

**Sistema de Gestão de Estudos e Monitoria para Alunos**



Interdisciplinar: Análise e Desenvolvimento de Sistemas

Turno: Noite

Turma B

# **Documentação do Sistema de Gestão de Estudos e Monitoria para Alunos**

## **1. Introdução**

A **Gestão de Estudos e Monitoria para Alunos** é um sistema desenvolvido em linguagem C, destinado a otimizar a interação entre alunos e monitores, proporcionando um ambiente onde questões de estudo podem ser registradas, resolvidas e geridas eficientemente. Este sistema foi projetado para atender instituições de ensino que desejam facilitar a gestão de atividades acadêmicas, melhorar o desempenho dos alunos e potencializar a comunicação entre alunos e monitores.

### **1.1 Objetivos do Sistema**

Os principais objetivos do sistema são:

* **Facilitar o Registro de Questões:** Permitir que os alunos registrem suas dúvidas e questões que necessitam de solução, contribuindo para um aprendizado mais ativo.
* **Gerenciar Monitores:** Cadastrar monitores e gerenciar suas disciplinas, assegurando que cada aluno tenha acesso ao suporte necessário em suas áreas de estudo.
* **Melhorar a Resolução de Questões:** Proporcionar um espaço para que monitores possam resolver questões, registrando suas respostas e feedbacks.
* **Aumentar a Consulta de Dados:** Oferecer uma interface para consulta rápida e eficiente das informações dos alunos e seu desempenho.

## **2. Termo de Abertura do Projeto**

### **2.1 Contexto do Projeto**

O projeto foi idealizado a partir da necessidade de instituições educacionais em melhorar o acompanhamento acadêmico dos alunos. Com o aumento da demanda por soluções que integrem tecnologia e educação, a ideia é desenvolver um sistema que permita um gerenciamento eficaz das atividades de monitoria, melhorando o desempenho estudantil e a interação entre alunos e monitores.

### **2.2 Público-Alvo**

O público-alvo do sistema inclui:

* **Alunos:** Que buscam apoio nas suas atividades de estudo.
* **Monitores:** Que desejam gerenciar e auxiliar alunos em suas dúvidas.
* **Administradores:** Que precisam de uma visão geral do desempenho dos alunos e da eficiência dos monitores.

## **3. Funcionalidades do Sistema**

### **3.1 Cadastro de Questões**

Os alunos podem registrar questões que precisam de resolução. Para cada questão, o sistema coleta informações como:

* **Enunciado da Questão:** O texto que descreve a dúvida do aluno.
* **Resposta Esperada:** A solução correta ou feedback do monitor.
* **Status da Questão:** Um indicador se a questão foi resolvida ou não.

### **3.2 Resolução de Questões**

Monitores têm a capacidade de visualizar e resolver as questões registradas. Eles podem:

* **Acessar Questões:** Consultar todas as questões pendentes de resolução.
* **Registrar Respostas:** Oferecer soluções e feedbacks, contribuindo para o aprendizado dos alunos.
* **Marcar Questões como Resolvidas:** Indicar que a questão foi abordada e que o aluno pode revisitar a resposta.

### **3.3 Cadastro e Gerenciamento de Monitores**

Os monitores podem ser cadastrados no sistema, com as seguintes informações:

* **Nome do Monitor:** Identificação do profissional.
* **Disciplina:** A área de conhecimento que o monitor irá cobrir.
* **Contatos:** Informações adicionais que facilitam a comunicação entre alunos e monitores.

### **3.4 Consulta de Alunos**

O sistema permite consultar informações sobre os alunos, que incluem:

* **Nome:** Identificação do aluno.
* **Matrícula:** Número de matrícula do aluno para fins de registro e controle.
* **Desempenho:** Histórico de questões resolvidas e pendentes.

### **3.5 Geração de Cronograma de Estudos**

A partir das questões registradas e do desempenho dos alunos, o sistema pode gerar um cronograma personalizado que inclui:

* **Prioridade das Questões:** Questões mais desafiadoras são priorizadas.
* **Distribuição do Tempo:** Sugestões sobre quanto tempo dedicar a cada questão ou tema.
* **Revisão de Conteúdo:** Recomendações de revisão baseadas nas questões pendentes.

## **4. Estrutura do Código**

A estrutura do código do sistema é dividida em duas partes, que juntas fornecem todas as funcionalidades necessárias. As partes são organizadas para garantir modularidade e clareza, facilitando futuras expansões e manutenções.

### **4.1 Parte 1: Registro e Cadastro**

#### **Descrição do Código**

A primeira parte do código é responsável pelo cadastro de alunos, registro de questões e cadastro de monitores. O código é organizado em funções específicas, promovendo a reutilização e a clareza na implementação.

void cadastrarAluno(Aluno \*aluno){

int contstring;// variável para contar a quantidade de caracteres

do{

printf("Nome do Aluno: ");

fgets(aluno -> nome, 50, stdin); // Linha a linha de entrada para o nome do aluno

contstring = strlen(aluno->nome); // Obtem o comprimento da string

if(contstring <=1 ){

printf("Nome inválido! Quantidade mínima 1 caractere.\n");

}

}while(contstring <= 1); // Garante que o nome tenha pelo menos um caractere

aluno -> nome[strcspn(aluno -> nome, "\n")] = '\0'; // Remove o caractere de nova linha

