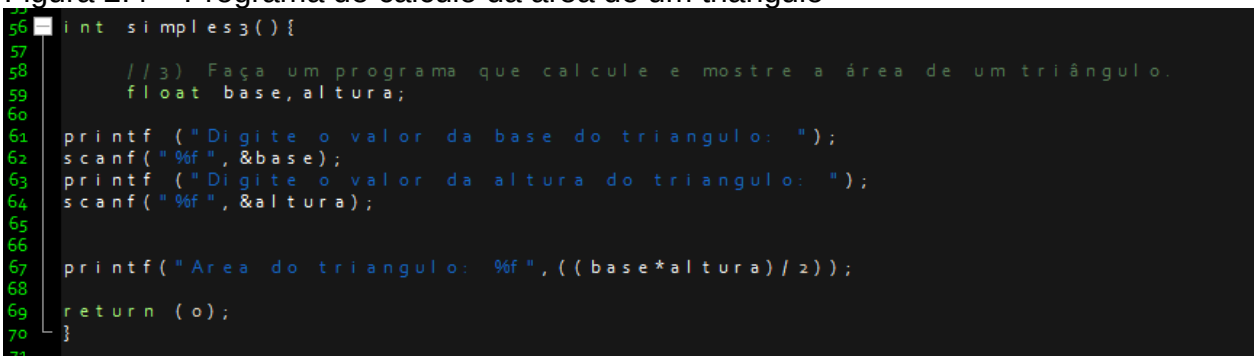
```
int simples2(){
    //2) Elabore um programa que receba quatro notas
    //e calcule a média aritmética entre elas.
    float n1, n2, n3, n4, md;

printf ("Digite a primeira nota: ");
scanf("%f", &n1);
printf ("Digite a segunda nota: ");
scanf("%f", &n2);
printf ("Digite a terceira nota: ");
scanf("%f", &n3);
printf ("Digite a quarta nota: ");
scanf("%f", &n4);

md=(n1+n2+n3+n4)/4;

printf("Media: %f", md);

return (0);
}
```

Fonte: Elaborado pela autora, 2023.

O terceiro exercício de estrutura simples é iniciado pela função simples3(). O programa calcula a área de um triangulo, a função inicia declarando as variáveis base e altura de tipo float, sabemos que a área é o resultado de base*altura/2 então os

valores armazenados são utilizados para a realização do cálculo e tem seu resultado exibido direto ao usuário sendo executado dentro do printf.

Figura 2.4 – Programa de cálculo da área de um triângulo

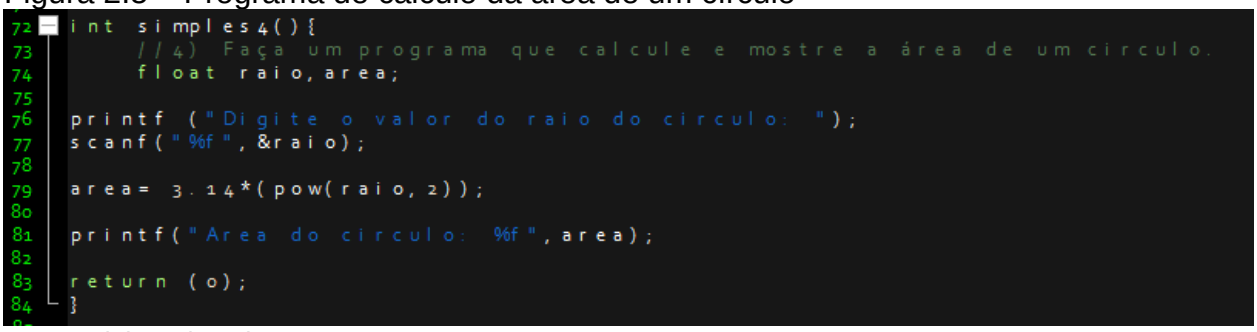


```
int simples3(){

      //3) Faça um programa que calcule e mostre a área de um triângulo.
      float base,altura;

printf ("Digite o valor da base do triangulo: ");
scanf("%f",&base);
printf ("Digite o valor da altura do triangulo: ");
scanf("%f",&altura);


printf("Area do triangulo: %f",((base*altura)/2));

return (0);
}
```

Fonte: Elaborado pela autora, 2023.

O quarto exercício de estrutura simples é iniciado pela função simples4(). O exercício, assim como o anterior, também calcula área, mas desta vez de um círculo. Duas variáveis do tipo float são declaradas, uma para armazenar o valor do raio e a outra para armazenar o resultado. Sabendo que a área é igual a π(pi)*raio ². Para o cálculo de raiz quadrada é necessário utilizar a função pow, para chama-la fazemos da seguinte forma: pow(base, expoente), ou seja a base será nosso raio e o expoente será dois. Para descobrir o valor da área, utilizamos então os valores como na imagem abaixo:

Figura 2.5 – Programa de cálculo da área de um círculo



```
int simples4(){
      //4) Faça um programa que calcule e mostre a área de um circulo.
      float raio,area;

printf ("Digite o valor do raio do circulo: ");
scanf("%f",&raio);

area= 3.14*(pow(raio,2));

printf("Area do circulo: %f",area);

return (0);
}
```

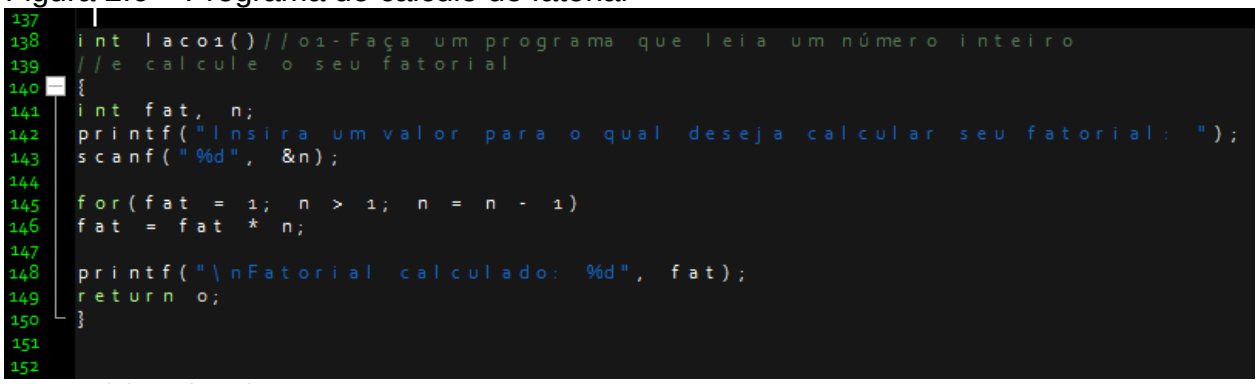
Fonte: Elaborado pela autora, 2023.

2.5 EXERCÍCIOS DE LAÇO DE REPETIÇÃO

O primeiro exercício de laço de repetição deve calcular o fatorial de um número digitado pelo usuário. A função laco1() é iniciada declarando duas variáveis inteiras, uma para receber o valor do fatorial e a outra o número na qual o fatorial será calculado. Para a resolução foi utilizado um laço FOR onde o n é multiplicada por fat

e declaramos que fat é igual a n e a cada iteração é subtraído 1 do valor de n até que n seja igual a 1, ou seja, a cada iteração fat é multiplicada pelo antecessor do número escolhido.

Figura 2.6 – Programa de cálculo de fatorial

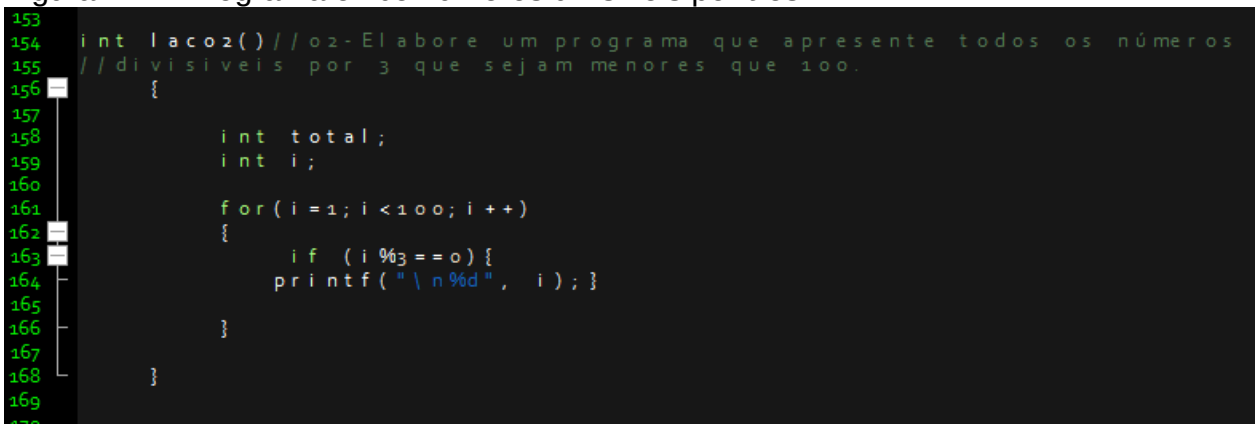


```
int laco1()//01-Faça um programa que leia um número inteiro
//e calcule o seu fatorial
{
int fat, n;
printf("Insira um valor para o qual deseja calcular seu fatorial: ");
scanf("%d", &n);

for(fat = 1; n > 1; n = n - 1)
fat = fat * n;

printf("\nFatorial calculado: %d", fat);
return 0;
}
```

Fonte: Elaborado pela autora, 2023.

O segundo exercício de laço de repetição é iniciado através da função laco2(). O programa exibe números inteiros divisíveis por três e menores que cem. Não há entrada de dados do usuário, somente uma variável inteira chamada i, na qual utilizamos no nosso laço. O laço faz com que para i igual a 1, i menor que 100, i recebe +1. Ou seja, foi criado um contador de 1 a 99. Dentro desse contador tem um IF informando que se o valor do resto da divisão de i por 3 for igual a 0, i será exibido na tela. O programa verifica isso nos números de 1 a 99 como foi estabelecido no laço.

Figura 2.7 – Programa exibe números divisíveis por três



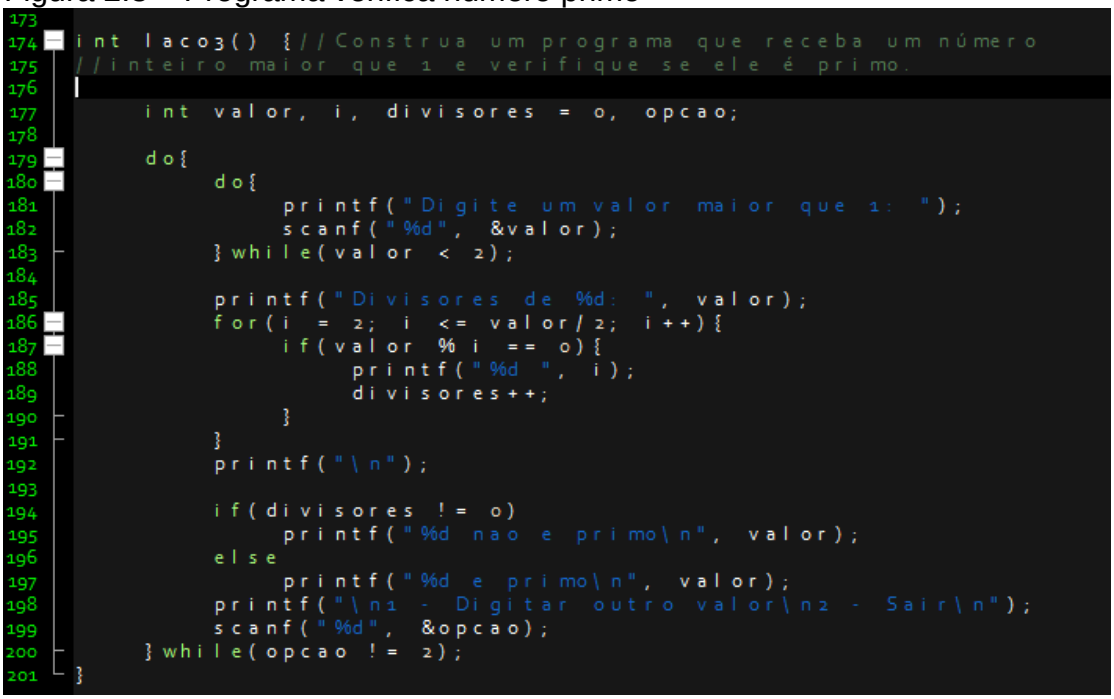
```
int laco2()//02-Elabore um programa que apresente todos os números
//divisiveis por 3 que sejam menores que 100.
    {

        int total;
        int i;

        for(i=1;i<100;i++)
        {
            if (i%3==0){
            printf("\n%d", i);}

        }

    }
```

Fonte: Elaborado pela autora, 2023.

O terceiro exercício de laço de repetição é iniciado na função laco3(), o usuário deve digitar um número inteiro maior que 1. Um Laço de repetição força essa ação do usuário enquanto o valor for diferente do solicitado. Outro laço fica testando os

números divisores do valor, para um número ser primo ele precisa ser divisível somente por 1 e por ele mesmo, então, caso seja provado que o valor é divisível por outro número, um contador incrementa o valor 1 na variável divisores. São exibidos os divisores, a variável divisores é testada e se o seu valor for diferente de 0 e é anunciado que o número não é um número primo. Se divisores for igual a zero, é exibido o resultado dizendo que o número é primo. Esse exercício conta com um laço DO WHILE para verificar se o usuário deseja continuar e testar outro valor. O exercício é repetido enquanto a resposta for igual diferente de 2.

