FEUP
Universidade do Porto
Faculdade de Engenharia

Projeto AED

LUIS RELVAS – UP20210866

RODRIGO DOS SANTOS – UP202108749

RODRIGO RODRIGUES – UP202108847

Prof.
Ana Paula Rocha
Algoritmo e Estrutura de Dados

Índice

Problema

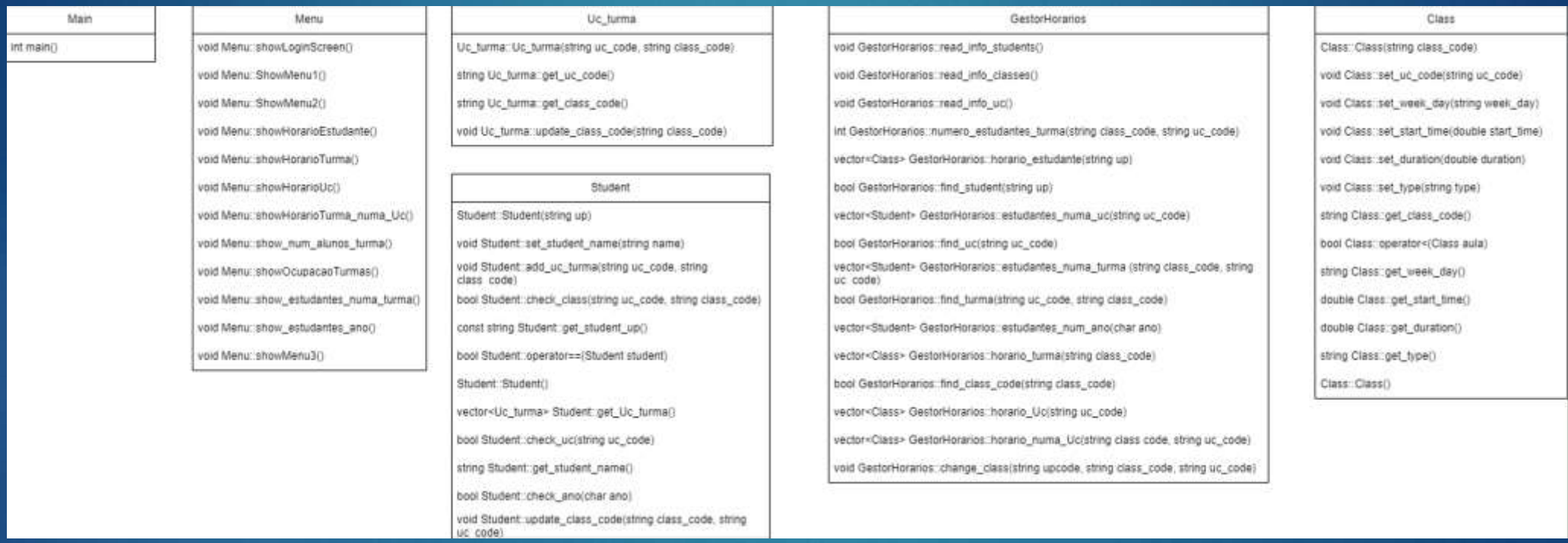
- Á semelhança do que nos foi pedido, este trabalho foi desenvolvido a pensar na gestão dos horários após a sua elaboração;

Resposta

- Como resposta ao problema tentamos sempre simplificar o código de modo a que seja fácil de interpretar;
- Para tornar a leitura e o desenvolvimento do código mais fácil, optamos por definir tudo na forma de classes;
- Esquema :
- Leitura dos dados ->Processamento dos dados-> Salvar os dados

->Dar uma resposta ao pedido do utilizador;

Diagrama


Main
int main()
Menu
void Menu::showLoginScreen()
void Menu::ShowMenu1()
void Menu::ShowMenu2()
void Menu::showHorarioEstudante()
void Menu::showHorarioTurma()
void Menu::showHorarioUc()
void Menu::showHorarioTurma_numa_Uc()
void Menu::show_num_alunos_turma()
void Menu::showOcupacaoTurmas()
void Menu::show_estudantes_numa_turma()
void Menu::show_estudantes_ano()
void Menu::showMenu3()
Uc_turma
Uc_turma::Uc_turma(string uc_code, string class_code)
string Uc_turma::get_uc_code()
string Uc_turma::get_class_code()
void Uc_turma::update_class_code(string class_code)
Student
Student::Student(string up)
void Student::set_student_name(string name)
void Student::add_uc_turma(string uc_code, string class_code)
bool Student::check_class(string uc_code, string class_code)
const string Student::get_student_up()
bool Student::operator==(Student student)
Student::Student()
vector<Uc_turma> Student::get_Uc_turma()
bool Student::check_uc(string uc_code)
string Student::get_student_name()
bool Student::check_ano(char ano)
void Student::update_class_code(string class_code, string uc_code)
GestorHorarios
void GestorHorarios::read_info_students()
void GestorHorarios::read_info_classes()
void GestorHorarios::read_info_uc()
int GestorHorarios::numero_estudantes_turma(string class_code, string uc_code)
vector<Class> GestorHorarios::horario_estudante(string up)
bool GestorHorarios::find_student(string up)
vector<Student> GestorHorarios::estudantes_numa_uc(string uc_code)
bool GestorHorarios::find_uc(string uc_code)
vector<Student> GestorHorarios::estudantes_numa_turma (string class_code, string uc_code)
bool GestorHorarios::find_turma(string uc_code, string class_code)
vector<Student> GestorHorarios::estudantes_num_ano(char ano)
vector<Class> GestorHorarios::horario_turma(string class_code)
bool GestorHorarios::find_class_code(string class_code)
vector<Class> GestorHorarios::horario_Uc(string uc_code)
vector<Class> GestorHorarios::horario_numa_Uc(string class code, string uc_code)
void GestorHorarios::change_class(string upcode, string class_code, string uc_code)
Class
Class::Class(string class_code)
void Class::set_uc_code(string uc_code)
void Class::set_week_day(string week_day)
void Class::set_start_time(double start_time)
void Class::set_duration(double duration)
void Class::set_type(string type)
string Class::get_class_code()
bool Class::operator<(Class aula)
string Class::get_week_day()
double Class::get_start_time()
double Class::get_duration()
string Class::get_type()
Class::Class()