Sleep(600); // Pausa a execução por 600 milissegundos

do{

printf("Digite sua matrícula: ");

fgets(aluno->matriculaAluno,20,stdin);

contstring = strlen(aluno->matriculaAluno);// Obtem o comprimento da string

if(contstring != 11){

printf("Matrícula inválida! Digite 10 números.\n");

}

}while(contstring != 11);//Garante que a matrícula tenha pelo menos 10 caracteres

aluno -> matriculaAluno[strcspn(aluno -> matriculaAluno, "\n")] = '\0'; // Remove a nova linha do final da string

Sleep(500); // Pausa a execução por 500 milissegundos

// Captura o curso do aluno

do{

printf("Curso: ");

fgets(aluno -> curso, 50, stdin);// Lê a entrada do usuário

contstring = strlen(aluno->curso);

if(contstring <= 1){

printf("Curso inválido! Quantidade mínima 1 caractere.\n");

}

}while(contstring <= 1); //garantindo que não seja vazio

aluno -> curso[strcspn(aluno -> curso, "\n")] = '\0';// Remove a nova linha do final da string

Sleep(500);// Pausa a execução por 500 milissegundos

// Captura o período em que o aluno está, garantindo que esteja entre 1 e 16

do{

printf("Período: ");

scanf("%d", &aluno -> periodo);

getchar(); // Limpa o buffer de entrada

if (aluno->periodo < 1 || aluno->periodo > 12 ){

printf("Período inválido! Digite um período entre 1 e 12.\n");

}

}while(aluno->periodo < 1 || aluno->periodo > 12); // Garante que o período esteja entre 1 e 12

// Captura a quantidade de disciplinas, garantindo um valor entre 1 e 10

do{

printf("Quantidade de disciplinas MAX 10: ");

scanf("%i",&aluno->quantdisciplinas);// Lê a entrada do usuário

getchar();

} while (aluno->quantdisciplinas > 10 || aluno->quantdisciplinas < 1); // Garante que a quantidade esteja entre 1 e 10

for(int i = 0; i < aluno->quantdisciplinas; i++){

Sleep(500); // Pausa a execução para melhorar a experiência do usuário

do{

printf("Disciplina %d: ", i + 1);

fgets(aluno -> disciplinas[i], 80, stdin); // Lê a entrada do nome da disciplina

contstring = strlen(aluno->disciplinas[i]);

if(contstring <= 1){

printf("Disciplina inválida! Quantidade mínima 1 caractere.\n");

}

}while(contstring <= 1);

aluno -> disciplinas[i] [strcspn(aluno -> disciplinas[i], "\n")] = '\0'; // Remove a nova linha do final da string

}

Sleep(500); // Pausa a execução por 500 milissegundos

printf("Cadastrando...\n"); // Mensagem de cadastro e gravação dos dados em arquivos

Sleep(2000); // Pausa para simular o tempo de cadastro

printf("Aluno cadastrado.\n");

Sleep(1000); // Pausa a execução por 1 segundo

system("pause");

// Cria e abre um arquivo com o nome do aluno (matrícula) para escrita

FILE \*alunotxt = fopen(aluno->matriculaAluno ,"w");

// Escreve os dados do aluno no arquivo

fprintf(alunotxt, "Nome: %s\n", aluno->nome);//função fprintf serve para escrever o nome do aluno no arquivo

fprintf(alunotxt, "Matrícula: %s\n", aluno->matriculaAluno);

fprintf(alunotxt, "Curso: %s\n", aluno->curso);//função fprintf serve para escrever o nome do curso do aluno no arquivo

fprintf(alunotxt, "Período: %d\n", aluno->periodo);//função fprintf serve para escrever o período que o aluno esta

fprintf(alunotxt, "Quantidade de Disciplinas: %d\n", aluno->quantdisciplinas);//função fprintf serve para escrever a quantidade de disciplinas do aluno

for (int i = 0; i < aluno->quantdisciplinas; i++) {//este for serve para listar os nomes das disciplinas

fprintf(alunotxt, "Disciplina %d: %s\n", i + 1, aluno->disciplinas[i]);

}

fclose(alunotxt); // Escreve os dados do aluno no arquivo e fecha

// Escreve os dados do aluno no arquivo

FILE \*nomealunotxt = fopen("cadastroaluno.txt","a"); //criar um novo arquivo caso não exista, se existir ele imprime no final as novas informações

fprintf(nomealunotxt, "Nome: %s\n", aluno->nome);//função fprintf serve para escrever o nome do aluno no arquivo

fprintf(nomealunotxt, "Matrícula: %s\n", aluno->matriculaAluno);

fprintf(nomealunotxt, "Curso: %s\n", aluno->curso);//função fprintf serve para escrever o nome do curso do aluno no arquivo

fprintf(nomealunotxt, "Período: %d\n", aluno->periodo);//função fprintf serve para escrever o período que o aluno esta

fprintf(nomealunotxt, "Quantidade de Disciplinas: %d\n", aluno->quantdisciplinas);//função fprintf serve para escrever a quantidade de disciplinas do aluno

for (int i = 0; i < aluno->quantdisciplinas; i++) {//este for serve para listar o nome das disciplinas

fprintf(nomealunotxt, "Disciplina %d: %s\n", i + 1, aluno->disciplinas[i]);

}

fclose(nomealunotxt); // Fecha o arquivo e salva

}

void cadastrarQuestao(Questao \* questao){

system("cls");