Figura 2.8 – Programa verifica número primo

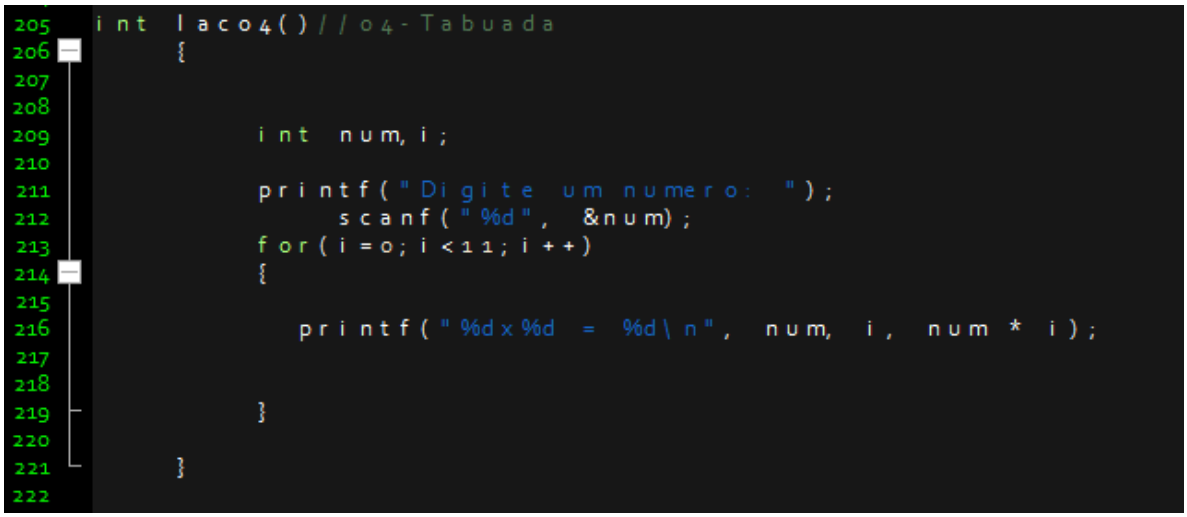


```
173
174 int laco3() {//Construa um programa que receba um número
175 //inteiro maior que 1 e verifique se ele é primo.
176
177     int valor, i, divisores = 0, opcao;
178
179     do{
180         do{
181             printf("Digite um valor maior que 1: ");
182             scanf("%d", &valor);
183         }while(valor < 2);
184
185         printf("Divisores de %d: ", valor);
186         for(i = 2; i <= valor/2; i++){
187             if(valor % i == 0){
188                 printf("%d ", i);
189                 divisores++;
190             }
191         }
192         printf("\n");
193
194         if(divisores != 0)
195             printf("%d nao e primo\n", valor);
196         else
197             printf("%d e primo\n", valor);
198         printf("\n1 - Digitar outro valor\n2 - Sair\n");
199         scanf("%d", &opcao);
200     }while(opcao != 2);
201 }
```

Fonte: Elaborado pela autora, 2023.

O quarto exercício é iniciado na função laco4(). O programa exibe a tabuada de um número escolhido pelo usuário. Foi utilizado apenas um laço FOR, que determina um contador de variável i com valores de 0 a 10. A cada contagem o número digitado é multiplicado por i e exibido.

Figura 2.9 – Programa Tabuada



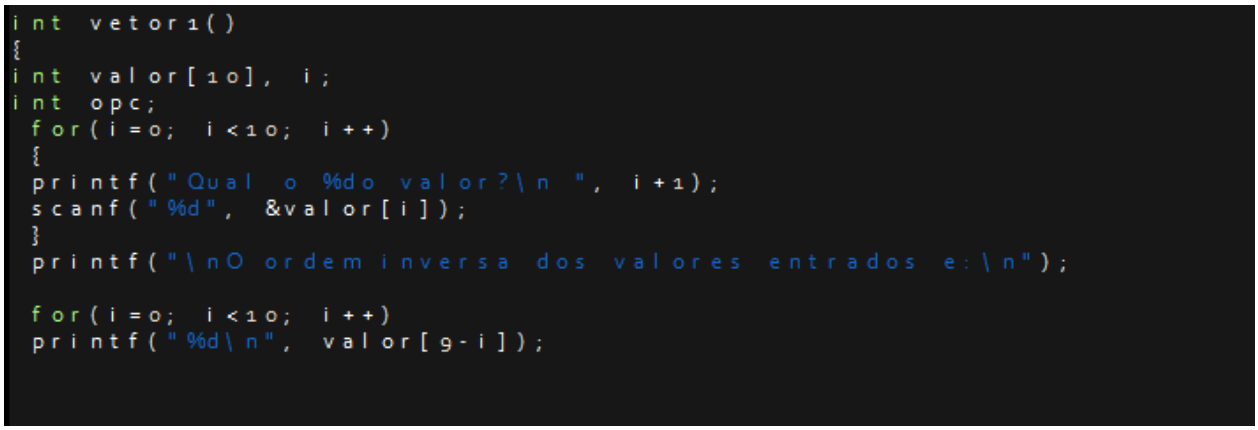
```
205  int laco4()//o4-Tabuada
206       {
207
208
209            int num,i;
210
211            printf("Digite um numero: ");
212                 scanf("%d", &num);
213            for(i=0;i<11;i++)
214            {
215
216               printf("%dx%d = %d\n", num, i, num * i);
217
218
219            }
220
221       }
222
```

Fonte: Elaborado pela autora, 2023.

2.6 EXERCÍCIOS DE VETORES

O primeiro exercício de vetores é iniciado na função vetor1(). O programa exibe a ordem inversa dos números armazenados no vetor valor[10] (vetor de nome valor com 10 posições). Um contador determinando números de 0 a 9 é utilizado para armazenar os números, incrementando cada. Para exibir esses valores ao contrário foi preciso utilizar o mesmo contador, mas a diferença está na hora de imprimir o resultado, começamos imprimindo pela última posição do vetor e decrementando números do contador.

Figura 2.10 – Programa exibe vetor ao contrário



```
int vetor1()
{
int valor[10], i;
int opc;
 for(i=0; i<10; i++)
 {
 printf("Qual o %do valor?\n ", i+1);
 scanf("%d", &valor[i]);
 }
 printf("\nO ordem inversa dos valores entrados e:\n");

 for(i=0; i<10; i++)
 printf("%d\n", valor[9-i]);
```

Fonte: Elaborado pela autora, 2023.

O segundo exercício de vetores é iniciado na função vetor2(). O programa pega os valores de dois vetores, armazena em um terceiro e salva os números em ordem crescente. Primeiro o programa solicita que o usuário preencha dois vetores com números de sua escolha cada vetor tem 5 posições. Um terceiro vetor de 10 posições foi criado e enquanto os valores são digitados e armazenados nos vetores vet1 e vet2, também está sendo diretamente gravado no vet3.

O vet3 é exibido com os números na ordem em que foram gravados para depois ser feito o processo para arrumar os números em ordem crescente. Para gravar os valores na ordem correta, foi criado um laço para ler o vetor dez vezes, é feito um laço para determinar o número das posições do vetor e é começada a leitura. Se o valor de posição i for maior que o valor da próxima posição, utilizamos uma variável auxiliar para fazer a troca dos valores, onde o valor menor é gravado na posição anterior.

Figura 2.11 – Programa exibe valores em ordem crescente

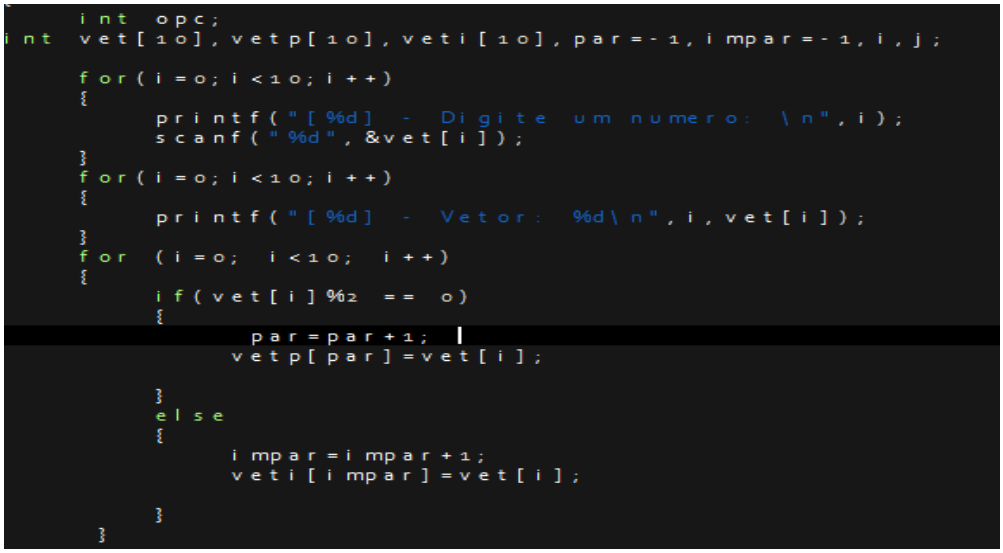
```
//PROCESSO CRESCENTE!!!!!!!!!!!!!!!

printf("\nEstes são os valores lidos em ordem crescente:\n");

        while ( ordenador <= 10 )
        {
                for ( i = 0; i < 10; i++ )
                {
                        if ( vet3[i] > vet3[i+1] )
                        {
                                aux = vet3[i];
                                vet3[i] = vet3[i+1];
                                vet3[i+1] = aux ;
                        }
                }

                ordenador = ordenador + 1;

        }

        for ( i = 0; i < 10; i++ ){

                printf("\nvetor 3[%d]: %d\n", i+1, vet3[i]);}
```

Fonte: Elaborado pela autora, 2023.

O terceiro exercício de vetores é iniciado na função vetor3(). O programa separa valores digitados em pares e ímpares. São solicitados 10 números ao usuário que são gravados em um vetor. Logo após é feito a leitura desse vetor onde a cada posição é verificado se o número é par, se for verdade, o valor correspondente ao

vetor e a posição são salvos no vetor que irá conter números pares. A posição do vetor de números pares recebe um incremento para que na próxima verificação, sendo verdadeiro que o número é par, o valor seja salvo na próxima posição. Sendo falso, o número é ímpar e o mesmo processo é repedido, dessa vez sendo salvo no vetor correspondente a números ímpares. Para finalizar os vetores pares e ímpares são impressos na tela.

Figura 2.12 – Programa separa valores pares e ímpares

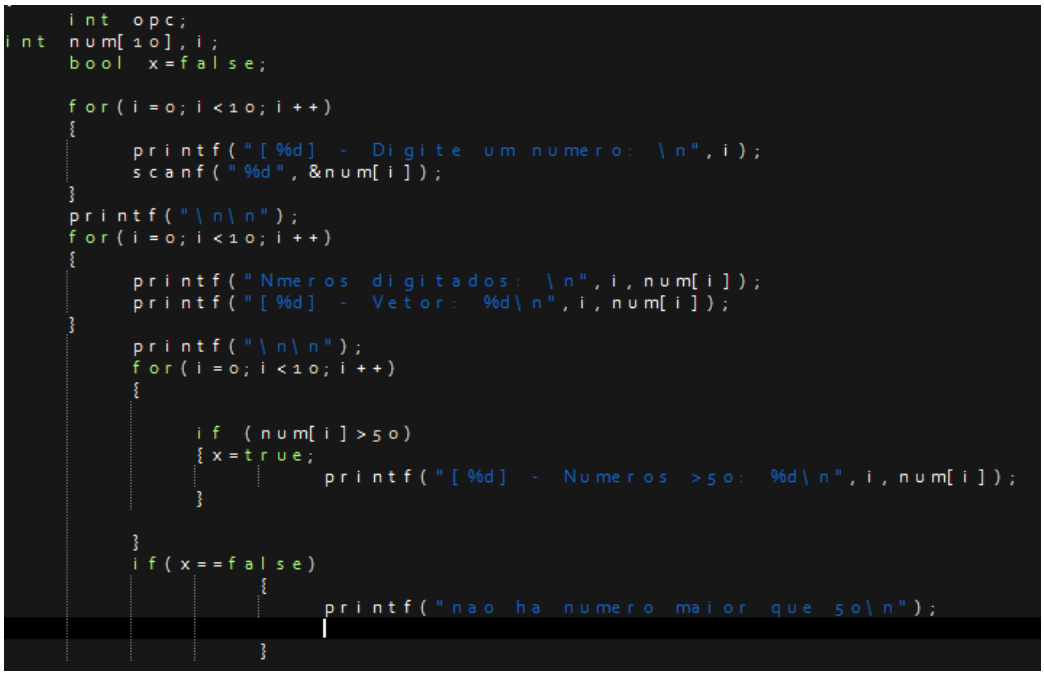


```
    int opc;
int vet[10],vetp[10],veti[10],par=-1,impar=-1,i,j;

    for(i=0;i<10;i++)
    {
        printf("[%d] - Digite um numero: \n",i);
        scanf("%d",&vet[i]);
    }
    for(i=0;i<10;i++)
    {
        printf("[%d] - Vetor: %d\n",i,vet[i]);
    }
    for (i=0; i<10; i++)
    {
        if(vet[i]%2 == 0)
        {
            par=par+1;
            vetp[par]=vet[i];

        }
        else
        {
            impar=impar+1;
            veti[impar]=vet[i];

        }
    }
```

Fonte: Elaborado pela autora, 2023.

O quarto exercício de vetores é iniciado na função vetor4(). O programa exibe valores maiores que 50. É iniciada uma variável de nome x do tipo booleana que recebe o valor false, em seguida é solicitado ao usuário 10 números que são gravados em um vetor. Esse vetor é exibido na tela com as posições e valores digitados pelo usuário. O programa verifica em cada posição através de um laço de repetição quais são os valores armazenados e se eles são maiores que 50. Se for verdade a variável x recebe o valor true e é exibido na tela qual a posição e o valor do número. Se no final do programa a variável x permanecer com o valor false, significa que não há número maior que 50.

Figura 2.13 – Programa exibe valores maiores que 50



```
    int opc;
int num[10],i;
    bool x=false;

    for(i=0;i<10;i++)
    {
        printf("[%d] - Digite um numero: \n",i);
        scanf("%d",&num[i]);
    }
    printf("\n\n");
    for(i=0;i<10;i++)
    {
        printf("Nmeros digitados: \n",i,num[i]);
        printf("[%d] - Vetor: %d\n",i,num[i]);
    }
        printf("\n\n");
        for(i=0;i<10;i++)
        {

            if (num[i]>50)
            {x=true;
                printf("[%d] - Numeros >50: %d\n",i,num[i]);
            }

        }
        if(x==false)
                {
                    printf("nao ha numero maior que 50\n");
                }
```

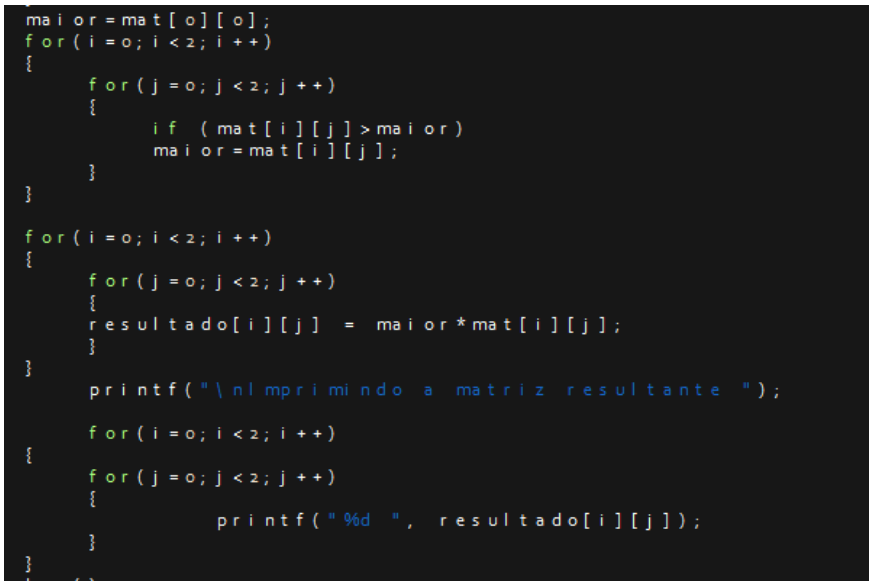
Fonte: Elaborado pela autora, 2023.

