SISTEMA DE GESTÃO ESCOLAR

PROJETO C++

Maria Marcolina Lima Cardoso

INSPIRAÇÃO

- A gestão escolar depende de bons aplicativos para o registro das aulas, dos alunos, e geração da grade horária. Inspirada nas experiências pessoais, como professora, decidi fazer um **programa de Gestão Escolar** para entender como programas do tipo funcionam e conseguir aprofundar o conhecimento em linguagens orientadas a objeto como o C++

OBJETIVOS

Menu Gestor

- *Cadastrar Turmas*
- *Cadastrar Alunos*
- *Gerar Grade Horária*
- *Sumários*
  - ✓ *Por aluno*
  - ✓ *Por turma*
  - ✓ *Geral*

Menu Professor

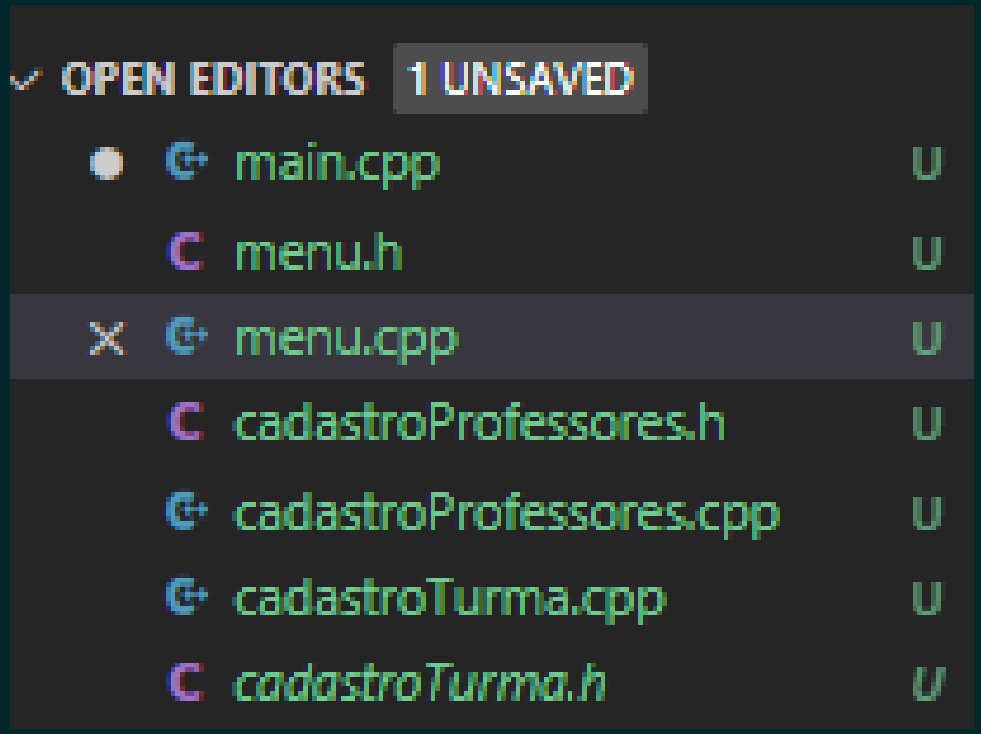
- *Registro de Aulas*
- *Registro de Frequências*
- *Registro de Notas*
- *Sumários*
  - ✓ *Por aluno*
  - ✓ *Por turma*
  - ✓ *Geral*

Os CADASTROS incluem: Adicionar, Procucar, Atualizar, Deletar, Listar
Os SUMÁRIOS incluem: Médias, resultados finais, número de faltas.

IMPLEMENTAÇÃO ATÉ O MOMENTO

Programa Multi-Arquivo:

Main, Menu.c, Menu.h | Arquivos de Cadastro.c e .h



IMPLEMENTAÇÃO ATÉ O MOMENTO

Função switch no menu

```
int menu(int &c, vector <string> &vec){

    system("cls");
    cout << "Bem-Vindo ao Gerenciamento Escolar!\n" <<  std::endl;//criarConta();
    cout << "1 - Adicionar professor \n2 - Achar cadastro de professor \n3 - Atualizar ca
    cin >> c;

    system("cls");
    switch(c){
        case 1: //add
            cadastrarProfessor(vec);
            break;
        case 2: //add
            findProf(vec);
            system("pause");
            break;
        case 3: //add
            replaceProf(vec);
            break;
        case 4: //remove
            deleteProf(vec);
            break;
        case 5: //print
            printProf(vec);
            break;
        case 6: // exit
            exit(0);
            break;
        default:
            cout << "Digite uma opcao valida!" << endl;
            break;
    }
}
```

