GESTÃO DE HORÁRIOS

Adriano Machado /Francisco Pires da Ana /José Pedro Evans

DESCRIÇÃO DO PROBLEMA

Este projeto tem como objetivo o desenvolvimento de um sistema capaz de ajudar na gestão de horários (alteração, visualização). Para este efeito, deve-se escolher as estruturas de dados mais apropriadas e eficientes.

Estruturas de dados utilizadas

Ao longo do nosso trabalho usamos quatro estruturas de dados da stl (set, map, vector, queue).
Set's e map's são implementados recorrendo a árvores binárias balanceadas (*red black tree*).

- **Set**

  Esta estrutura de dados é usada, por exemplo, no armazenamento dos estudantes. Ao longo do programa usamos os seguintes métodos.

Método	Complexidade Temporal
Insert()	O(log n)
Erase(valor)	O(log n)
Size()	O(1)
Find()	O(log n)

- **Map**

  Esta estrutura de dados é usada, por exemplo, para fazer corresponder um dia da semana a um conjunto de aulas.

Método	Complexidade Temporal
Operador [ ]	O(log n)
Empty()	O(1)

Estruturas de dados utilizadas

- **Vector**

  Esta estrutura de dados foi usada, no armazenamento de horários de um determinado estudante.

Método	Complexidade temporal
Push_back()	O(1)
At()	O(1)
Size()	O(1)

Implementamos um método, binarySearchSchedules que aplica pesquisa binária no vetor de horários e retorna o índice no horário da UcClass pretendida. Este método apresenta complexidade temporal O(logN).

- **Queue**

  Esta estrutura de dados foi utilizada no armazenamento temporário de pedidos de trocas/inscrições/desinscrições de turmas.

Método	Complexidade Temporal
Push()	O(logN)
Empty()	O(1)
Pop()	O(logN)
Front()	O(1)

Classes

Student
id: str name: str classes: vector<UcClass>

UcClass
ucId: str classId: str

Slot
weekDay: str startTime: float endTime: float type: str

Request
student: Student desiredClass: UcClass type: str

ClassSchedule
ucClass: UcClass slots: vector<Slot> students: set<Student>

ScheduleManager
students: set<Student> schedules: vector<ClassSchedule> changingRequests: queue <Request> removalRequests: queue <Request> enrollmentRequests: queue<Request> rejectedRequests: vector<Request>

Funcionalidades implementadas

Leitura dos dados fornecidos

O método readFiles() é chamado. Este por sua vez chama os métodos:

- createSchedules() – Lê o ficheiro classes_per_uc.csv e adiciona ao vetor de horários(schedules) objetos da classe ClassSchedule com a devida UcClass, mas com o vetor de slots e sets de estudantes vazios;
- setSchedules() – Lê o ficheiro classes.csv e atualiza os objetos do vetor schedules com os devidos slots,
- createStudents() – Lê o ficheiro students_classes.csv e adiciona objetos Student ao set Students; adiciona também a cada objeto do vetor schedules os estudantes que pertencem ao mesmo par turma/cadeira (UcClass);

```
1  UcCode,ClassCode
2  L.EIC001,1LEIC01
3  L.EIC001,1LEIC02
```

Fig 1. – classes_per_uc.csv

```
1  ClassCode,UcCode,Weekday,StartHour,Duration,Type
2  1LEIC01,L.EIC001,Monday,10.5,1.5,TP
3  1LEIC02,L.EIC001,Thursday,9.5,1.5,TP
```

Fig 2. – classes.csv

```
1  StudentCode,StudentName,UcCode,ClassCode
2  201920727,Ines,L.EIC001,1LEIC05
3  201920727,Ines,L.EIC002,1LEIC05
```

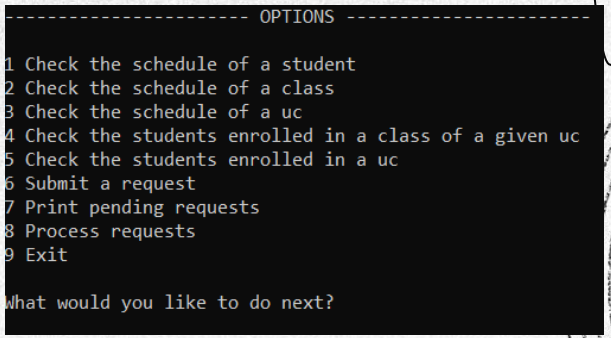
Fig 3. – students_classes.csv

Funcionalidades implementadas

Visualização dos dados fornecidos

O nosso programa permite-nos visualizar:

- Horário de um estudante;
- Horário de uma turma;
- Horário de uma cadeira;
- Estudantes inscritos num dado par turma/cadeira;
- Estudantes inscritos numa cadeira



Funcionalidades implementadas

Pedidos de alteração de horários

Existem três tipos de pedidos:

- Pedido de mudança de turma a uma dada cadeira (Changing);
- Pedido de inscrição numa dada cadeira (Enrollment)
- Pedido de cancelamento de inscrição (Removal)

Destaque de Funcionalidade

Dificuldades encontradas / Esforço