2.7 EXERCÍCIOS DE MATRIZES

O primeiro exercício de matrizes é iniciado na função matriz1(). O exercício irá verificar qual o maior número gravado em uma matriz e multiplicar os números por ele. Para gravar valores em uma matriz são necessários dois laços de repetição, um para incrementar as linhas e outro para as colunas.

Após estar definido e os valores armazenados na matriz, o programa determina que o maior número é aquele na linha 0 e coluna 0, a informação é armazenada na variável de nome maior. É iniciado um laço para percorrer todas as linhas e colunas da matriz para verificar se são de valor maior que a variável de mesmo nome. Caso seja encontrado um número maior, o valor da variável é substituído por ele. Um novo laço é iniciado, onde cada um dos elementos da matriz é multiplicado pelo maior número encontrado na etapa anterior e salvo em uma nova matriz.

Figura 2.14 – Programa multiplica matriz pelo maior valor

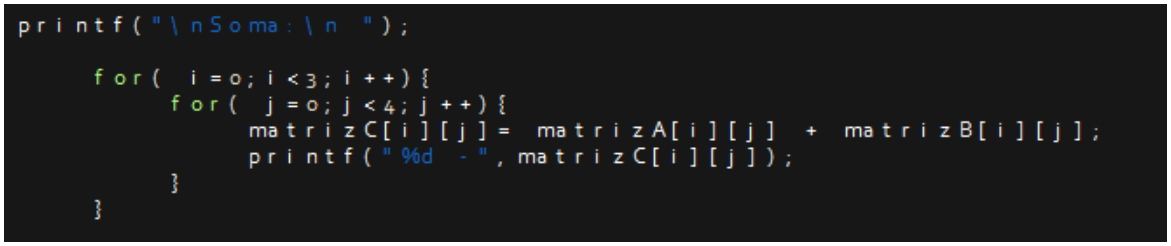


```
maior=mat[0][0];
for(i=0;i<2;i++)
{
      for(j=0;j<2;j++)
      {
            if (mat[i][j]>maior)
            maior=mat[i][j];
      }
}

for(i=0;i<2;i++)
{
      for(j=0;j<2;j++)
      {
      resultado[i][j] = maior*mat[i][j];
      }
}
      printf("\nImprimindo a matriz resultante ");

      for(i=0;i<2;i++)
{
      for(j=0;j<2;j++)
      {
                printf("%d ", resultado[i][j]);
      }
}
```

Fonte: Elaborado pela autora, 2023.

O segundo exercício de matriz é iniciado pela função matriz2(). O programa pega o valor gravado em duas matrizes e salva a soma em uma terceira. O usuário digita os valores para preencher duas matrizes 3x4. Após terminar de gravar as informações, é iniciado um laço para gravar as informações na terceira matriz. Como serão utilizadas as mesmas posições o mesmo laço é utilizado, somente determinando que a matriz recebera os valores da soma das outras duas matrizes

Figura 2.15 – Programa soma matrizes



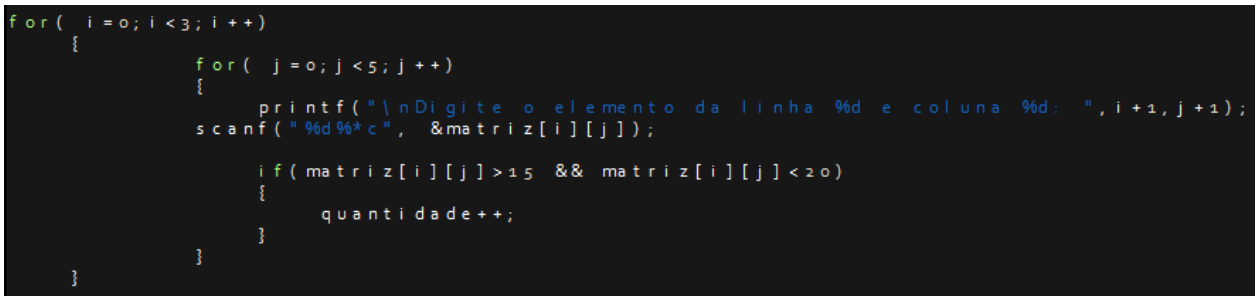
```
printf("\nSoma:\n ");

      for( i=0;i<3;i++){
            for( j=0;j<4;j++){
                  matrizC[i][j]= matrizA[i][j] + matrizB[i][j];
                  printf("%d -",matrizC[i][j]);
            }
      }
```

Fonte: Elaborado pela autora, 2023.

O terceiro exercício de matriz é iniciado pela função matriz3(). O programa exibe a quantidade de números entre 16 e 20 estão armazenados na matriz. O usuário digita os números a serem armazenados em uma matriz 3x5. É feita a leitura dessa matriz e verificada a cada posição se o valor armazenado esta entre 15 e 20, ou seja se ele é igual a 16, 17, 18 ou 19. Caso seja verdadeiro, é incrementado um número

ao contador. No final do programa é exibido o contador que mostra quantos números foram encontrados.

Figura 2.16 – Programa exibe a quantidade de números entre 16 e 20

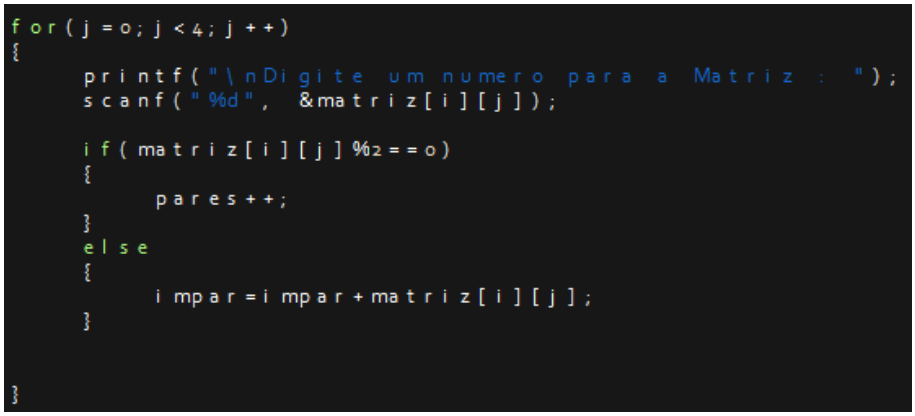


```
for(  i=0;i<3;i++)
    {
            for(  j=0;j<5;j++)
            {
                printf("\nDigite o elemento da linha %d e coluna %d: ",i+1,j+1);
            scanf("%d%*c",  &matriz[i][j]);

                if(matriz[i][j]>15  && matriz[i][j]<20)
                {
                    quantidade++;
                }
            }
    }
```

Fonte: Elaborado pela autora, 2023.

O quarto exercício de matriz é iniciado na função matriz4(). O programa faz uma contagem dos números pares e a soma dos números ímpares. Após o usuário informar os valores de uma matriz 3x4. O programa faz a leitura dos valores através de um laço de repetição, a cada valor é verificado se o mesmo é um número par, sendo verdade é incrementado 1 no contador de números pares, se for falso o número é ímpar, então é declarado que a variável de números ímpares é igual a ela mesma mais o valor validado. Por exemplo, a variável é iniciada valendo zero e recebe o valor ímpar validado no laço. Na próxima leitura, se o valor também for ímpar, será somado ao valor já salvo na variável utilizando: ímpar = ímpar + número. De outra forma (ímpar = número) o valor seria substituído e não somado. No final o programa exibe o contador de números pares e a somatória contida na variável ímpar.

Figura 2.17 – Programa contador de números pares e soma ímpares



```
for(j=0;j<4;j++)
{
      printf("\nDigite um numero para a Matriz : ");
      scanf("%d",  &matriz[i][j]);

      if(matriz[i][j]%2==0)
      {
            pares++;
      }
      else
      {
            impar=impar+matriz[i][j];
      }


}
```

Fonte: Elaborado pela autora, 2023.

Eu já tinha uma experiência superficial com a linguagem C, mas esta disciplina específica ampliou meu conhecimento, especialmente em vetores e matrizes. Adquiri habilidades práticas na declaração, inicialização e manipulação dessas estruturas de dados. Essa experiência não apenas fortaleceu minha base em C, mas também me capacitou para abordar desafios mais complexos na programação.

3 DISCIPLINA PROFISSIONAL DO 3º SEMESTRE: ESTRUTURA DE DADOS

Para a elaboração do trabalho foi utilizada a linguagem C. Desenvolvida na aula de Estrutura de dados, ministrada pelo professor Julio Fernando Lieira. A atividade compôs a nota do segundo bimestre e consiste no desenvolvimento de um programa que implementa um o sistema de uma livraria, onde é possível cadastrar e listar autores, livros, categorias e editoras, consultar e excluir categorias. O código esta disponível para download no portfólio digital.

3.1 SISTEMA LIVRARIA

A função principal, int main(), é onde a execução começa. Foram declaradas as variáveis op e cod do tipo inteira para armazenar a opção escolhida pelo usuário no menu, e o código que será usado na consulta. A variável tit é do tipo char, para armazenar títulos e palavras-chave. O programa entra em um loop e continua sendo executado até o usuário escolher a opção que o encerra. Um menu é exibido na tela com as opções para o usuário cada uma corresponde a uma função do programa. O switch é utilizado para executar o código que corresponde à opção.

Figura 3.1 – Menu Livraria

Fonte: Elaborado pela autora, 2023.

3.2 STRUCT

Na estrutura (struct) é possível agrupar variáveis de diferentes tipos em uma só unidade deixando os dados organizados. Foi definida uma estrutura chamada reg_livro para armazenar informações referentes ao livro como: código, titulo, codigoautor1, codigoautor2, codigoautor3, codigoautor4, codigocategoria, codieditora, edicao e estoque. Também foram definidas estruturas para armazenar informações do autor, a reg_autor com as variáveis: código, sobrenome e nome. A estrutura reg_categoria é a responsável por armazenar os dados das categorias dos livros com as variáveis: código e descricao. Para os dados das editoras foi usada a estrutura reg_editoras com as variáveis: código, nome, contato, fone e email.

Figura 3.2 – Struct Livraria

```
STRUCT REG_LIVRO{

   INT CODIGO;
   CHAR TITULO[30];
   INT CODIGOAUTOR1;
   INT CODIGOAUTOR2;
   INT CODIGOAUTOR3;
   INT CODIGOAUTOR4;
   INT CODIGOCATEGORIA;
   INT CODIEDITORA;
   INT EDICAO;
   INT ESTOQUE;

};
STRUCT REG_EDITORAS{

    INT CODIGO;
    CHAR NOME[20];
    CHAR CONTATO[20];
    CHAR FONE[20];
    CHAR EMAIL[20];
};
```

```
STRUCT REG_AUTOR{

   INT CODIGO;
   CHAR SOBRENOME[20];
   CHAR NOME[50];

};
STRUCT REG_CATEGORIAS{

    INT CODIGO;
    CHAR DESCRICAO[20];

};
```

Fonte: Elaborado pela autora, 2023.

3.3 CADASTROS LIVRARIA

É iniciada a função, que tem a finalidade de cadastrar livros. Foi declarada uma variável chamada fplivros que é um ponteiro para um arquivo, em seguida foi criada uma variável chamada livro que é usada para armazenar informações seguindo a estrutura que foi definida antes em reg_livro. Após o usuário preencher os dados solicitados no cadastro, é feita a confirmação se ele deseja gravar as informações,

caso a resposta seja negativa, o programa retorna ao menu inicial, se a resposta for afirmativa o arquivo livros.dat é aberto em modo de escrita binária e o ponteiro é associado ao arquivo. O modo “ab” significa append binary, que permite adicionar dados binários ao final do arquivo, mantendo o que já existia nele intacto. Caso o arquivo não exista, um novo é criado.

A Fwrite é uma função de escrita em arquivos, permite escrever um númeroespecífico de bytes a partir de uma localização de memória. “&livro” é um ponteiro para a estrutura livro no arquivo. Sizeof é utilizada para determinar o tamanho dos dados a serem gravados e o número 1 indica que apenas uma unidade do livro será escrita. Após os dados serem escritos o fclose fecha o arquivo. Uma linha de feedback fornece ao usuário uma mensagem de que os dados foram gravados.

Todas as funções de cadastro possuem a mesma estrutura e é possível um exemplo na imagem abaixo:

Figura 3.3 – Cadastrar Livro

```
void cadastrarLivro(){
  FILE *fpLivros;
  struct reg_livro livro;
  char opc;

  //Fazer a Entrada dos dados do Livro
  printf("\nCodigo: ");
  fflush(stdin); scanf("%i",&livro.codigo);
   printf("Titulo: ");
  fflush(stdin); gets(livro.titulo);
  printf("\nCodigo do autor 1: ");
  fflush(stdin); scanf("%i",&livro.codigoautor1);
  printf("\nCodigo do autor 2: ");
  fflush(stdin); scanf("%i",&livro.codigoautor2);
  printf("\nCodigo do autor 3: ");
  fflush(stdin); scanf("%i",&livro.codigoautor3);
  printf("\nCodigo do autor 4: ");
  fflush(stdin); scanf("%i",&livro.codigoautor4);
  printf("\nCodigo da categoria: ");
  fflush(stdin); scanf("%i",&livro.codigocategoria);
  printf("\nEditora: ");
  fflush(stdin); gets(livro.codieditora);
  printf("\nEdicao: ");
  fflush(stdin); scanf("%i",&livro.edicao);
printf("\nEstoque: ");
  fflush(stdin); scanf("%i",&livro.estoque);

 printf("\nGravar Livro(S/N)? ");
 fflush(stdin); scanf("%c",&opc);

 if ((opc=='N')||(opc=='n')){
   printf("\nOperacao Cancelada!");
   return;
 }

 //Abrir o Arquivo para acrescentar um Livro
 fpLivros = fopen(Livros,"ab");

 //Gravar no Arquivo
 fwrite(&livro,sizeof(livro),1,fpLivros);

 //Fechar o arquivo
 fclose(fpLivros);

 printf("\nLivro Gravado com sucesso.");

//Fim cadastrarLivro()
```

Fonte: Elaborado pela autora, 2023.

