U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

PLATAFORMA DE GESTÃO DE HORÁRIOS DE LEIC

Grupo 44 - AED

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

A NOSSA EQUIPA

Carlos Daniel Lopes Rebelo

202108885

Jaime Francisco Rodrigues Fonseca

202198789

Hélder Gabriel Silva Costa

202108719

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

ÍNDICE

01	Descrição do Problema
02	Descrição da solução
03	Diagrama e enumeração de classes
04	Lista de Funcionalidades Implementadas
05	Destaque de Funcionalidade
06	Principais Dificuldades Encontradas

Descrição do Problema

- Desenvolvimento de um sistema capaz de ajudar na gestão de horários na linguagem C++

- Implementação de funcionalidades que o auxiliem

Descrição da solução

1. Leitura de dados
2. Tratamento de dados
3. Manipulação de dados

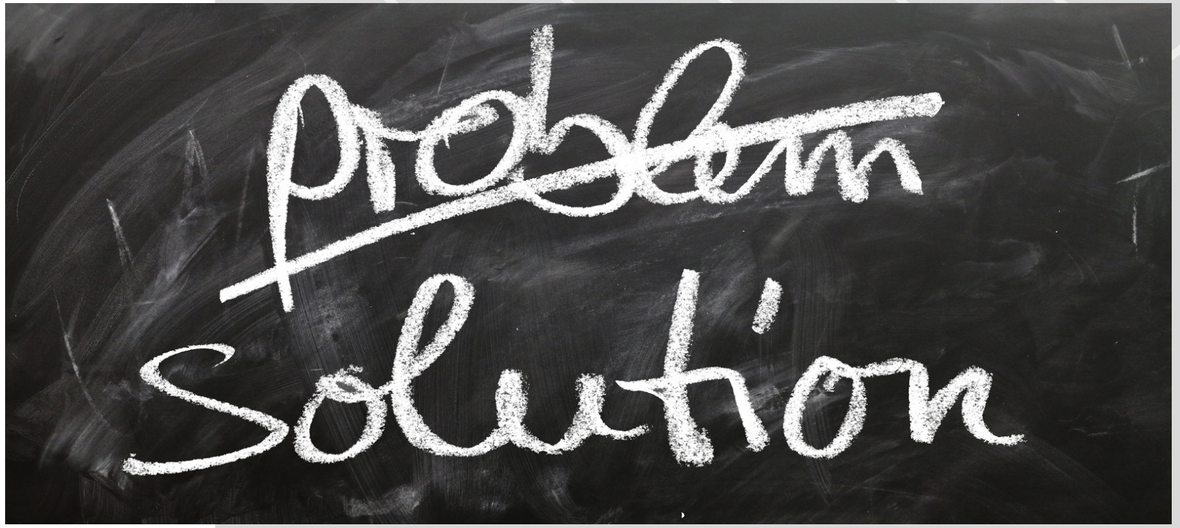

Diagrama e enumeração de classes

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

C Curso	
C FileReader	
C PedidoAlteracao	
C Slot	
C Student	
C studentComparator	
C studentComparator2	
C studentComparatorCode	
C studentComparatorDecreasingCode	
C Turma	
C turmaComparator	

Lista de Funcionalidades Implementadas

Algoritmo de Pesquisa

```
Menu de informações dos estudantes

1.Procurar por nome de estudante
2.Procurar por número do estudante
0.Voltar
```

```
Menu de informações dos estudantes

1.Procurar por nome de estudante
2.Procurar por número do estudante
0.Voltar

Escolha opção: 1

Inserir nome do estudante: maNuEl aLcIno
```

```
Menu de informações dos estudantes

1.Procurar por nome de estudante
2.Procurar por número mecanográfico do estudante
0.Voltar

Escolha opção: 2

Inserir número mecanográfico do estudante: 202075807
```

```
Estudante encontrado!
Nome: Manuel Alcino     Número de estudante: 202075807

1.Ver UC's/Turmas em que está inscrito
2.Ver horário
0.Voltar

Escolha opção:
```

Lista de Funcionalidades Implementadas

Algoritmo de Ordenação

1

```
Listagens

O que deseja fazer?
1. Ver todos os estudantes inscritos
2. Ver todos os estudantes inscritos em um ano específico
3. Ver todas as turmas
4. Ver número de estudantes inscritos em todas as UCs
5. Ver estudantes inscritos em certo número de UCs
0. Voltar
```

2

```
1.Ver estudantes inscritos em mais do que n UCs
2.Ver estudantes inscritos em n UCs
3.Ver estudantes inscritos em menos de n UCs
0.Voltar
```

3

```
Escolha opção: 1

Definir um número de UCs: 5

Escolha o tipo de ordenação:
1.Crescente
2.Decrescente

Escolha opção: 1
```

4

```
Conceicao               6
Claudia                 6
Cidalia                 6
Celina                  6
Rafaela                 6
Celeste                 6
Raquel                  6
Matilde                 6
Humberto                6
Gustavo                 6
Goncalo                 6
Gloria                  6
Manuel Tadeu            6
Galileu                 6
Gabriela                6
Manuel Valter           6
Erica                   6
Mateus                  6
Esmeralda               6
Jose Nelson             7
Amelia                  7
Manuel Carlos           7
Leonor                  7
Raul                    7
Bento                   7
Valentina               7
Marta                   7
Joao Rodrigo            7
```

Destaque de Funcionalidade

```
struct studentComparator
{
    bool operator()(Student* s1, Student* s2) const {
        if (s1->get_student_Code() != s2->get_student_Code()) return (s1->get_Name() < s2->get_Name());
        return false;
    }
};

struct turmaComparator
{
    bool operator()(Turma* t1, Turma* t2) const {
        if (t1->get_ucCode() != t2->get_ucCode()) return (t1->get_ucCode() < t2->get_ucCode());
        else if (t1->get_turmaCode() != t2->get_turmaCode()) return (t1->get_turmaCode() < t2->get_turmaCode());
        return false;
    }
};
```

```
string studentCode = line_vector[0],
studentName = line_vector[1],
ucCode = line_vector[2],
turmaCode = line_vector[3].substr( pos: 0,  n: line_vector[3].find( s: "\r"));
Turma *turma = new Turma( &: turmaCode,  &: ucCode);
auto itr : const_iterator<...> = allTurmas.find( k: turma);
if (itr != allTurmas.end()) {
    turma = *itr;
    Student *student = new Student( &: studentName,  &: studentCode);
    auto it : const_iterator<...> = students.find( k: student);
    if (it != students.end()) {
        student = *it;
    } else students.insert( v: student);
    turma->AddStudent(student);
    student->UpdateTurmas(turma);
} else {
    cout << "error : no class found";
}
```

```
Turma *turma = new Turma( &: turmaCode,  &: ucCode);
auto itr : const_iterator<...> = allTurmas.find( k: turma);
if (itr != allTurmas.end()) {
    turma = *itr;
} else {
    allTurmas.insert( v: turma);
}
```

Principais dificuldades

LEITURA/ANÁLISE DADOS

- Arranque
- Estrutura

GITHUB

- Primeiro contacto em contexto de trabalho
- Metodologia de trabalho
- Branches
- Pull Requests

DEPENDÊNCIAS C++

- Maior desafio ao longo do trabalho

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

OBRIGADO PELA ATENÇÃO!