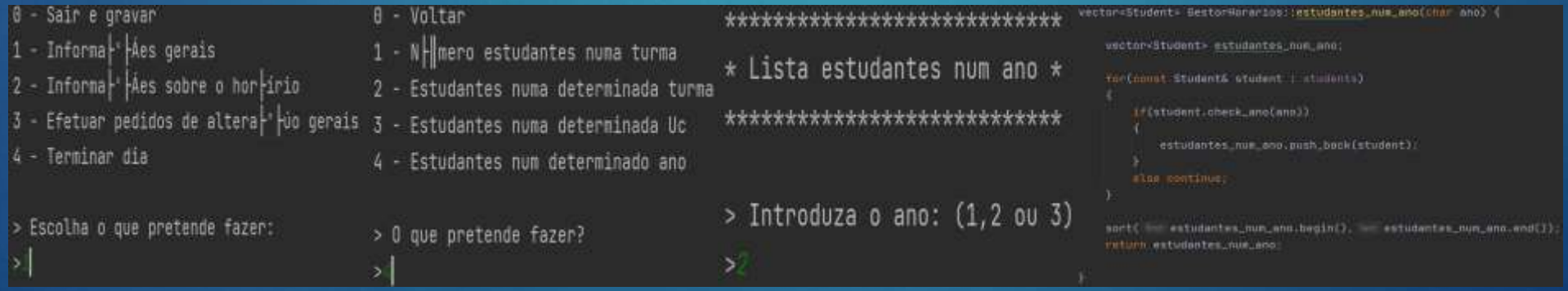
Dificuldades

- No desenvolvimento do projeto deparamo-nos com alguns obstáculos, sendo alguns deles mais fáceis de resolver que os outros e desse modo, para nós, as maiores dificuldades que encontramos foram:

1. Programação paralela -> o facto de usarmos classes que dependem de outras classes dificultou-nos bastante o trabalho visto que estávamos a desenvolve-las em separado (utilizamos o GitHub);
2. Ponto 3 -> Terminado o ponto 2, começamos logo a pensar no ponto 3, no entanto e tendo em conta que o nosso código era limitado em alguns aspetos, tivemos que restruturar grande parte do nosso código;

Funcionalidades

- Todas as nossas funcionalidades implementadas foram desenvolvidas com base em dois métodos fundamentais:
- Verificar e prosseguir (sempre que respondemos a um pedido do utilizador verificamos se tal pedido é ou não possível de concretizar);
- Listagens:



Funcionalidades:

Pesquisa

```
bool Student::check_ano(char ano) const {

    bool flag = false;

    for(const Uc_turma& aula : horario)
    {
        if(aula.get_class_code()[0] == ano)
        {
            flag = true;
            return flag;
        }
        else continue;
    }

    return flag;

}
```

Leitura

```
ifstream in;
in.open( s: "C:\\Users\\LuisRelvas\\Desktop\\shedule\\students_classes.csv");
string line,word, word1;

getline( &: in,  &: line);

getline( &: in,  &: line);
stringstream str( str: line);

getline( &: str,  &: word,  delim: ',');
Student student = Student( up: word);

getline( &: str,  &: word,  delim: ',');
student.set_student_name( name: word);

getline( &: str,  &: word,  delim: ',');
getline( &: str,  &: word1,  delim: '\r');
student.add_uc_turma( uc_code: word,  class_code: word1);
```

Main Feature:

- Neste projeto decidimos também melhorar e aprofundar a parte estética, desse modo decidimos criar um menu com todas as funcionalidades que o nosso código proporciona e desse modo a nossa main feature é o nosso menu, deixamos aqui alguns exemplos:

```
0 - Sair e gravar
1 - Informacoes gerais
2 - Informacoes sobre o horario
3 - Efetuar pedidos de alteracao gerais
4 - Terminar dia
```

```
0 - Voltar
1 - Ocupacao das turmas
2 - Estudantes numa determinada turma
3 - Estudantes numa determinada Uc
4 - Estudantes num determinado ano
5 - Estudantes por unidade curricular
```

```
201920727 Ines
201930842 Rosarinho
202020047 Maria
202020132 Matilde
202020217 Leonor
202020302 Carolina
202020387 Beatriz
202020472 Mariana
202020557 Ana
202020642 Sofia
202020812 Margarida
202020897 Francisca
202020982 Lara
```

Esforço:

- O tempo dedicado, por cada elemento do grupo, ao projeto foi aproximadamente o mesmo;

1. Luis -> Encarregue dos ficheiros (Class.ccp, Class.h, Uc_turma.h, PowerPoint , Doxyfile);
2. Rodrigo dos Santos -> Encarregue dos ficheiros (Menu.h, Menu.cpp, Uc_turma.cpp, Doxyfile);
3. Rodrigo Rodrigues -> Encarregue dos ficheiros (GestorHorarios.cpp, GestorHorarios.h, Student.cpp, Student.h);

Conclusão

- Após o projeto estar concluído concluímos que o mesmo foi importante para aprofundar os nossos conhecimentos em áreas como:
- Programação em equipa/paralelo;
- Programação extensa;
- A importância do Doxyfile na documentação;