3.4 LISTAR

O arquivo livros.dat é aberto em modo de leitura binaria, “rb”, permitindo que o programa leia os dados do arquivo. A função fread lê uma unidade de dados, o loop while é executado e a função retorna 1 quando lê com sucesso e 0 quando alcança o final do arquivo. Dentro do loop o código imprime informações do livro lido no arquivo. Após imprimir as informações de um livro, o código continua lendo e imprimindo os próximos até o arquivo chegar ao final. Quando o loop termina o arquivo é fechado.

Todas as funções de listar possuem a mesma estrutura e é possível acompanhar na imagem abaixo um dos exemplos:

Figura 3.4 – Listar Livros

```
void listarLivros(){
  FILE *fplivros;
  struct reg_livro livro;

  //Abrir o Arquivo para Leitura
  fplivros = fopen(livros,"rb");

  while (fread(&livro,sizeof(livro),1,fplivros) ==1) {
     printf("\n\nCodigo: %i \nTitulo: %-30s \nAutor 01: %i
    \nAutor 02: %i  \nAutor 03: %i  \nAutor 04: %i
    \nCategoria: %i \n CodiEditora: %-30s\nEdicao: %i
    \nEstoque: %i \n----------------------------------------",
    livro.codigo, livro.titulo, livro.codigoautor1,
    livro.codigoautor2, livro.codigoautor3 , livro.codigoautor4
   , livro.codigocategoria, livro.codiEditora,
   livro.edicao, livro.estoque);

    }
     //Fechar o Arquivo
    fclose(fplivros);
} //Fim listarLivros()
```

Fonte: Elaborado pela autora, 2023.

3.5 EXCLUIR CATEGORIA

Para a exclusão foi criada a função excluirCategoria(). Foram declarados dois ponteiros de arquivo, o primeiro é usado para abrir o arquivo com as categorias existentes, o segundo cria um arquivo temporário com as categorias que ficarão salvas após a exclusão.

Uma variável de controle tem a função de verificar se a categoria digitada pelo usuário é existente no arquivo. A variável cod armazena a categoria a ser excluída e a variável op é responsável por guardar a resposta do usuário quando for questionado se deseja realmente prosseguir com a exclusão. Após o usuário digitar a categoria, o arquivo que contêm as informações é aberto em modo de leitura binária e um loop percorre o arquivo buscando a informação dada pelo usuário. Se encontrada, as informações referentes a aquela categoria são impressas na tela e a variável achou é incrementada. Após o loop percorrer o arquivo, o mesmo é fechado. Se a variável “achou” permanecer com seu valor zerado, significa que nenhum dado foi encontrado e uma mensagem é exibida, se encontrado é necessária a confirmação do usuário para a exclusão. Caso seja negada, o programa volta para o menu principal, quando confirmada a exclusão o arquivo categorias.dat é aberto em modo de leitura binária e o arquivo temporário, categorianew.dat é criado e aberto em modo de escrita binária. Um loop copia as informações do primeiro arquivo para o temporário, com exceção da categoria que deve ser excluída. O categorias.dat é substituído pelo categorianew.dat que é renomeado com o nome do primeiro. Uma mensagem informa o usuário que a categoria foi excluída.

Figura 3.5 – Excluir Categoria

```
void excluirCategoria(){
  FILE *fpcategorias, *fpcategoriasnew;
  struct reg_categorias categoria;
  int achou=0;
  int cod;
  char op;

  printf("\nDigite o Codigo da Categoria a ser Excluida: ");
  fflush(stdin); scanf("%i",&cod);

  fpcategorias = fopen(CATEGORIAS,"rb");

  while ((achou==0)&&(fread(&categoria,sizeof(categoria),1,fpcategorias) ==1)) {
     if (categoria.codigo == cod){
        printf("\nCodigo %6i \nDescricao %20s \n", categoria.codigo, categoria.descricao );
        achou=1;
     }
  }

  fclose(fpcategorias);

  if (achou==0){
     printf("\nNenhum Livro com o codigo %i foi Encontrado!!",cod);
     return;   //Retorna ao Menu Principal
  }

  //Se o Livro foi Localizado, continua aqui
  printf("\nConfirma Exclusao?(S/N) ");
  fflush(stdin); scanf("%c",&op);

  if ((op=='N')||(op=='n')){
      printf("\nOperacao Cancelada!");
      return; //retorna ao Menu Principal
  }

  //Se confirmou a exclusao, continua Aqui
  fpcategorias = fopen(CATEGORIAS,"rb");
  fpcategoriasnew = fopen("categoriasnew.dat","wb");

  while (fread(&categoria,sizeof(categoria),1,fpcategorias) ==1){
      if (categoria.codigo != cod){
          fwrite(&categoria,sizeof(categoria),1,fpcategoriasnew);
      }
  }
  fclose(fpcategorias);
  fclose(fpcategoriasnew);

  system("del Categorias.dat"); //Excluindo o arquivo antigo
  system("ren categoriasnew.dat Categorias.dat"); //Renomeia o novo arquivo

  printf("\nCategoria excluida com Sucesso.");

} //Fim excluirLivro()
```

Fonte: Elaborado pela autora, 2023.

3.6 CONSULTAR

A função permite que o usuário consulte livros cujo título contenha a palavra-chave específica. O arquivo livros.dat é aberto em modo de leitura binária, o código entra em loop verificando se o título do livro contém a palavra fornecida usando a funçãostrstr(). Se o título corresponde à condição, o livro é impresso na tela e a variável “achou” tem o mesmo propósito descrito anteriormente na função de excluir categorias. Após todos os livros serem verificados, uma mensagem é exibida se a busca não trouxe resultados e o arquivo é fechado.

Figura 3.6 – Consultar palavra no título

```
void consultarPalavraTitulo(char palavra[]){
FILE *fpLivro;
struct reg_livro livro;
int achou=0;

//Abrir o arquivo
fpLivro = fopen(livros."rb");

//Ler registro por registro e
// se o tÃtulo do livro lido contÃ©m a palavra-chave, imprimir
printf("\nCodigo Titulo                                    Preco");
while (fread(&livro, sizeof(livro), 1, fpLivro) ==1){
    if (strstr(livro.titulo, palavra) !=NULL){
        printf("\n%-6i  %-30s  %5.2f", livro.codigo, livro.titulo);
        achou++;
    }
}

if (achou==0){
    printf("\nNenhum Livro contem a palavra %s no Titulo.", palavra);
}
//Fechar o Arquivo
fclose(fpLivro);
}//Fim consultarPalavraTitulo()
```

Fonte: Elaborado pela autora, 2023.

Parecida com a função anterior, aconsultarInicialAutor() abre o arquivo autores.dat em modo de leitura e um loop percorre os registros utilizando a função strncmp(), são comparados os primeiros caracteres do autor com a parte inicial fornecida, se iguais, as informações do autor são impressas na tela e a variável “achou” desempenha a mesma função descrita anteriormente exibindo uma mensagem para o usuário caso a busca não tenha resultados. O arquivo é fechado após a conclusão da pesquisa

Figura 3.7 – Consultar pela inicial

```
void consultarInicialAutor(char palavra[]){
FILE *fpautores;
struct reg_autor autor;
int achou=0;

//Abrir o arquivo
fpautores = fopen(autores,"rb");

//Ler registro por registro e
// se o tÃtulo do livro lido contÃ©m a parte inicial, imprimir
while (fread(&autor,sizeof(autor),1,fpautores)==1){
      if (strncmp(autor.nome,palavra,strlen(palavra))==0){
            printf("\nNome: %-50s \nSobrenome: %-20s", autor.nome, autor.sobrenome);
            achou++;
      }
}

if (achou==0){
      printf("\nNenhum Autor contem a parte inicial %s no Nome.",palavra);
}
//Fechar o Arquivo
fclose(fpautores);
} //Fim consultarInicialTitulo(),
```

Fonte: Elaborado pela autora, 2023.

A função consultarLivroPorCategoria abre o arquivo Livros.dat em modo de leitura binária e um loop percorre cada registro. É verificado se o valor passado pelo parâmetro é correspondente ao código da categoria, caso encontre um livro com a categoria escolhida, as informações são impressas na tela

Figura 3.8 – Consultar por categoria

```
void consultarLivroPorCategoria(int cod){
FILE *fplivros;
struct reg_livro livro;

int achou=0;


fplivros = fopen(livros,"rb");
while ((achou==0)&&(fread(&livro,sizeof(livro),1,fplivros) ==1)) {
    if (livro.codigocategoria == cod){

        printf("\n\nCodigo da categoria: %i \nTitulo: %-30s",livro.codigocategoria,livro.titulo);
        achou=1;
    }
 }

 fclose(fplivros);

 if (achou==0){
   livro.codigocategoria = -1; //Livro Nao Localizado
 }



}
```

Fonte: Elaborado pela autora, 2023.

A função consultarLivroPorAutor() passa o código do autor como parâmetro, procurando e exibindo informações sobre todos os livros de um autor especifico.

Semelhante à anterior, a função consultarLivroPorAutor1() também procura e exibe informações sobre todos os livros associados a um autor, mas verifica em todos os campos referentes ao autor nos registros de livros.

Figura 3.9 – Consultar código do autor

```
VOID CONSULTARLIVROPORAUTOR(INT COD)
{

FILE *FPAUTOR;
STRUCT REG_AUTOR AUTOR;
INT ACHOU=0;
FPAUTOR = FOPEN(AUTORES,"RB");

     WHILE ((ACHOU==0)&&(FREAD(&AUTOR,SIZEOF(AUTOR),1,FPAUTOR) ==1))
     {
     IF (AUTOR.CODIGO == COD)
     {

PRINTF("CODIGO: %I        AUTOR: %-50S\N", AUTOR.CODIGO, AUTOR.NOME);
ACHOU=1;
                                    }
     }

FCLOSE(FPAUTOR);
     }

VOID CONSULTARLIVROPORAUTOR1(INT COD)
{

FILE *FPLIVROS;
STRUCT REG_LIVRO LIVRO;
 INT ACHOU=0;

FPLIVROS = FOPEN(LIVROS,"RB");
WHILE ((ACHOU==0)&&(FREAD(&LIVRO,SIZEOF(LIVRO),1,FPLIVROS) ==1))
     {
     IF ((LIVRO.CODIGOAUTOR1 == COD)||(LIVRO.CODIGOAUTOR2 == COD)||(LIVRO.CODIGOAUTOR3 == COD)
     ||(LIVRO.CODIGOAUTOR4 == COD))

     {

        PRINTF("\NLIVRO: %-30S",LIVRO.TITULO);
         ACHOU=1;
     }
}

  FCLOSE(FPLIVROS);
}
```

Fonte: Elaborado pela autora, 2023.

A disciplina de Estrutura de Dados representou um desafio particular, mas revelou-se extremamente enriquecedora. Adquiri uma compreensão mais profunda das funções, o que se refletiu no desenvolvimento do capítulo anterior. A capacidade de organizar todas as atividades por meio de um menu interativo foi um avanço. Explorei a manipulação de arquivos binários, uma habilidade que deixou evidente uma clara evolução. Anteriormente, meu envolvimento se limitava a projetos com menos de 50 linhas, e agora estou lidando com projetos que ultrapassam as 500 linhas.

4 DISCIPLINA PROFISSIONAL DO 4º SEMESTRE: PROGRAMAÇÃO AVANÇADA ORIENTADA A OBJETOS

O Arduino foi criado em 2005 por um grupo de 5 pesquisadores, Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. O desejo de ensinar eletrônica e programação à seus alunos unido com a falta de recursos acessíveis e com esquemas simplificados, fizeram surgir um objetivo, e assim Massimo elaborou o que serviria de base em seus projetos, uma ferramenta adaptável e de baixo custo. Com características inovadoras, foram vendidas mais de 50 mil unidades em seus primeiros anos, popularizando o Arduino.

É uma plataforma de prototipagem eletrônica, seu principal componente é um microcontrolador Atmel e circuitos de entrada e saída via IDE (ambiente de desenvolvimento integrado), composto por hardware e software. O software é desenvolvido por meio de linguagem c/c++ e seu ambiente gráfico é escrito em java. Utilizando componentes e sensores é possível automatizar a casa com fechaduras eletrônicas, luzes com sensor de movimento, sistemas de alarme ou até construir carrinhos controlados por controle, controles remotos, mão animatrônica, entre outros.

O nome da marca possui proteção de direitos autorais, mas seu código é open source, os usuários podem compartilhar toda a propriedade intelectual e soluções para aprimorar a plataforma.

Existem vários tipos de Arduino, desde opções básicas como o Arduino UNO até opções complexas como o Arduino DUE, a escolha do modelo depende da aplicação que será feita, considerando o número de portas necessárias e a complexidade do projeto. A empresa, atualmente dedica-se também no desenvolvimento de Arduino para uso industrial: Arduino Portenta, Arduino Nicla, Arduino Opta, Arduino Nano, Arduino MKR e Arduino Edge Control.

O microcontrolador é o cérebro do dispositivo, os programas recebidos são executados por ele, são avaliadas as portas de entrada e saída e seu desempenho é definido pelo “clock”. O Arduino UNO, por exemplo, tem o “clock” de 16MHz.

4.1 ESTRUTURA

O software do Arduino possui um editor de textos onde o código é inserido e é possível digitar, fazer testes, procurar e resolver erros e transferir para o dispositivo via cabo usb. Para a transferência, o Arduino deve ser conectado ao computador via cabo USB, após a transferência não é necessária a conexão via cabo para o microcontrolador executar o código, ele fica salvo em sua memória e o microcontrolador é capaz de executar desde que haja fonte de energia.

Com a IDE aberta, é possível iniciar um programa utilizando a estrutura básica que é composta por dois blocos: `setup()` e `loop()`. No bloco setup serão configuradas as opções iniciais como valores de variáveis, definir se uma porta será de entrada ou saída e escrever mensagens para o usuário. O bloco loop é responsável por repetir comandos, para interromper o loop é necessário um comando de pausa.

4.2 COMPONENTES

- Placa Arduino UNO
- Display OLED 0,96 128x64 Spi
- Mini Umidificador de Ar USB
- Sensor de Umidade e Temperatura DHT22
- Módulo Rele 1 canal- 5V/10ª
- Jumpers M/M e M/F
- Botão
- Protoboard

4.3 MINI UMIDIFICADOR DE AR

Como representação da disciplina de Programação Avançada Orientada a Objetos, aplicada pelo Prof. Me. Alciano Gustavo G. Oliveira foi escolhido o trabalho que compôs a nota final do semestre. A proposta do trabalho era considerar o que foi estudado sobre o microcontrolador Arduino e sua linguagem de programação, elaborar um projeto e simular uma aplicação real.