Questao questoes;

int opcao;//opção no switch case

char buffer[255];// variável que ira receber a leitura do banco de dados e imprimir

int contstring; // Variável para contar o comprimento das strings

// Captura a resposta correta

int respostaCorreta = 0;// inicializa em 0

Sleep(500); // Pausa para melhorar a experiência do usuário

// Captura a matéria da questão

do{

printf("Matéria: ");

fgets(questao -> materia, 80, stdin);// Lê a entrada do usuário

contstring = strlen(questao->materia);// Armazena o comprimento da matéria

if(contstring <= 1){

printf("Matéria inválida! Quantidade mínima 1 caractere.\n");

}

}while(contstring <= 1); //Continua pedindo se a entrada estiver vazia

questao -> materia[strcspn(questao -> materia, "\n")] = '\0';// Remove a nova linha do final da string

Sleep(600); // Pausa para melhorar a experiência do usuário

// Captura o enunciado da questão

do{

printf("Enunciado da questão: ");

fgets(questao -> enunciado, 250, stdin); // Lê a entrada do usuário

contstring = strlen(questao->enunciado); // Verifica o comprimento do enunciado

if (contstring <= 1){

printf("Enunciado inválido! Quantidade mínima 1 caractere.\n");

}

}while(contstring <= 1); // Continua pedindo se a entrada estiver vazia

questao -> enunciado[strcspn(questao -> enunciado, "\n")] = '\0'; // Remove a nova linha do final da string

Sleep(500); // Pausa para melhorar a experiência do usuário

// Captura as alternativas da questão

for(int i=0; i < 5;i++){

do{

printf("Alternativa %i: ",i+1);

fgets(questao -> alternativas[i], 100, stdin); // Lê a alternativa

contstring = strlen(questao->alternativas[i]); // Verifica o comprimento da alternativa

if (contstring <=1){

printf("Alternativa invalida! Quantidade mínima 1 caractere.\n");

}

}while(contstring <= 1) ; // Continua pedindo se a entrada estiver vazia

}

do{

printf("Número da questão correta(1 a 5): ");

fgets(questao -> resposta, 100, stdin);// Lê a resposta correta

respostaCorreta = atoi(questao->resposta);// converte a string para int

}while(respostaCorreta < 1 || respostaCorreta > 5); // Continua pedindo até ser digitado o número valido

questao -> resposta[strcspn(questao -> resposta, "\n")] = '\0'; // Remove a nova linha da resposta

Sleep(500); // Pausa para melhorar a experiência do usuário

printf("Cadastrando...\n");

Sleep(1000);// Pausa para simular o tempo de cadastro

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

Sleep(1000);// Pausa para simular o tempo de cadastro

printf("Deseja cadastrar mais questões? 1 = SIM / 2 = NÃO: ");

scanf("%d",&opcao);

getchar();

switch (opcao){

case 1:

cadastrarQuestao(&questoes);

break;

case 2:

break;

default:

printf("Opção inválida.\n");

printf("Retornando ao menu principal.\n");

Sleep(600);

break;

}

// Abre o arquivo para armazenar a questão cadastrada

FILE \* questaotxt = fopen(questao->materia,"a"); // Grava a matéria no arquivo

fprintf(questaotxt, "Matéria: %s\n",questao->materia);

fprintf(questaotxt,"Enunciado da questão: %s\n",questao->enunciado);

for(int i=0; i <5; i++){

fprintf(questaotxt,"Alternativa %i: ",i+1);

fprintf(questaotxt,"%s",questao->alternativas[i]);

}

fprintf(questaotxt,"%s\n",questao->resposta);//Vai imprimir no banco de dados apenas o número da resposata, assim facilitando na hora da comparação

fclose(questaotxt); // Fecha o arquivo apois a escrita

FILE \*listatxt = fopen("listamaterias.txt", "r"); // Abre o arquivo para leitura

if (listatxt == NULL) { // Se o arquivo não existir ( NULL) ele ira entrar na condição e criar

listatxt = fopen("listamaterias.txt", "a"); //Abre o arquivo para escrita caso não exista ele cria

}

fclose(listatxt); // fecha e salva o arquivo

int materia\_ja\_cadastrada = 0; // incia o contador de matéria cadastrada

FILE \*listaMateriasTxt = fopen("listamaterias.txt", "r+"); //abre o arquivo para leitura e escrita

// percorre todas as linhas para verificar se a matéria já existe

while (fgets(buffer, sizeof(buffer), listaMateriasTxt)) {

buffer[strcspn(buffer, "\n")] = 0; // Remove o '\n' para fazer a comparação corretamente

if (strcmp(buffer, questao->materia) == 0) { // compara o buffer que foi lido pelo while e a materia do cadastro se as strings são iguais

materia\_ja\_cadastrada = 1; // caso a função strcmp retorne verdadeiro(== 0 ) Significa que a materia ja esta no arquivo listamaterias.txt

break; // Se a matéria já foi encontrada, não precisa continuar o loop

}

}

// Se a matéria não foi encontrada, adiciona ao arquivo

if (materia\_ja\_cadastrada == 0) { // se o contador de materias for = 0

strupr(questao->materia); // imprime no banco de dados o nome da materia maiúscula

fprintf(listaMateriasTxt, "%s\n", questao->materia); // imprime a materia no arquivo

}

fclose(listaMateriasTxt); // Fecha o arquivo após a operação

}

void cadastrarMonitor(Monitor \*monitor){

system("cls");

int contstring;