IMPLEMENTAÇÃO ATÉ O MOMENTO

Prototipagem das Funções

main.cpp U | menu.h U | menu.cpp U | cadastroProfessores.h U | cadastroProfessores.cpp U | cadastroTurma.cpp U | cadastr

cadastroProfessores.h > ...

```
1   #ifndef CADASTROPROFESSORES_H
2   #define CADASTROPROFESSORES_H
3
4   //Cadastrar professor
5   std::vector<std::string> cadastrarProfessor(std::vector <std::string> &a);// Cadastrar um professor
6   //Imprimir professores cadastrados
7   void printProf(const std::vector <std::string> &vec); //imprime professores
8   //Achar professor
9   int findProf(std::vector <std::string> &vec);
10  //Atualizar cadastro
11  std::vector<std::string> replaceProf(std::vector <std::string> &vec);
12  // Deletar registro
13  std::vector<std::string> deleteProf(std::vector <std::string> &vec);
14
15  //Sort names
16  bool mycomp(std::string a, std::string b);  //compare strings to sort names
17  std::vector<std::string> alphabaticallySort(std::vector <std::string> a); //sortnames
18
19
20  #endif // CADASTROPROFESSORES_H
```

IMPLEMENTAÇÃO ATÉ O MOMENTO

Funções

CadastrarProfessor

PrintProf – Imprime a lista cadastrada

```
vector<string> cadastrarProfessor(vector <string> &vec){
    string tempName = "";
    cout << "Digite o nome do professor:" << endl;
    cin.ignore();
    getline(cin, tempName);
    vec.push_back(tempName);
    system("cls");
    cout << "Professor " << tempName << " Cadastrado com Sucesso!\n\n" << endl;
    system("Pause");

    return vec;
}
```

```
void printProf(const vector <string> &vec){
    //function to sort names alphabetically
    if (vec.empty()){
        cout << "Nao ha professores cadastrados.\n\n" << endl;
        system("pause");
    }else{
        vector <string> names;
        names = alphabaticallySort(vec);

        cout << "Professores cadastrados:" << endl;
        for(int i = 0; i < names.size(); i++){
            cout << i+1  << ": " << names[i] << endl;
        }
        puts("\n\n");
        system("pause");
    }
}
```

IMPLEMENTAÇÃO ATÉ O MOMENTO

Funções

replaceProfessor – Atualizar cadastro
deleteProf – deleta cadastro

```
vector<string> deleteProf(vector <string> &vec){

    int op;
    int index = findProf(vec);

    cout << "Deseja remover este Cadastro (Sim- 1; Nao:0)? " << endl;
    cin >> op;

    if(op==1){
        vec.erase(vec.begin() + index);
        cout << "Cadastro removido com sucesso" << endl;
        system("pause");
    }else{
        cout << "Cadastro nao removido" << endl;
        system("pause");
    }
    //remove(vec.begin(),vec.end(),item); //remove by value

    return vec;
}
```

```
vector<string> replaceProf(vector <string> &vec){

    string item, new_name;

    cout << "Digite o nome que deseja atualizar:" << endl;
    cin >> item;

    if (find(vec.begin(), vec.end(), item)!=vec.end()) {
        cout << "Registro Encontrado." << endl;
        cout << "Digite o nome atualizado:" << endl;
        cin >> new_name;
        replace(vec.begin(), vec.end(), item, new_name);

    } else {
        cout << "Nao Registrado" << endl;
    }
    system("pause");

   return vec;
}
```

PRÓXIMOS PASSOS

- TERMINAR MENUS E CADASTROS DE ALUNOS E TURMAS
- ACRESCENTAR POO
- INCLUIR A CRIAÇÃO E LEITURA DE ARQUIVOS
- FAZER PROGRAMA DA GRADE HORÁRIA
- FAZER O SUMMARY (AS MÉDIAS POR TURMA E ALUNO, FREQUENCIAS, APROVAÇÕES)
- FAZER UMA GUI

OBRIGADA!

Maria Marcolina Cardoso