O projeto desenvolvido é um mini umidificador de ar controlado por um sensor de umidade e temperatura (código disponível para download no portfólio digital). No começo do código são importadas as bibliotecas: DHT (sensor de umidade e temperatura), SPI (Serial Peripheral Interface, utilizado para a comunicação rápida com dispositivos periféricos), Wire (permite enviar e receber dados em uma rede de dispositivos ou sensores), Adafruit_GFX (permite a exibição de elementos gráficos no display OLED, como pontos, linhas, triângulos, circulos) e Adafruit_ssd1306 (resolve a complexidade do controlador SSD1306, oferece comandos mais simples facilitando o controle do display OLED).

Figura 4.1 – Inclusão de biblioteca

```
//INCLUSÃO DE BIBLIOTECA

#include <DHT.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

Fonte: Elaborado pela autora, 2023.

Em seguida são declaradas as variáveis e as constantes, as variáveis que irão receber os valores de umidade e temperatura são declaradas com o valor igual a zero. Para o display são declaradas a altura e largura em pixels e as portas a serem utilizadas, esses parâmetros são passados ao display. O sensor recebe tratamento parecido, onde é declarado seu modelo e qual porta será utilizada, esses parâmetros são passados em uma função para o sensor DHT. O relé e o botão de ligar também possuem portas declaradas.

Figura 4.2 – Definição de parâmetros

```
float umidade = 0;
bool estadorele = 0;

#define SCREEN_WIDTH 128 // OLED display width em pixels
#define SCREEN_HEIGHT 64 // OLED display height em pixels
#define DHTPIN 12 // DHT22
#define DHTTYPE DHT22 //DEFINE O MODELO DO SENSOR

DHT dht(DHTPIN, DHTTYPE); //PASSA OS PARÂMETROS PARA A FUNÇÃO

#define OLED_MOSI  10 //sda
#define OLED_CLK   11 //sck
#define OLED_DC     8 //dc
#define OLED_CS     7 //cs
#define OLED_RESET  9 //res

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);
//PASSA OS PARÂMETROS PARA O DISPLAY

#define rele 13
#define botao 2
```

Fonte: Elaborado pela autora, 2023.

Para mostrar os valores de umidade no display OLED, foi criada uma função chamada printUmidade. A função é iniciada limpando os caracteres gravados no display para garantir que ele esteja pronto para receber os valores sem sobrescrever em dados já armazenados. O cursor, o tamanho e o textos a serem exibidos são declarados e efetivados no display.

Figura 4.3 – Função printUmidade()

```
//Escreve o texto no display com o valor da umidade lido pelo DHT22
void printUmidade() {

  display.clearDisplay(); // apaga todos os caracteres do display

  display.setCursor(45,10); //POSIÇÃO EM QUE O CURSOR IRÁ FAZER A ESCRITA
  display.setTextSize(1,2);
  display.print("UMIDADE"); //ESCREVE O TEXTO NO DISPLAY

  display.setCursor(30,30); //POSIÇÃO EM QUE O CURSOR IRÁ FAZER A ESCRITA
  display.setTextSize(2,4);
  display.print(umidade); //ESCREVE O TEXTO NO DISPLAY
display.print("%");
delay(100);
  display.display(); //EFETIVA A ESCRITA NO DISPLAY


}
```

Fonte: Elaborado pela autora, 2023.

A função botao_liga é responsável por iniciar todo o processo do umidificador. O umidificador está conectado a um relé que recebe o comando quando o botão é acionado. Se o relé estiver desligado ao botão ser pressionado o estado do relé é alterado, ou seja, de desligado ele vai para ligado e assim liga o umidificador.

Figura 4.4 – Função botao_liga()

```
void botao_liga(){
if (digitalRead(botao) == LOW) // Se o botão for pressionado
  {
    estadorele = !estadorele; // Troca o estado do rele
    digitalWrite(rele, estadorele);

    while (digitalRead(botao) == LOW);
    delay(100);

  }
```

Fonte: Elaborado pela autora, 2023.

Para desligar o umidificador foi utilizada uma função que faz a leitura para saber se o relé está ligado, sendo verdade ela verifica se a umidade é maior ou igual a 70%, se sim, o estado do rele é alterado, o mesmo então é desligado, desligando o umidificador conectado a ele.

Figura 4.5 – Função desliga()

```
void desliga(){
  if (umidade>=70){ //se a umidade for maior que 70
  estadorele = !estadorele; // troca o estado do rele
    digitalWrite(rele, estadorele);

    while (digitalRead(rele) == HIGH);
    delay(100);
    digitalWrite(rele, estadorele);

  }
}
```

Fonte: Elaborado pela autora, 2023.

Como comentado anteriormente, a função setup é responsável por definir se as portas são de entrada ou saída. Também foram inicializadas bibliotecas e tamanho e cor de texto foram definidas

Figura 4.6 – Função setup()

```
void setup(){
  Serial.begin(9600); //INICIALIZA A SERIAL
  dht.begin(); //INICIALIZA A FUNÇÃO

  pinMode(rele, OUTPUT);
  pinMode(botao, INPUT_PULLUP);
  Wire.begin(); //INICIALIZA A BIBLIOTECA WIRE
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //INICIALIZA O DISPLAY COM ENDEREÇO I2C 0x3C
  display.setTextColor(WHITE); //DEFINE A COR DO TEXTO
  display.setTextSize(1); //DEFINE O TAMANHO DA FONTE DO TEXTO
  display.clearDisplay(); //LIMPA AS INFORMAÇÕES DO DISPLAY
delay(100);

}
```

Fonte: Elaborado pela autora, 2023.

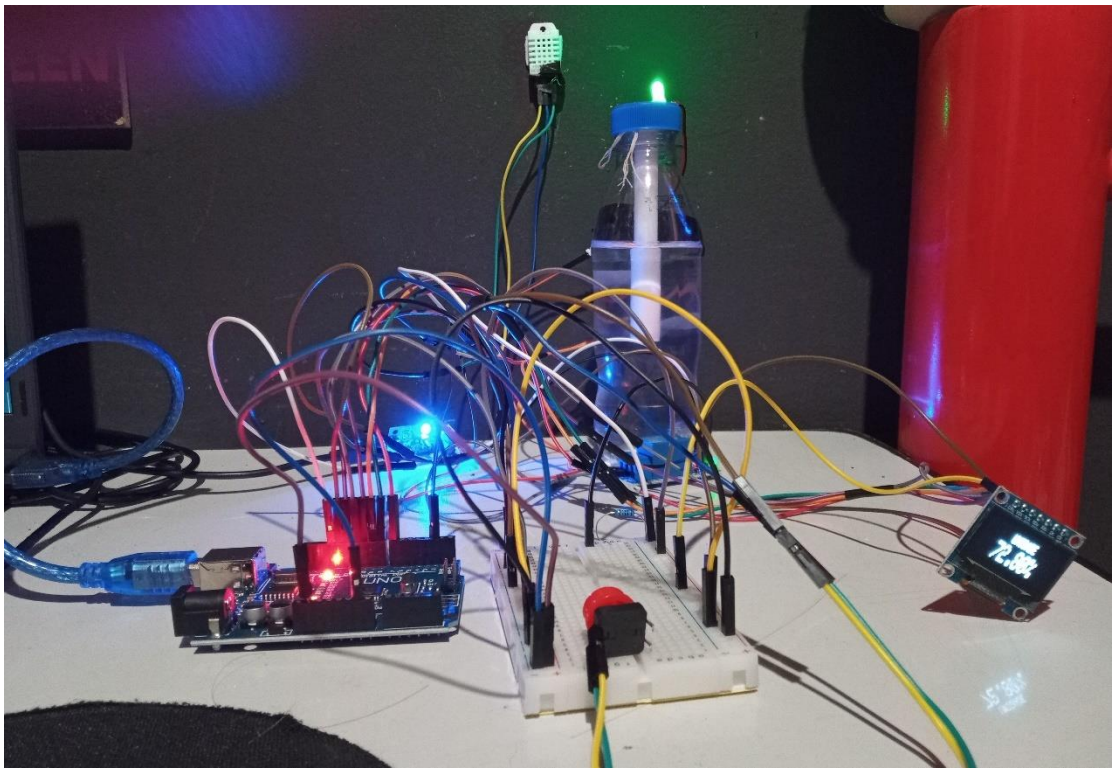
A função loop foi utilizada para declarar que a variável umidade recebe o valor lido pelo sensor e inicializar todas as funções.

Figura 4.7 – Função loop()

```
void loop()
{

umidade =dht.readHumidity();


botao_liga();
printUmidade();
desliga();
  delay(1000);
```

Fonte: Elaborado pela autora, 2023.

Figura 4.8 – Protótipo montado



Fonte: Elaborado pela autora, 2023.

Este projeto foi o que mais se destacou para mim, despertando meu fascínio pelo Arduino. Ao longo do desenvolvimento, atiçou minha curiosidade, e respondeu muitas dúvidas que eu tinha. A experiência de trabalhar com programação e testemunhar os resultados diretamente diante dos meus olhos foi única. À medida que mergulhava nos detalhes de cada componente, meu desejo por conhecimento crescia,

cada descoberta foi uma oportunidade para aprender mais. Este projeto alimentou minha paixão por explorar novas fronteiras na eletrônica e na programação.

5. DISCIPLINA PROFISSIONAL DO 5º SEMESTRE: PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

Como atividade avaliativa do primeiro bimestre, o professor Me. Felipe Maciel Rodrigues disponibilizou um site de sua autoria onde o objetivo era implementar algumas funções, que serão comentadas abaixo, utilizando HTML, CSS e JavaScript.

5.1 LINGUAGENS UTILIZADAS

HTML, ou HyperText Markup Language, é a linguagem de marcação usada para criar e estruturar páginas da web. Fornece um conjunto de elementos e tags que definem o conteúdo e a estrutura de uma página, como cabeçalhos, parágrafos, imagens, links e muito mais. O HTML é interpretado pelo navegador, que exibe o conteúdo de uma página da web de forma legível aos usuários. HTML é uma parte fundamental do desenvolvimento web e é frequentemente combinado com CSS (Cascading Style Sheets) e JavaScript para criar páginas web interativas e visualmente atraentes.

CSS é uma linguagem de estilo usada para controlar a apresentação e o design de páginas da web. Ele permite definir como elementos HTML são exibidos, como cores, fontes, espaçamento, posicionamento e muito mais. O CSS é essencial para separar o conteúdo (HTML) da apresentação (design) de uma página da web, o que facilita a manutenção e a personalização do visual das páginas. Além disso, o CSS pode ser aplicado de várias maneiras, como em folhas de estilo externas, embutidas no HTML ou inline diretamente nos elementos, oferecendo flexibilidade no controle visual das páginas web.

JavaScript é uma linguagem de programação de alto nível amplamente usada na web para tornar as páginas interativas. Ele permite adicionar comportamento dinâmico a páginas da web, como validação de formulários, animações, manipulação do DOM (Document Object Model) e comunicação com servidores. JavaScript é executado nos navegadores dos usuários e é uma parte essencial do desenvolvimento web moderno, permitindo a criação de aplicativos web complexos e responsivos. Além disso, é uma linguagem versátil que também pode ser usada no lado do servidor (Node.js) para desenvolvimento de aplicativos fora do navegador

5.2 INICIALIZAÇÃO

Quando o HTML está pronto, o código JS é executado. O uso do “$(document).ready(function ());” que envolve todo o código, garante a execução somente após o carregamento completo da página. Quando o elemento "btn-floating" é clicado, a função é chamada. Alguns eventos são acionados com essa função. “$('select').formSelect();” ativa o componente "Select" do Materialize CSS para aprimorar a aparência dos elementos <select>.

“$('.btn-floating').floatingActionButton();” ativa o componente "Floating Action Button" do Materialize CSS para melhorar a aparência dos botões flutuantes. “$("#cadastro").modal()” abre um modal com o ID "cadastro". O código mantém um contador de cidades com a variável contador iniciando em 1.

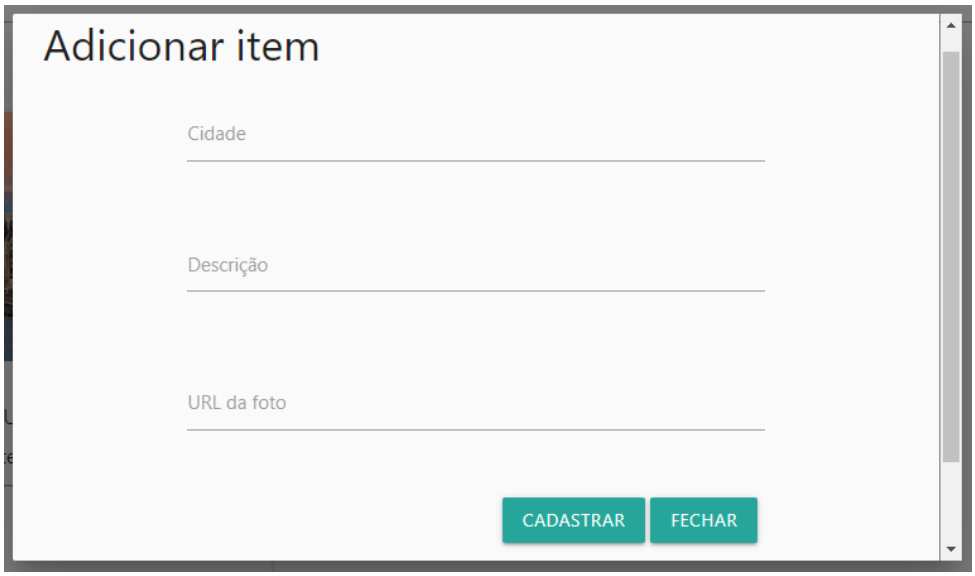
5.3 CADASTRAR CIDADE

Quando um elemento com o ID "btn-cadastrar" é clicado, a função associada a ele é chamada acionando os eventos.Os valores dos campos de entrada (cidade, descrição e foto) são coletados e um contador é incrementado.

Um novo bloco de HTML é gerado com os valores coletados e o contador atual, e é anexado à lista com o ID "lista-cidades".

Os campos de entrada são limpos.

Figura 5.1 – Tela de Cadastro



Fonte: Elaborado pela autora, 2023.

Figura 5.2 – Codigo cadastro