Sleep(500);// Pausa a execução por 500 milissegundos antes de iniciar o cadastro

do{

printf("Nome do Monitor: ");

fgets(monitor->nome, 50, stdin); // Captura o nome do monitor

contstring = strlen(monitor->nome);

if(contstring <= 1){

printf("Monitor inválido! Quantidade mínima 1 caractere.\n");

}

}while(contstring <=1); //garantindo que o nome não seja vazio

monitor->nome[strcspn(monitor->nome, "\n")] = '\0'; // Remove o caractere de nova linha

(monitor->quant = 1); // Inicializa a quantidade de monitores

Sleep(500); // Pausa a execução por 500 milissegundos

// Captura o horário disponível para monitoria, garantindo que seja maior que 5 caracteres

do{

printf("Horário disponível para monitoria: ");

fgets(monitor->horario, 80, stdin); // Lê a entrada do horário

contstring = strlen(monitor->horario);

if (contstring <= 5){

printf("Formato de hora inválido!\n");

}

}while(contstring <= 5);// Garante que o horário tenha pelo menos 5 caracteres

monitor->horario[strcspn(monitor->horario, "\n")] = '\0'; // Remove o caractere de nova linha

Sleep(500); // Pausa a execução por 500 milissegundos

do{

printf("Telefone para contato: 9. ");

fgets(monitor->telefone, 80, stdin); // Lê a entrada do telefone

contstring = strlen(monitor->telefone);

if (contstring != 9 && contstring != 10){

printf("Telefone invalido!\n");

}

}while(contstring != 9 && contstring != 10); // Garante que o telefone tenha pelo menos 8 ou 9 caracteres

monitor->telefone[strcspn(monitor->telefone, "\n")] = '\0'; // Remove o caractere de nova linha

Sleep(500);// Pausa a execução por 500 milissegundos

printf("Cadastrando....\n");

Sleep(600); // Pausa a execução para simular o tempo de cadastro

printf("Monitor cadastrado com sucesso.\n");

Sleep(1000);// Pausa a execução para simular o tempo de cadastro

FILE \*monitortxt = fopen("cadastromonitor.txt","a"); // Abre o arquivo para armazenar o monitor cadastrado

// Escreve os dados do monitor no arquivo

fprintf(monitortxt,"\nNome do Monitor: %s\n",monitor->nome); // Escreve o nome do monitor

fprintf(monitortxt,"Horário disponivel para monitoria: %s\n",monitor->horario); // Escreve o horário disponível

fprintf(monitortxt,"Telefone para contato:9.%s\n",monitor->telefone); // Escreve o telefone, prefixando com '9.'

fclose(monitortxt);// Fecha o arquivo após a escrita

}

### **4.2 Parte 2: Resolução e Consulta**

#### **Descrição do Código**

A segunda parte do código foca na resolução de questões e na consulta de informações dos alunos, monitores e questões. Essa estrutura permite que o sistema seja fácil de usar, possibilitando que monitores e alunos naveguem pelas funcionalidades com simplicidade.

void resolverQuestoes(Questao \*questoes, int numQuestoes, int \*acertos) {

int resposta; // Variável para armazenar a resposta do usuário

char rquestao[80];// Variável para armazenar a matéria selecionada pelo usuário

char buffer[7500]; // Buffer para leitura de linhas do arquivo

char bufferResposta[255];

int RespostaSn;

\*acertos = 0; // Inicializa o contador de acertos

FILE \*lista;// Ponteiro para o arquivo da lista de matérias

printf("Lista de todas as Disciplinas disponíveis para estudo:\n");

lista = fopen("listamaterias.txt","r"); // Acessa e ler o arquivo com a lista de todas as disciplinas

while (fgets(buffer, sizeof(buffer), lista)) {

printf("%s", buffer);

}

fclose(lista); // Fecha o arquivo apois a leitura

printf("Qual matéria você deseja resolver as questões?\n");

fgets(rquestao, 80, stdin);

strupr(rquestao);// deixa o texto maiúsclo

rquestao[strcspn(rquestao, "\n")] = '\0'; // Remove nova linha

FILE \*file = fopen(rquestao, "r"); // Abre o arquivo da matéria selecionada

if (file == NULL) {

Sleep(500);

printf("Erro ao abrir o arquivo de questões.\n");

Sleep(800);

return;// Retorna em caso de erro

}

// Inicializa o contador de questões

numQuestoes = 0;

// Lê as questões do arquivo

while (fgets(questoes[numQuestoes].materia, sizeof(questoes[numQuestoes].materia), file) != NULL) {

questoes[numQuestoes].materia[strcspn(questoes[numQuestoes].materia, "\n")] = '\0'; // Remove nova linha

fgets(questoes[numQuestoes].enunciado, sizeof(questoes[numQuestoes].enunciado), file);

questoes[numQuestoes].enunciado[strcspn(questoes[numQuestoes].enunciado, "\n")] = '\0'; // Remove nova linha

for (int i = 0; i < 5; i++) {

fgets(questoes[numQuestoes].alternativas[i], sizeof(questoes[numQuestoes].alternativas[i]), file);

questoes[numQuestoes].alternativas[i][strcspn(questoes[numQuestoes].alternativas[i], "\n")] = '\0'; // Remove nova linha

}

fgets(questoes[numQuestoes].resposta, sizeof(questoes[numQuestoes].resposta), file);

questoes[numQuestoes].resposta[strcspn(questoes[numQuestoes].resposta, "\n")] = '\0'; // Remove nova linha