```
$("#btn-cadastrar").click(function () {
    let cidade = $('#cidade').val();
    let descricao = $('#descricao').val();
    let foto = $('#foto').val();
    contador=contador+1;
    let content = `<div class="row" id="lista-cidades">
        <div class="col s12 m6 l4">
            <div class="card">
             <div class="card-image waves-effect waves-block waves-light">
                <img class="activator" src="${foto}">
                    </div>
                <div class="card-content">
                    <span class="card-title activator grey-text text-darken-4">
                        #${contador} - ${cidade}<i class="material-icons right"> more_vert</i>
                    </span>
                    <p class="limit"> ${descricao}</p>
                    <hr>
                        <div class="btn-actions">
                            <i class="material-icons btn-editar"> edit</i>
                            <i class="material-icons btn-delete " id="del"> delete</i>
                        </div>
                    </div>
                    <div class="card-reveal">
                        <span class="card-title grey-text text-darken-4"> ${cidade}<i class="material-icons right"> close</i> </span>
                        <p class="break">
                            ${descricao}
                          </p>
                          </div>
                         </div>
                    </div>
                </div>
             </div>`;

    $('#lista-cidades').append(content);
    $('#cidade').val('');
    $('#foto').val('');
    $('#descricao').val('');
    $('select').val('');
    $('select').formSelect();
});
```

Fonte: Elaborado pela autora, 2023.

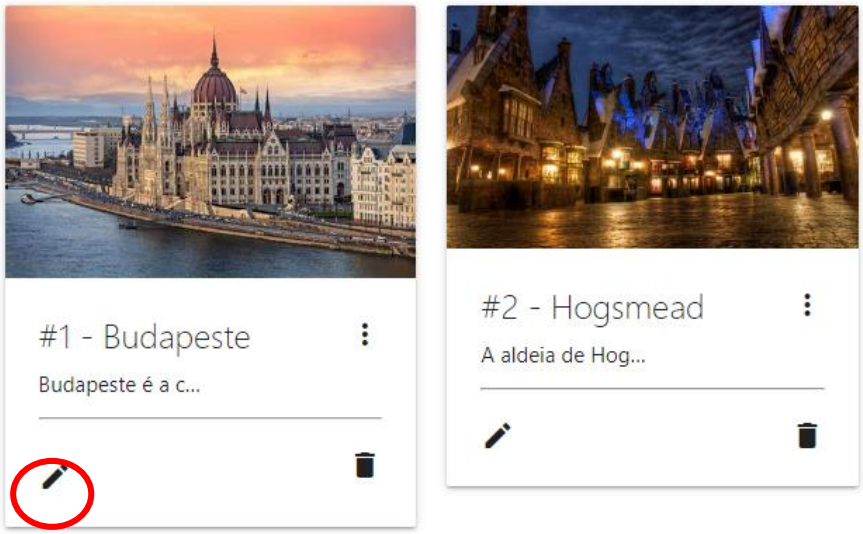
5.4 EDITAR CIDADE

Quando o elemento, dentro dos blocos de cidade, com a classe CSS "btn-editar" é clicado, a função é chamada. Nessa função o bloco da lista de cidades e o campo de pesquisa são ocultados. Informações da cidade selecionada são coletadas, como o número da cidade, a imagem, a cidade e a descrição. Essas informações são preenchidas nos campos de edição no formulário com IDs "id", "cidadeEdicao", “descricaoEdicao” e “fotoEdicao”. Um elemento com o ID "edicao" é exibido.

Quando um elemento do botão dentro do formulário de edição com o ID "btn-editar" é clicado a função é chamada. As informações editadas são coletadas dos campos de edição.

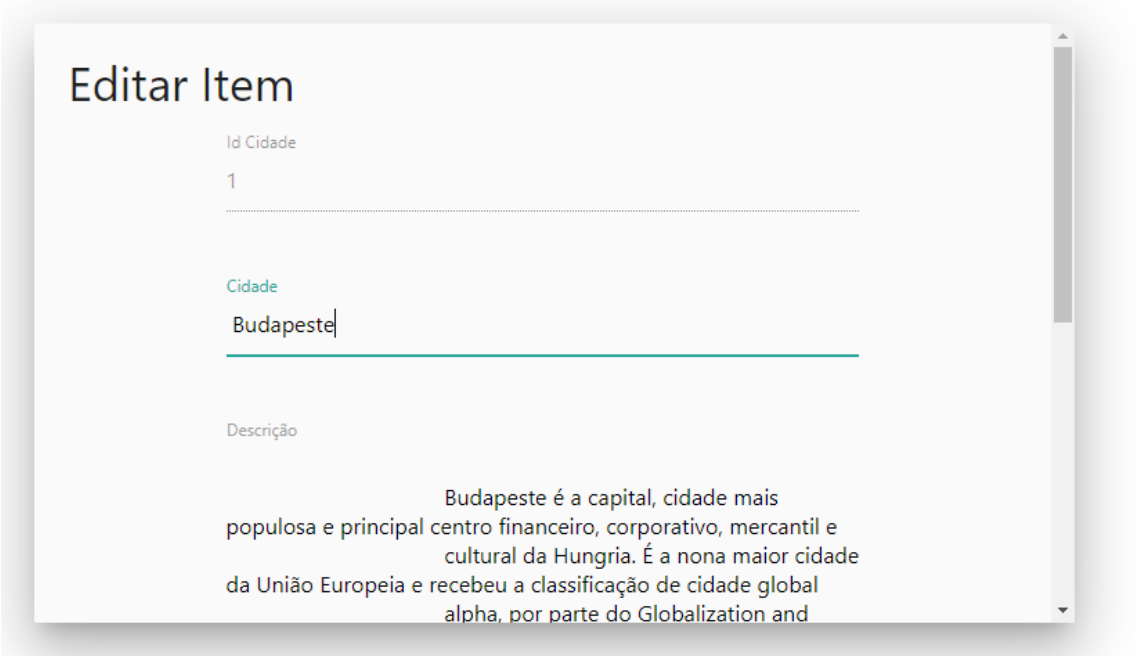
A cidade correspondente é encontrada na lista e as informações são atualizadas. O formulário de edição é ocultado, e a lista de cidades e o campo de pesquisa são exibidos novamente.

Figura 5.3 – Edição



Fonte: Elaborado pela autora, 2023.

Figura 5.4 – Tela de edição



Fonte: Elaborado pela autora, 2023.

Figura 5.5 – Código edição

```
$(document).on('click', '#btn-editar', function () {
    let numero = $('#id').val();
    let foto = $('#fotoEdicao').val();
    let cidade = $('#cidadeEdicao').val();
    let valDescricao = $('#descricaoEdicao').val();
    let resultado = cortarDescricao(valDescricao);

    console.log(resultado);

    let card = $('.card-title:contains("#' + numero + '")').parent().parent();
    card.find('img').attr('src', foto);
    card.find('.card-title').text('#' + numero + ' - ' + cidade);
    card.find('.card-reveal span.card-title').text(cidade);
    card.find('.card-content p:first').text(resultado);
    card.find('.card-reveal p').text(valDescricao);

    $('#edicao').hide();
    $('#lista-cidades').show();
    $('#pesquisar').show();

});
```

Fonte: Elaborado pela autora, 2023.

5.5 DELETAR CIDADE

Quando o botão de exclusão dentro dos blocos de cidade com o ID "del" é clicado a função é chamada. Nessa função, o bloco da cidade correspondente é removido da lista.

Figura 5.6 – Exclusão

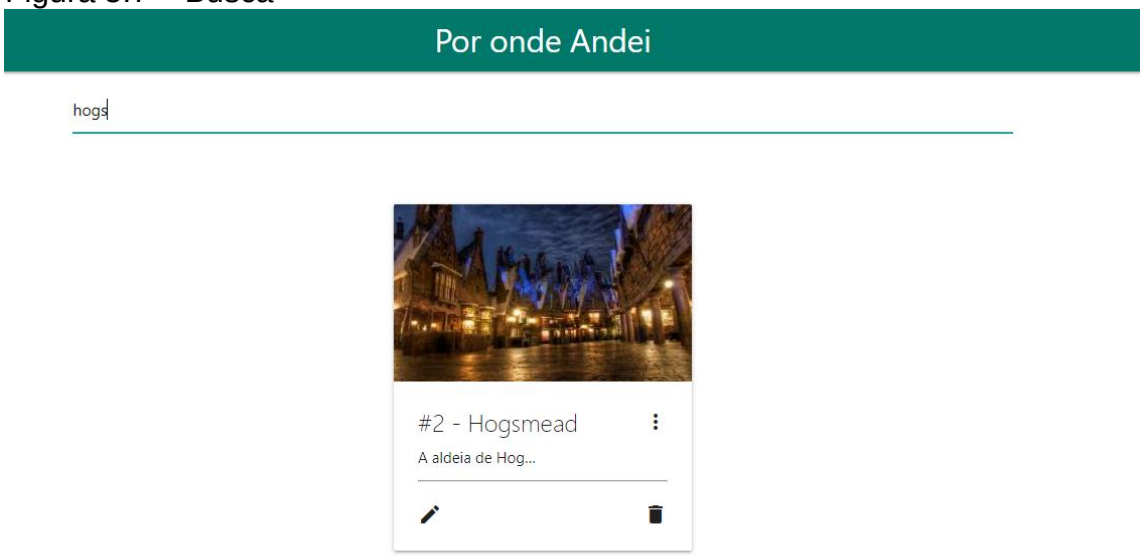
```
$('#lista-cidades').on('click', '#del', function () {
        $(this).parent().parent().parent().remove();

});
```

Fonte: Elaborado pela autora, 2023.

5.6 PESQUISAR CIDADE

Quando um usuário digita algo no campo de pesquisa com o ID "pesquisar", a função é chamada, O valor de pesquisa é coletado em letras minúsculas. Os títulos dos cartões de cidade são filtrados de acordo com o valor de pesquisa, ocultando os que não correspondem ao critério.

Figura 5.7 – Busca



Fonte: Elaborado pela autora, 2023.

Figura 5.8 – Código busca