(numQuestoes)++; // Incrementa o número de questões lidas

}

fclose(file); // Fecha o arquivo de questões

do{

printf("Quantas questões deseja responder de %s: ",rquestao);//solicita ao usuário a quantidade de questões que deseja responder

scanf("%i",&numQuestoes);

getchar();

if(numQuestoes <= 0 || numQuestoes > 30){

printf("Informe um valor de 1 até 30\n");

}

}while(numQuestoes <= 0 || numQuestoes > 30 );

questoes->totQuest = numQuestoes;// recebe o número de questão que o usuário deseja responder, para mostrar no fim do questionario

// exiba as questões

system("cls");

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

Sleep(500); // Pausa para melhorar a experiência do usuário

// Exibe a matéria e o enunciado da questão

printf("%s\n", questoes[i].materia);

Sleep(500);

printf("Questão %d:\n%s\n", i + 1, questoes[i].enunciado);

Sleep(500);

// Exibe as alternativas

for (int j = 0; j < 5; j++) {

printf("%s\n", questoes[i].alternativas[j]);

}

// Solicita a resposta do usuário

int respostaInt = atoi(questoes[i].resposta); // Transforma a reposta cadastrada em int para melhor fazer a comparação e tratamento de erros

do {

printf("Número da questão correta(1 a 5): ");

scanf("%i",&resposta); // Lê a resposta

printf("\n");

if(resposta < 1 || resposta > 5 ){

printf("Valor inválido!\n");

}

} while (resposta < 1 || resposta > 5 ); // Continua pedindo até ser um número válido entre 1 e 5

if (respostaInt == resposta) { // Verifica se a resposta está correta

(\*acertos)++;

printf("Questão %d correta.\n\n",i+1);//exibe se a respota foi correta

//Estrutura para salvar em um arquivo txt todos as repostas corretas

FILE \*respostaCorretatxt = fopen("RespostaCorreta.txt","a");

fprintf(respostaCorretatxt,"%s\n",questoes[i].materia);

fprintf(respostaCorretatxt,"Questão %d:\n%s\n", i + 1, questoes[i].enunciado);

for(int j = 0; j < 5; j++){

fprintf(respostaCorretatxt,"%s\n",questoes[i].alternativas[j]);

}

fprintf(respostaCorretatxt,"Alternativa correta: %i\n\n",resposta);

fclose(respostaCorretatxt);

}else{

printf("Questão %d errada.\n\n",i+1);//exibe se a repsota foi errada

}

}

//caso o aluno tenha certado mais que uma questão. Vai ser feita validação se ele deseja visualizar as repostas correta

//system("cls");

if((\*acertos) >= 1){

printf("Deseja visualizar as questões que você acertou? 1 = SIM / (QUALQUER TECLA PARA CONTINUAR SEM VISUALIZAR) \n");

scanf("%d",&RespostaSn);

getchar();

if(RespostaSn == 1){

FILE \*respostaCorreta = fopen("RespostaCorreta.txt","r");

system("cls");

printf("Lista de reposta correta:\n");

mostrarlinha();

while( fgets(bufferResposta, sizeof(bufferResposta),respostaCorreta)){ // Ler e imprime o conteúdo do arquivo

Sleep(500);

printf("%s", bufferResposta);

}

mostrarlinha();

fclose(respostaCorreta);

}

Sleep(1000);

}

remove("RespostaCorreta.txt");

}

void consultarAlunos() {

FILE \*file; // Ponteiro para o arquivo que armazena dados de um aluno específico

FILE \*lista; // Ponteiro para o arquivo que armazena a lista de todos os alunos

Aluno cadAluno;

char matricula[80]; // Variável para armazenar a matrícula do aluno para consulta

char buffer[255]; // Buffer para ler linhas dos arquivos

int opcao, opcao1; // Variável para armazenar a opção selecionada pelo usuário // Variável para armazenar a opção selecionada pelo usuário

int parar = 1; // Variável de controle para o loop de consulta

printf("Consultar Alunos: \n");

Sleep(500);

while (parar == 1) { // Loop para continuar a consulta até que o usuário decida sair

printf("\n1 -> Acessar toda a lista de alunos cadastrados:\n");

printf("2 -> Consultar um aluno por matricula:\n");

printf("0 -> Voltar ao menu anterior\n");

printf("Sua opção: ");

scanf("%i",&opcao); // Ler a opção escolhida pelo usuário

getchar(); // Limpa o buffer de entrada

system("cls");

Sleep(500);

switch (opcao){

case 1 : // Acessa e ler o arquivo com a lista de todos os alunos

lista = fopen("cadastroaluno.txt","r");

if (lista == NULL) {

//caso não exista o arquivo cadastroaluno, fara um validação se o aluno deseja cadasrtar um aluno

printf("Lista de alunos não encontrada\n");

printf("Deseja cadastrar uma aluno ? 1 = SIM / 2 = NÃO: ");

scanf("%d",&opcao1);

getchar();

switch (opcao1){

case 1:

system("cls");

cadastrarAluno(&cadAluno);

return ;

break;

case 2:

printf("Retornando ao menu anterior.\n");

Sleep(1200);

return ;

break;

default:

printf("Opção inválida.\n");

printf("Retornando ao menu anterior.\n");

Sleep(1200);

return ;

break;

}

} else {

while (fgets(buffer, sizeof(buffer), lista)) { // Ler e imprime o conteúdo do arquivo

printf("%s", buffer);

}

fclose(lista); // Fecha o arquivo após a leitura

}

system("pause");

system("cls");

break;

case 2 :

// Consulta um aluno específico pela matrícula

printf("Digite a matrícula do aluno:");

fgets(matricula, 80, stdin);

Sleep(500);

matricula[strcspn(matricula, "\n")] = '\0';

if (strcmp(matricula, "0") == 0){

break;

}

file = fopen(matricula, "r"); // Acessa e ler o arquivo correspondente e matrícula do aluno

if (file == NULL) {

//se o aluno digitar uma matricula invalida , será feita uma validação para cadastar uma aluno caso desejado

printf("Aluno não encontrado.\n");

printf("Deseja cadastrar uma aluno ? 1 = SIM / 2 = NÃO: ");

scanf("%d",&opcao1);

getchar();

switch (opcao1){

case 1:

system("cls");

cadastrarAluno(&cadAluno);

return ;

break;

case 2:

printf("Retornando ao menu anterior.\n");

Sleep(1200);

return ;

break;

default:

printf("Opção inválida.\n");

printf("Retornando ao menu anterior.\n");

Sleep(1200);

return ;

break;

}

} else {

while (fgets(buffer, sizeof(buffer), file)) {

printf("%s", buffer);// Ler e imprime o conteúdo do arquivo

}

fclose(file);//fecha o arquivo após leitura

}

system("pause");

system("cls");

break;

case 0: // Sai do loop e volta ao menu anterior

parar ++;

break;

default:

printf("Operação inválida.");

break;

}

}

}

void consultarQuestoes() {

system("cls");

int opcao;

Questao questoes;

FILE \*lista; // Ponteiro para o arquivo que armazena a lista de disciplinas

FILE \*file; // Ponteiro para o arquivo que armazena as questões de uma disciplina

char materia[80]; // Variável para armazenar o nome da disciplina para consulta

char buffer[550]; // Buffer para ler linhas dos arquivos

printf("Consultar Questões:\n");

Sleep(500);

while (1) { // Loop para continuar a consulta até que o usuário decida sair

lista = fopen("listamaterias.txt","r"); // Acessa e ler o arquivo com a lista de todas as disciplinas

if (lista == NULL) { // Se o arquivo não existir ( NULL) ele ira entrar na condição e criar

printf("Não tem disciplinas cadastradas\n");

printf("Deseja cadastrar uma disciplina ? 1 = SIM / 2 = NÃO: ");

scanf("%d",&opcao);

getchar();

switch (opcao){

case 1:

cadastrarQuestao(&questoes);

return ;

break;

case 2:

printf("Retornando ao menu anterior.\n");

Sleep(1200);

return ;

break;

default:

printf("Opção inválida.\n");

printf("Retornando ao menu anterior.\n");

Sleep(1200);

return ;

break;

}

}else{

printf("Lista de todas as Disciplinas disponíveis para estudo:\n");

while (fgets(buffer, sizeof(buffer), lista)) {

printf("%s", buffer);

}

fclose(lista); // Fecha o arquivo após a leitura

}

printf("Digite a disciplina para consultar (ou '0' para voltar ao menu anterior): \n"); // Solicita ao usúrio o nome da disciplina para consultar questões

printf(">>> ");

fgets(materia, 80, stdin);

strupr(materia);

system("cls");

materia[strcspn(materia, "\n")] = '\0';

if (strcmp(materia, "0") == 0){ // Se o usuário digitar "0", sai do loop

break;

}

file = fopen(materia, "r"); // Acessa e ler o arquivo correspondente á disciplina

if (file == NULL) {

printf("Questão não encontrada.\n");

printf("Deseja cadastrar uma disciplina ? 1 = SIM / 2 = NÃO: ");

scanf("%d",&opcao);

getchar();

switch (opcao){

case 1:

cadastrarQuestao(&questoes);

return ;

break;

case 2:

printf("Retornando ao menu anterior.\n");

Sleep(1200);

return ;

break;

default:

printf("Opção inválida.\n");

printf("Retornando ao menu anterior.\n");

Sleep(1200);

return ;

break;

}

Sleep(800);

} else {

while (fgets(buffer, sizeof(buffer), file)) { // Ler e imprime o conteúdo do arquivo

printf("%s", buffer);

Sleep(300);

}

system("pause");

system("cls");

fclose(file);

Sleep(800);

}

}

}

// Função para consultar monitores

void consultarMonitores() {

FILE \*file; // Ponteiro para o arquivo que armazena dados dos monitores

char buffer[255]; // Buffer para ler linhas do arquivo

int opcao;

Monitor monitor;

system("cls");

printf("Consultar Monitores:\n");

Sleep(500);

while (1) { // Loop para continuar a consulta até que o usuário decida sair

file = fopen("cadastromonitor.txt", "r"); // Acessa e ler o arquivo com os dados dos monitores

if (file == NULL) {

printf("Analisando...\n");

Sleep(800);

printf("Monitor não encontrado.\n");

Sleep(1000);

printf("Deseja cadastrar um monitor ? 1 = SIM / 2 = NÃO: ");

scanf("%d",&opcao);

getchar();//limpa o buffer de entrada

switch (opcao){

case 1:

cadastrarMonitor(&monitor);//chama a função para cadastrar monitor

return;

break;

case 2:

printf("Retornando ao menu anterior.\n");

Sleep(1200);

return ;

break;

default:

printf("Opção inválida.\n");

printf("Retornando ao menu anterior.\n");

Sleep(1200);

return ;

break;

}

} else {

while (fgets(buffer, sizeof(buffer), file)) { // Ler e imprime o conteúdo do arquivo

printf("%s", buffer);