```
$('#pesquisar').on('keyup', function () {
                var pesquisa = $(this).val().toLowerCase();
    $('#lista-cidades .card-title').filter(function () {
        $(this).parents('.card').toggle($(this).text().toLowerCase().indexOf(pesquisa) > -1)
    });
});
```

Fonte: Elaborado pela autora, 2023.

A disciplina revelou-se um desafio em diversas dimensões. Em muitos momentos, encontrei a necessidade de aprofundar ainda mais meu entendimento, realizando pesquisas e aulas adicionais. Tive a percepção de que esse esforço adicional seria crucial para o desenvolvimento do meu portfólio digital. Todo o esforço não foi somente para alcançar os objetivos do projeto, mas foi uma oportunidade para aprimorar minhas habilidades. Este projeto me fez perceber a importância de se dedicar integralmente à aprendizagem para atingir resultados.

6. DISCIPLINA PROFISSIONAL DO 6º SEMESTRE: TÓPICOS ESPECIAIS EM INFORMÁTICA

Para a elaboração do trabalho foram utilizadas as linguagens PHP e SQL junto com o framework Bootstrap, utilizado para implementar o front-end da aplicação que foi desenvolvida na aula de Tópicos especiais em informática, ministrada pelo professor Julio Fernando Lieira. A atividade compôs a nota do primeiro bimestre e consiste no desenvolvimento de uma aplicação em PHP para a manutenção de usuários incluindo as funcionalidades de cadastro, listagem, exclusão edição e controle de login.

6.1 PHP

O PHP (Hypertext Preprocessor) é uma linguagem de código aberto utilizada para o desenvolvimento de aplicações web. Ele é especialmente adequado para a criação de páginas dinâmicas e interativas.

Foi criado por Rasmus Lerdorf em 1994 como uma ferramenta para rastrear visitantes de sua página pessoal. Evoluindo posteriormente para uma linguagem de programação completa para o desenvolvimento web. O código PHP é executado no servidor web antes de ser enviado ao navegador do usuário.

Sua sintaxe é semelhante à de outras linguagens de programação, como C visto anteriormente. O código PHP é utilizado diretamente no HTML, o que torna fácil de integrar em páginas da web.

6.2 SQL

Foi inicialmente desenvolvido pela IBM nos anos 1970, o SQL (Structured Query Language), é uma linguagem de programação para a manipulação e gerenciamento de dados armazenados em um sistema de gerenciamento de banco de dados relacional. A linguagem é dividida em categorias como : DDL (Data Definition Language) com comandos utilizados para a definição da estrutura do banco de dados, como criar, modificar e excluir tabelas; DML (Data Manipulation Language) comandos que manipulam dados armazenados, como inserir, excluir e alterar registros; DQL (Data Query Language) para a consulta de dados; DCL (Data Control Language para o controle de acesso e permissões no banco de dados; TCL

(Transaction Control Language) para o gerenciamento de transações no banco de dados, como confirmar ou desfazer transações.

6.3 BOOTSTRAP

A construção do front-end da aplicação conta com o uso do framework de código aberto Bootstrap, uma criação conjunta do designer Mark Otto e do desenvolvedor Jacob Thornton, ambos ligados ao Twitter. Inicialmente denominado Twitter Blueprint quando ainda era um projeto interno da empresa, o Bootstrap foi posteriormente lançado publicamente em agosto de 2011.

O Bootstrap destaca-se por sua ênfase na responsividade, garantindo que as páginas desenvolvidas com ele se adaptem de forma automática a diversos dispositivos, proporcionando uma experiência de usuário coesa. Além disso, o framework disponibiliza uma variedade de estilos pré-construídos, facilitando a criação de páginas consistentes. Sua compatibilidade com os principais navegadores garante que a aplicação funcione de maneira uniforme em diferentes ambientes.

6.4 XAMPP

Para facilitar a configuração do ambiente de desenvolvimento local, foi utilizado o XAMPP, que consiste em um pacote de software livre e de código aberto que inclui servidor web, servidor de banco de dados e linguagens necessárias para as aplicações.

6.5 INDEX

A página principal contém uma tela de login, onde inicialmente faz a inclusão do cabeçalho onde o conteúdo está em ‘header.php’. Um formulário de login é exibido e ao clicar no botão ‘Entrar’ é iniciada a sessão e a verificação do método de requisição, é feita a conexão com o banco de dados e a verificação das credenciais do login utilizando uma consulta SQL para a verificação da existência de um usuário com as informações fornecidas. Caso exista um usuário `$_SESSION["loggedin"]` é definida como verdadeira e a pagina é

direcionada para a ‘home.php’, se a informação for falsa, uma mensagem de erro é exibida:

Figura 6.1 – Controle de login

Fonte: Elaborado pela autora, 2023.

Figura 6.2 – Erro de login

Fonte: Elaborado pela autora, 2023.

6.6 HOME

É iniciada a sessão que permite o uso das variáveis da pagina em PHP. A autenticação do usuário é feita e a variável da sessão `$_SESSION['usuario']` é verificada. Caso o usuário não esteja autenticado, a execução do script é interrompida e a página é redirecionada ao ‘index.php’ novamente. Caso contrário, os dados do

usuário são carregados do banco e são exibidos na tela o nome, o login, o e-mail e a permissão.

Figura 6.3 – Dados do usuário

Fonte: Elaborado pela autora, 2023.

6.7 CADASTRAR USUÁRIO

Um bloco de código condicional verifica se existe mensagem salva em uma variável para ser exibida. Caso haja mensagem ela é exibida e a variável de sessão é desarmada para não ser exibida novamente caso a página seja recarregada. Os campos para inserir os dados de cadastro são exibidos em um formulário HTML. Quando o formulário é enviado, os dados são encaminhados para o script PHP ‘salvarUsuario.php’.

Figura 6.4 – Cadastrar usuário

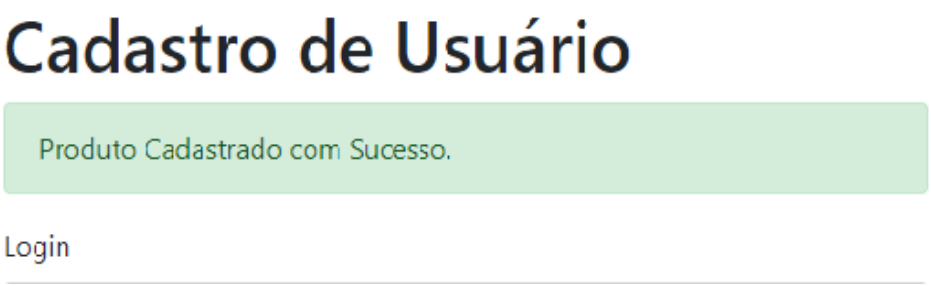
Fonte: Elaborado pela autora, 2023.

6.7.1 SALVAR USUÁRIO

Esse código é responsável por realizar o cadastro dos dados do usuário no banco de dados. A sessão é iniciada e o arquivo ‘sanitizar.php’ é incluído. Posteriormente vou falar sobre esse arquivo e sua função. Uma variável(‘$errovalidacao’) é utilizada para verificar se houve erro na validação. É conferido se o campo de login está vazio, se estiver a variável de erro é definida como verdadeira. Se ocorrer erro durante a validação, os dados são armazenados na sessão para ser recarregado no formulário e a página redireciona novamente ao formulário.

São definidas as credenciais de conexão, o banco é conectado e a conexão é verificada para saber se foi bem-sucedida. Os dados do formulário são recuperados e os valores são atribuídos as variáveis de login, nome, e-mail, senha e permissão. É construída uma instrução SQL para carregar os dados do usuário na tabela do banco de dados. É verificada a inserção de dados na tabela analisando o número de linhas, se maior que zero, é exibida uma mensagem de sucesso, caso contrário, é exibida uma mensagem de erro. A conexão com o banco é encerrada e a página é direcionada novamente ao formulário, contendo uma mensagem de retorno de erro ou sucesso no cadastro.

Figura 6.5 – Salvar usuário



Fonte: Elaborado pela autora, 2023.

6.8 SANITIZAR

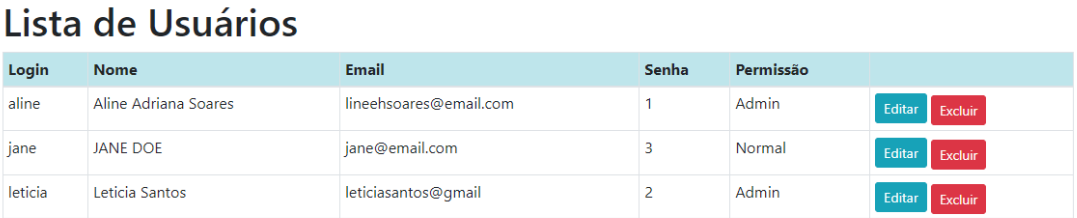
A função sanitizar($dados) tem o objetivo de realizar a validação e limpeza nos dados de entrada, o que é de extrema importância em aplicações web, para garantir a segurança protegendo contra ameaças de injeção de código. A função verifica se é um array, caso seja `foreach ($dados as $chave => $valor) {`: Itera sobre cada elemento

do array, onde `$chave` é a chave e `$valor` é o valor associado a essa chave. A função `filter_var` é usada com o filtro `FILTER_SANITIZE_STRING` para remover tags HTML e caracteres especiais dos dados da variável `$valor`. É removido qualquer prefixo hexadecimal, barras invertidas e armazena o valor limpo no array removendo também espaços em branco no início e no fim. No caso de a verificação constatar que não se trata de um array, o código realiza o mesmo processo porem, diretamente na variável `$dados`.

6.9 LISTAR USUÁRIO

São definidas as credenciais de acesso ao banco de dados, e é realizada a conexão, conforme explicação anterior. Após a conexão, uma consulta é realizada para selecionar todos os registros da tabela 'usuario'. Se o número de linhas no resultado for maior que 0, uma tabela HTML é exibida contendo colunas para 'Login', 'Nome', 'Email', 'Senha' e 'Permissão'. Um laço de repetição `while` percorre os resultados da consulta para exibir todos os dados na tabela. Os botões 'Editar' e 'Excluir' são incluídos para cada um dos usuários, com links que redirecionam para as páginas 'editarUsuario.php' e 'excluirUsuario.php'. Caso o número de linhas no resultado for igual a 0, exibe a mensagem "Nenhum Produto Encontrado." A conexão com o Banco de Dados é encerrada.

Figura 6.6 – Listar usuário



Fonte: Elaborado pela autora, 2023.

6.10 EDITAR

A sessão é iniciada e a sanitização é executada, o ID do usuário é extraído e armazenado. São definidas as credenciais de acesso ao banco, retornando uma mensagem de erro caso não seja possível a conexão. Uma consulta SQL é feita,

preparada para selecionar todos os dados do usuário onde o campo login for igual ao ID. É feita a verificação do resultado dessa consulta onde é comparado se o número de linhas é diferente de 1, caso não seja, uma mensagem de erro é exibida e o script é encerrado. Se o o resultado apontar que há apenas uma linha, os dados do usuário são recuperados e armazenados. A conexão com o banco é fechada, os dados armazenados na sessão e a página é direcionada para ‘editarUsuarioform.php'.

Figura 6.7 – Editar usuário

Fonte: Elaborado pela autora, 2023.

6.10.1 EDITAR USUÁRIO FORM

Um formulário é utilizado para a permissão da alteração dos dados. Os valores anteriores são exibidos nos campos. Mensagem de erro ou confirmação de exclusão são exibidas por esse script. O botão de salvar envia os dados para "editarUsuarioSalvar.php".

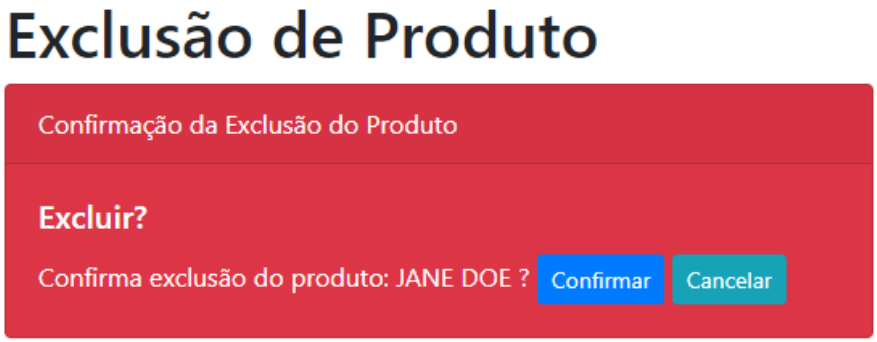
6.10.2 EDITAR USUÁRIO SALVAR

É feita a sanitização e validação dos dados, caso existam dados de formulário na sessão, eles são removidos e se houver erro na validação, o script é redirecionado para o formulário de edição. Após a conexão com o banco de dados, é construída uma consulta SQL para atualizar os dados na tabela do banco de dados. Uma consulta é feita para verificar se algum registro foi alterado e uma mensagem de erro ou de sucesso é exibida.

6.11 EXCLUIR

A sessão é iniciada e a sanitização é feita. A partir dos dados sanitizados é obtido o ID do usuário, os dados são armazenados na sessão e a página é direcionada para 'excluirUsuarioForm.php'.

Figura 6.8 – Excluir usuário



Fonte: Elaborado pela autora, 2023.

6.11.1 EXCLUIR USUÁRIO FORM

Um painel é exibido para a confirmação da exclusão contendo dois botões. O botão de confirmação envia o formulário para o script "excluirUsuarioExclusao.php". o botão de cancelar é um link que direciona o usuário para a página ‘listarUsuario.php.

6.11.2 EXCLUIR USUÁRIO EXCLUSÃO

A sessão é iniciada, a sanitização é feita e o ID do usuário é obtido. A conexão com o banco de dados é feita retornando uma mensagem de erro caso haja falha. Uma consulta é feita no banco para a exclusão dos dados com base no ID. Uma consulta é realizada para saber se se algum registro foi alterado, se alguma linha tiver sido afetada uma mensagem de sucesso é exibida, caso contrário uma mensagem de erro é definida. A conexão com o banco é encerrada e a página é redirecionada para ‘listarUsuario.php’

O PHP fazia parte do meu dia a dia quando concluí meu curso técnico em Informática para a Web na Etec de Lins. A ausência de prática e contato com a linguagem diminuiu

significativamente meu domínio sobre ela. Redescobrir a importância do PHP foi um desafio empolgante. O código do projeto está disponível para download no portfólio digital.

7 MANUAL DO USUÁRIO DO PORTFOLIO

Nesse capítulo é apresentado o manual para a utilização do Portfolio Digital de Análise e Desenvolvimento de Sistemas. Trata-se de um Web Site que contém os trabalhos acadêmicos, contato, currículo e redes sociais para demonstrar o aprendizado.

O objetivo do portfolio é destacar os conhecimentos refletindo as conquistas durante a trajetória acadêmica.

7.1 FUNCIONALIDADES

Para o desenvolvimento do portfólio, foram empregadas diversas ferramentas especializadas. Para a construção das páginas web, foi usado o HTML5, enquanto o design e estilo dessas páginas foram moldados com CSS3, a responsividade do layout, foi assegurada pela utilização do framework web Bootstrap 4, integrando HTML, CSS e JavaScript de maneira eficiente. Para criar e editar as páginas, o Visual Studio Code foi utilizado.

Cada uma dessas escolhas e ferramentas desempenhou um papel crucial na concretização do portfólio pessoal, contribuindo para que ele atingisse a forma desejada.

7.2 MENU PORTFOLIO

A barra de navegação foi definida com classes do Bootstrap que ajudam a estilizar, posicionar e fixar a barra. "HOME" é o texto exibido como um link que traz o usuário de volta ao topo da página. Caso o site seja acessado de um dispositivo menor, como um celular por exemplo, os itens de navegação são exibidos ou ocultados utilizando os atributos “data-bs-toggle” e “data-bs-target”, através de um botão de alternância.

Figura 7.1 – Barra de navegação

Fonte: Elaborado pela autora, 2023.

7.3 PÁGINA INICIAL

A página inicial contém uma seção de introdução com uma imagem pessoal, uma mensagem de boas-vindas e um botão que leva a outra seção da página. É representada com a estrutura de um cabeçalho “<header>” onde é apresentada a introdução pessoal. A largura e o espaçamento são controlados por um contêiner. Um outro contêiner foi utilizado para envolver uma imagem e aplicar um estilo para deixá-la circular. O texto exibido é um título de nível 1 que utiliza css para ajustar o sombreamento e estilos de cor e alinhamento específicos. Um botão é utilizado como link que aponta para a seção de trabalhos acadêmicos da página.

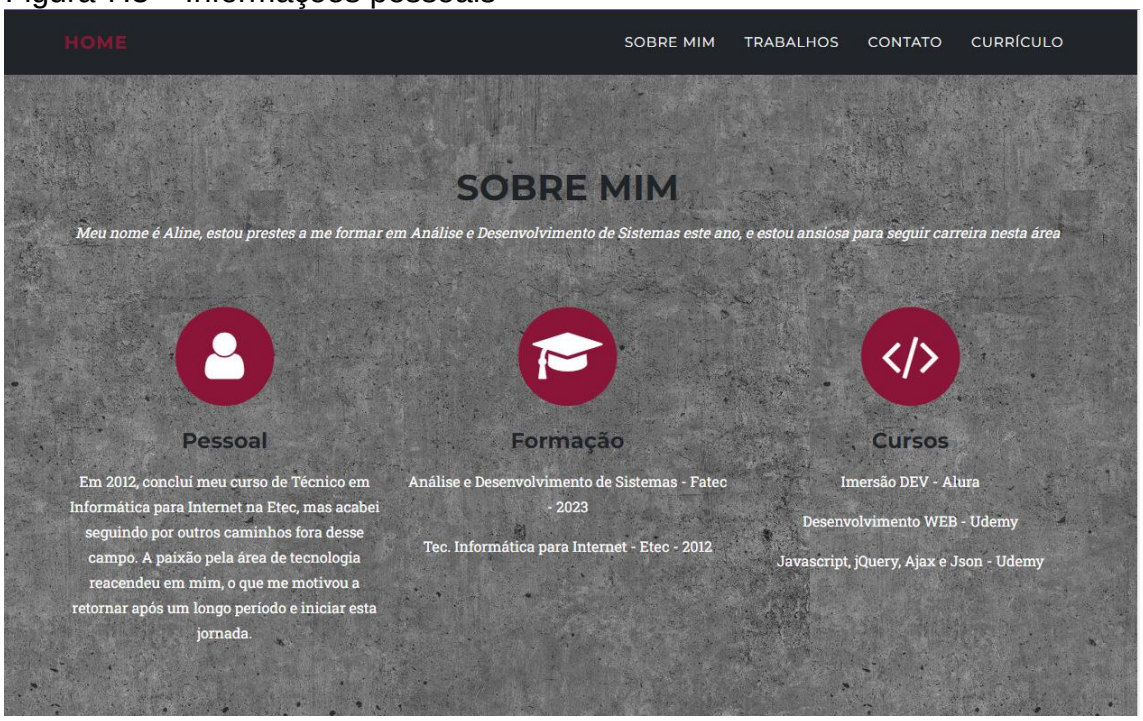
Figura 7.2 – Página inicial

Fonte: Elaborado pela autora, 2023.

7.4 INFORMAÇÕES PESSOAIS

Na seção “Sobre mim”, também foi utilizado um contêiner para envolver uma imagem e aplicar um estilo para deixá-la circular. As imagens dentro do círculo foram introduzidas através de uma biblioteca e kit de ferramentas de ícones da Internet chamada Font Awesome. Para as informações de formação e cursos, foi utilizada uma lista ordenada.

Figura 7.3 – Informações pessoais

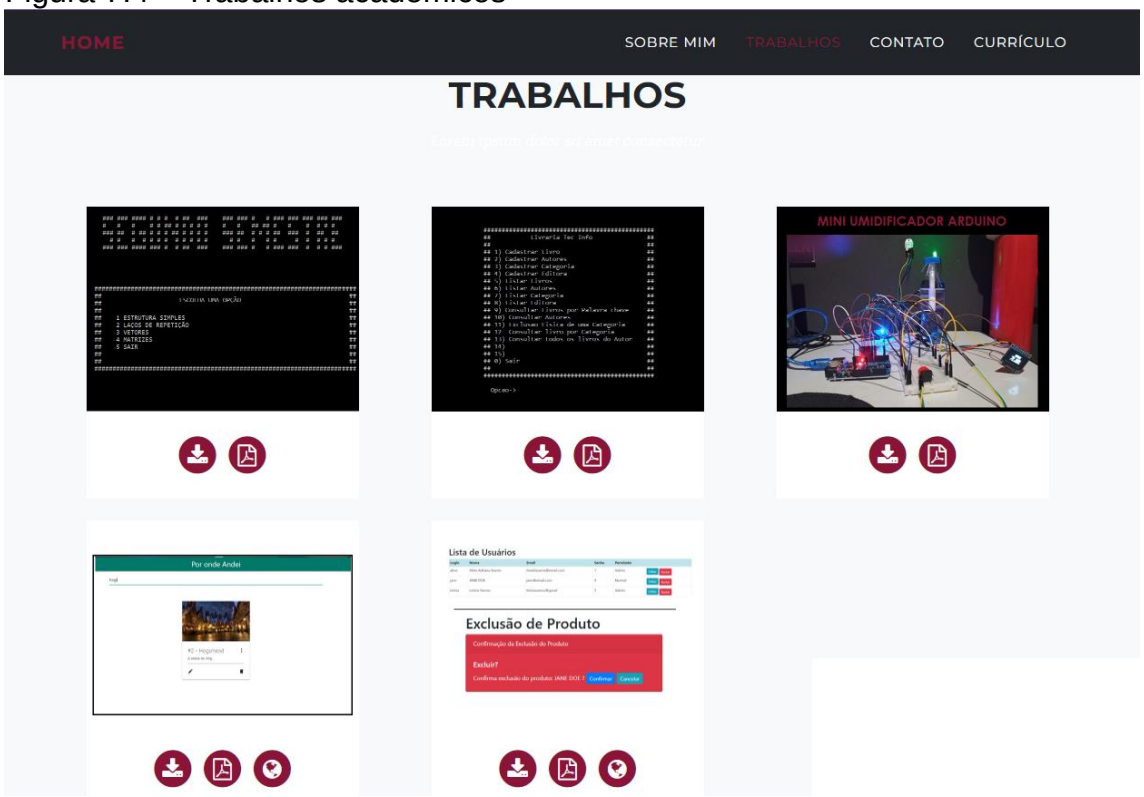


Fonte: Elaborado pela autora, 2023.

7.5 TRABALHOS

Todas as atividades são representadas por uma imagem, ao passar o mouse sobre a imagem, uma sobreposição com um ícone de adição é exibida, indicando que há mais informações disponíveis. Ao clicar no ícone, as informações daquele trabalho são exibidas em um modal. Abaixo da imagem foram inseridos ícones com link para baixar o capítulo referente a disciplina e o projeto. Os trabalhos do quinto e sexto semestre, por serem aplicações web, possuem um ícone a mais que contém o link para a página desenvolvida

Figura 7.4 – Trabalhos acadêmicos



Fonte: Elaborado pela autora, 2023.

Figura 7.5 – Modal

Fonte: Elaborado pela autora, 2023.

7.6 CONTATO

Na seção de contato, foi disponibilizado um formulário que inclui campos para o preenchimento do nome, e-mail e mensagem, permitindo que os usuários estabeleçam contato. Foi utilizado o FormSubmit, um serviço online que recebe e processa os dados enviados pelo usuário e envia para o meu e-mail pessoal através do atributo ‘action’ do formulário. Para a autenticação dos dados, uma chave se acesso foi inserida em um campo escondido:

```
<form action=’https://formsubmit.co/lineehsoares20@gmail.com’>
<input type="hidden" name="accessKey" value="75cdb089-1c89-4bdb-91de-dde2d621a2e3"/>
```

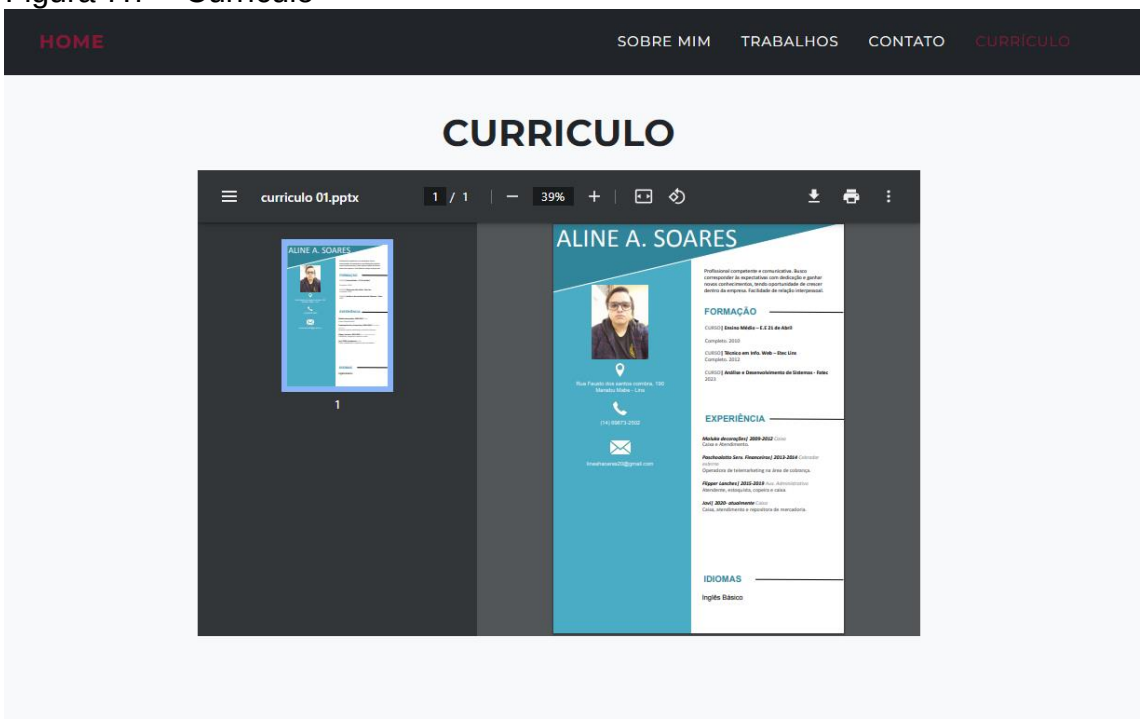
Figura 7.6 – Contato

Fonte: Elaborado pela autora, 2023.

7.7 CURRÍCULO

O currículo é exibido através de um visualizador de PDF onde é possível fazer o download dele. Foi utilizado o elemento “<object>” que é comum para incorporar documentos, como PDFs.

Figura 7.7 – Currículo



Fonte: Elaborado pela autora, 2023.

7.8 FOOTER

O rodapé é uma organização visualmente estruturada com diferentes elementos na página, incluindo informações sobre o desenvolvedor, links para redes sociais e um logotipo da Fatec. Os ícones das redes sociais foram adicionados utilizando os recursos do Font Awesome e tratados com css para mudar de cor ao passar o mouse em cima, cada ícone leva a rede social especifica. A imagem da Fatec é um link que direciona para a página da instituição

Figura 7.8 – Footer

Fonte: Elaborado pela autora, 2023.

8 CONSIDERAÇÕES FINAIS

Finalmente, o propósito deste projeto foi destacar o processo de aprendizado que me capacitou na elaboração do meu próprio portfólio digital, utilizando os conhecimentos adquiridos durante o curso de análise e desenvolvimento de sistemas. A intenção era apresentar, de maneira prática, meu perfil acadêmico e profissional, evidenciando descobertas, a aplicação de técnicas de software, linguagens de programação e frameworks.

Ao examinar o portfólio, é perceptível a evolução em relação às disciplinas, com projetos que não apenas aprimoraram em qualidade, mas também agregaram valor com a experiência acumulada a cada semestre. As experiências com ferramentas, métodos e linguagens de programação foram claramente evidenciadas, refletindo o aprendizado ao longo do tempo.

O design do meu portfólio digital profissional foi concebido com o intuito de oferecer páginas responsivas, uma interface leve e segura, proporcionando uma navegação fácil aos usuários. A acessibilidade foi uma prioridade em todas as páginas, com o menu que acompanha a página, facilitando o retorno ao topo. Disponibilizei acesso a todas as informações das atividades, capítulos em formato de PDF, o projeto para download e links das atividades desenvolvidas em formato de site. Na seção do rodapé, utilizei links para facilitar a interação do usuário, redirecionando para as páginas das redes sociais. Praticamente todas as páginas incluem botões de acesso que direcionam para outras partes do portfólio digital.

A construção desse projeto proporcionou valores significativos, experiências enriquecedoras e um amplo conhecimento, especialmente nas áreas de análise e desenvolvimento de sistemas. Dessa forma, foi possível compreender o propósito inicial do projeto, sempre voltado para o aprendizado e conhecimento, demonstrando minha capacidade individual ao desenvolver e concluir meu próprio portfólio digital profissional com o objetivo de apresentar meu perfil acadêmico e profissional.

9 REFERÊNCIAS BIBLIOGRÁFICAS

BALLERINI, R. **HTML, CSS e Javascript, quais as diferenças?** Disponível em: https://www.alura.com.br/artigos/html-css-e-js-definicoes. Acesso em: 6 set. 2023.
CASAVELLA, E. O que é Linguagem C? Disponível em: https://linguagemc.com.br/o-que-e-linguagem-c/. Acesso em: 18 mai. 2023.

DIAS, V. **PHP: conceitos, lidando com dados, loops e mais.** Disponível em: https://www.alura.com.br/conteudo/php-primeiros-passos?utm_term=&utm_campaign=%5BSearch%5D+%5BPerformance%5D+-+Dynamic+Search+Ads+-+Artigos+e+Conte%C3%BAdos&utm_source=adwords&utm_medium=ppc&hsa_acc=7964138385&hsa_cam=11384329873&hsa_grp=111087461203&hsa_ad=682526577071&hsa_src=g&hsa_tgt=aud539280195484:dsa-810524869174&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=CjwKCAiAjrarBhAWEiwA2qWdCKns5caJBnLKcC4qLS0W4tmDf6W3JnKyFopynPxqVvp5fy-hhmDEbxoC7LQQAvD_BwE. Acesso em: 8 out. 2023.

KRIGER, D. **10 COMANDOS SQL QUE TODO PROGRAMADOR TEM QUE CONHECER!** Disponível em: https://kenzie.com.br/blog/comandos-sql/?utm_source=adwords&utm_campaign=TRAFFIC-DIST-BLOG-SEARCH&utm_term=o%20que%20%C3%A9%20sql&utm_medium=google-ads&hsa_cam=20546388451&hsa_grp=153740622877&hsa_ad=674639067838&gad_source=1&gclid=CjwKCAiAjrarBhAWEiwA2qWdCGt9-7oEPt9HCbq9fPyQ94DIvcJeN1w62uXNEsd2UPfdZsT6uRlYMxoCsekQAvD_BwE. Acesso em: 23 out. 2023.

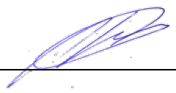
MAKIYAMA, M. **O que é arduino, para que serve, benefícios e projetos**. Disponível em: https://victorvision.com.br/blog/o-que-e-arduino/. Acesso em: 16 abr. 2023.

VAZ, J. **O que é linguagem C.** Disponível em: https://ebaconline.com.br/blog/o-que-e-linguagem-c#:~:text=A%20linguagem%20C%20existe%20desde,%C3%A9%20aplicada%20para%20criar%20softwares. Acesso em: 21 mai. 2023.

Bootstrap. Disponível em: https://getbootstrap.com.br/docs/4.1/about/overview/

Acesso em: 09 dez. 2023.

Página de assinaturas



Anderson Pazin
264.548.978-85
Signatário

Fernando Muzzi
145.859.038-01
Signatário

Julio Lieira
080.769.668-41
Signatário

HISTÓRICO

19 dez 2023 11:22:07 — **Anderson Pazin** criou este documento. (E-mail: anderson.pazin@fatec.sp.gov.br, CPF: 264.548.978-85)

19 dez 2023 11:22:07 — **Anderson Pazin** *(E-mail: anderson.pazin@fatec.sp.gov.br, CPF: 264.548.978-85)* visualizou este documento por meio do IP 177.95.133.194 localizado em Guaratingueta - Sao Paulo - Brazil

19 dez 2023 11:22:10 — **Anderson Pazin** *(E-mail: anderson.pazin@fatec.sp.gov.br, CPF: 264.548.978-85)* assinou este documento por meio do IP 177.95.133.194 localizado em Guaratingueta - Sao Paulo - Brazil

20 dez 2023 15:34:08 — **Julio Fernando Lieira** *(E-mail: julio.lieira3@fatec.sp.gov.br, CPF: 080.769.668-41)* visualizou este documento por meio do IP 189.126.178.186 localizado em Lins - Sao Paulo - Brazil

20 dez 2023 15:34:24 — **Julio Fernando Lieira** *(E-mail: julio.lieira3@fatec.sp.gov.br, CPF: 080.769.668-41)* assinou este documento por meio do IP 189.126.178.186 localizado em Lins - Sao Paulo - Brazil

19 dez 2023 11:48:59 — **Fernando Augusto Garcia Muzzi** *(E-mail: fernando.muzzi@fatec.sp.gov.br, CPF: 145.859.038-01)* visualizou este documento por meio do IP 191.187.108.173 localizado em Marília - Sao Paulo - Brazil

19 dez 2023 11:50:20 — **Fernando Augusto Garcia Muzzi** *(E-mail: fernando.muzzi@fatec.sp.gov.br, CPF: 145.859.038-01)* assinou este documento por meio do IP 191.187.108.173 localizado em Marília - Sao Paulo - Brazil