}

system("pause");

system("cls");

fclose(file);//fecha o arquivo após leitura

Sleep(500);

break;

}

}

}

## **5. Estrutura Principal**

#### **Descrição do Código**

Foi criado um menu usando e utilizando o (switch-case) para a chamada de cada uma das funções, que inicializam e dão vida ao sistema.

int main(){

*FILE* \*materias;

    setlocale(LC\_ALL, "Portuguese\_Brazil"); // Configura a localização para português do Brasil

*Aluno* aluno; // Declara uma variável do tipo Aluno

*Cronograma* cronograma; //Declara uma variável do tipo Cronograma

*Questao* questoes[50]; // Declara um array de Questão com capacidade para 50 questões

    questoes->quantQuest = 0; // Inicializa a quantidade de questões cadastradas para 0

*Monitor* monitor;  // Declara uma variável do tipo Monitor

    monitor.quant = 0; // Inicializa a quantidade de monitores cadastrados para 0

    int numQuestoes = 0;  // Variável para armazenar o número de questões cadastradas

    int opcao; // Variável para armazenar a opção escolhida pelo usuário

    int acertos; // Variável para armazenar o número de acertos nas questões

    do{

        system("cls"); // Limpa a tela do console

*HANDLE* hThread = CreateThread(NULL, 0, temporizador, &cronograma, 0, NULL);

        if (hThread == NULL) { // Verifica se houve erro ao criar a thread

            printf("Erro ao criar thread do temporizador.\n");

            return 1;// Retorna 1 indicando erro

            }

        // Cria a thread do temporizador

        system("cls");

        mostrarlinha();

        printf("\t    Menu\n\n");

        printf("1 -> Cadastrar Aluno\n");

        printf("2 -> Gerar Cronograma de Estudo\n");

        printf("3 -> Cadastrar Questões\n");

        printf("4 -> Resolver Questões\n");

        printf("5 -> Cadastrar Monitor\n");

        printf("6 -> Consultas\n");

        printf("0 -> Sair");

        mostrarlinha();

        Sleep(400);

        printf("Escolha uma opção: ");

        scanf("%d", &opcao);

        getchar();

        switch(opcao){

            case 1:

            system("cls");

            cadastrarAluno(&*aluno*);

            break;

            case 2:

            system("cls");

            gerarCronograma(&*cronograma*);

            system("pause");

            break;

            case 3:

            system("cls");

            if (numQuestoes < 20){ // Verifica se o número de questões é menor que 20

                cadastrarQuestao(&questoes[numQuestoes]); // Chama a função para cadastrar uma nova questões

                numQuestoes++;  // Incrementa o número de questões cadastradas

                questoes->quantQuest++; // Incrementa a quantidade de questões no array de questões

            }

            else{

                printf("Limite de questões atingida.\n");  // Mensagem de erro quando o limite de questões até atingido

            }

            break;

            case 4:

            system("cls");

                materias = fopen("listamaterias.txt","r");

                if (materias == NULL){ // Verifica se nenhuma questão foi cadastrada

                    printf("Analisando...\n");

                    Sleep(1000);

                    printf("\*\*Nenhuma questão foi cadastrada ainda!\*\*\n");

                    Sleep(1500);

                    break;

                }else{

                resolverQuestoes(questoes, numQuestoes, &acertos); // Chama a função para resolver questões

                Sleep(600);

                printf("Voce acertou %d de %d questões. \n", acertos, questoes->totQuest); // Exibe o número de acertos

                system("pause");

                Sleep(600);

                }

                break;

            case 5:

                system("cls");

                cadastrarMonitor(&*monitor*); // Chama a função para cadastrar um monitor

                break;

            case 6:

                menuConsulta(); // Chama a função para acessar o menu de consulta

                break;

            case 0:

                printf("Saindo do Programa...\n");

                Sleep(1000);

                printf("Programa FINALIZADO.");

                Sleep(1000);

                break;

            default:

                printf("Operação inválida, tente novamente.\n");

        }

            // Fecha o handle da thread

        CloseHandle(hThread);

    }while (opcao != 0); // Continua no menu até que o usuário escolha a opção 0 para sair

return 0; // Retorna 0 indicando que o programa terminou com sucesso

}

## **6. Implementação**

### **6.1 Ambiente de Desenvolvimento**

O sistema foi desenvolvido utilizando o compilador GCC no ambiente de desenvolvimento Visual Studio Code. O ambiente é configurado para suportar a linguagem C, com as seguintes ferramentas e extensões recomendadas:

* **Compilador GCC:** Para compilar e executar o código C.
* **Visual Studio Code:** IDE que fornece suporte a debugging e controle de versões.
* **GDB:** Para depuração do código.

### **6.2 Estrutura de Pastas**

A estrutura de pastas do projeto é organizada da seguinte maneira:

/gestao\_monitoria

|-- src

| |-- main.c // Código principal do sistema

| |-- questoes.c // Funções para gerenciamento de questões

| |-- monitores.c // Funções para gerenciamento de monitores

| |-- alunos.c // Funções para gerenciamento de alunos

|-- include

| |-- questoes.h // Cabeçalho com definições para questões

| |-- monitores.h // Cabeçalho com definições para monitores

| |-- alunos.h // Cabeçalho com definições para alunos

|-- Makefile // Arquivo para automação da construção do projeto

## **7. Instruções para o Usuário**

Este guia prático fornece instruções passo a passo sobre como utilizar as principais funcionalidades do sistema.

### **7.1 Registrar Monitor**

1. **Acesse a opção de registro de monitores no menu principal.**
2. **Digite o nome do monitor.**
3. **Digite o e-mail do monitor.**
4. **Pressione a tecla 'Enter' para confirmar o registro.**
5. **Uma mensagem de confirmação será exibida indicando que o monitor foi registrado com sucesso.**

### **7.2 Registrar Aluno**

1. **Selecione a opção de registro de alunos no menu principal.**
2. **Insira o nome do aluno.**
3. **Forneça o número de matrícula do aluno.**
4. **Pressione 'Enter' para finalizar o registro.**
5. **Verifique a mensagem de sucesso que aparece.**

### **7.3 Consultar Monitores**

1. **Escolha a opção de consulta de monitores.**
2. **Uma lista com os monitores registrados será exibida.**
3. **Você pode rolar a lista para visualizar todos os monitores.**

### **7.4 Consultar Alunos**

1. **Acesse a opção de consulta de alunos.**
2. **Uma lista com os alunos registrados aparecerá.**
3. **Utilize as setas para navegar pela lista.**

### **7.5 Registrar Questão**

1. **Selecione a opção de registro de questões.**
2. **Digite o enunciado da questão.**
3. **Forneça as alternativas (se houver).**
4. **Pressione 'Enter' para registrar a questão.**
5. **A confirmação do registro será exibida.**

### **7.6 Resolver Questão**

1. **Acesse a seção de resolução de questões.**
2. **Digite o número da questão que deseja resolver.**
3. **Forneça sua resposta.**
4. **Pressione 'Enter' para registrar sua resposta.**
5. **Uma mensagem confirmando a resolução será mostrada.**

### **7.7 Consultar Questões**

1. **Escolha a opção de consulta de questões.**
2. **As questões registradas serão listadas.**
3. **Use as setas para navegar entre as questões.**

### **7.8 Gerar Cronograma de Estudos**

1. **Selecione a opção para gerar o cronograma de estudos.**
2. **Escolha os alunos e as questões a serem incluídas.**
3. **Pressione 'Enter' para criar o cronograma.**
4. **O cronograma será exibido na tela e também poderá ser salvo**

## **8. Referências**

No decorrer do desenvolvimento do sistema tivemos alguns entraves com (funções, banco de dados e o uso de um temporizador), que foram vencidos com a ajudar de algumas pesquisas, que seguem link a baixo:

**Funções:** <https://www.youtube.com/watch?v=jd9PABbIvbA>

**Banco de dados:** <https://www.youtube.com/watch?v=eriDnpkh5kA>

**Temporizador:**<https://www.youtube.com/watch?si=GOS5INq-qt2Svp2o&v=hVJQI0ThHY8&feature=youtu.be>

<https://cursos.alura.com.br/forum/topico-exemplo-de-uma-segunda-thread-em-linguagem-c-397450>

## **9. Conclusão**

A **Gestão de Estudos e Monitoria para Alunos** é um sistema abrangente que atende às necessidades de alunos, monitores e administradores, melhorando a comunicação e a eficácia do aprendizado. A modularidade do código e a simplicidade da interface garantem que o sistema possa ser facilmente atualizado e expandido conforme novas necessidades surgirem.

### **9.1 Sugestões para Futuras Expansões**

* **Integração com Plataformas de Ensino:** Permitir que o sistema se conecte a plataformas de e-learning, facilitando a importação/exportação de dados.
* **Módulo de Avaliações:** Implementar uma funcionalidade para a criação e gestão de avaliações online, integrando com as questões registradas.
* **Relatórios de Desempenho:** Gerar relatórios detalhados sobre o desempenho dos alunos, ajudando os administradores a identificar áreas que necessitam de intervenção.

## **10. Lista de Alunos**

**Curso: Análise e Desenvolvimento de Sistemas**

**Turma: B**

|  |  |
| --- | --- |
| Adryan Lucas Reis Amorim da Silva | Luan Victor Duarte Silva |
| Alessandra Lopes da silva | Lucas Daniel roque de lima |
| Arthur Costa de Souza | Lucas Gabriel dos Santos Mendes |
| Caio Fernandes Alves de Oliveira | Marcos Felipe dos Santos Santana |
| Carlos Antonio Bezerra da Silva Filho | Maria Júlia Vicente Gomes Trajano |
| Chayane pereira leal | Meyson Sales dos Santos |
| Edson Marques de Lima | Pedro Ferreira da Silva |
| Fabio rodrigues dos Santos | Pedro Vinicius Freitas da Silva |
| Felipe Paulo das Neves | Raphael Eloi da Silva Fay |
| Glauber Monte de Oliveira | Ricardo Breno Souza Cavalcanti |
| Glebson Carlos da silva | Robson Felix de Albuquerque |
| Guilherme Galindo Ribeiro | Ruan Eryck Lima |
| Guilherme Roberto de Melo Muniz | Thiago Silva Gomes |
| Héctor Augusto leone de Brito times Carvalho | Vinicius Soares Ferreira De Lima |
| Hugo de Mesquita Rufino | Vittor Chacon Pádua |
| Italo Aleffy Matos do Nascimento | Walber Alex dos Santos Botelho |
| João Vitor Santana Lins | Wallace Jose dos Santos |
| Joás Vitor silva bezerra | Weslei de Santana Morais Brito |

José Carlos de Assis Lins Wilton José D. Ribeiro Junior