U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Algoritmos e Estruturas de Dados

Horários de estudantes L.EIC

Projecto 1 | 2LEIC10 | 6 de Novembro 2022

Ntsay Zacarias – **up202008863**
Eriton Felisberto Naife – **up202008859**

Problema

Há necessidade da criação de um sistema eficiente de gestão de horários dos estudantes L.EIC, seja na consulta ou na inserção de dados.

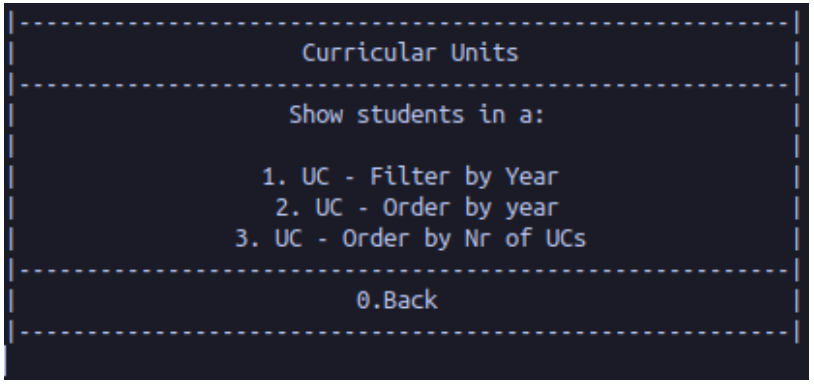
Em termos de consulta de dados é importante que o utilizador tenha toda a informação disponível para apresentação, garantindo opções de ordenação e filtragem dos mesmos.

Na inserção de dados é importante preservar a integridade dos mesmos, e que seja o mais conveniente possível para o utilizador, fornecendo feedback em qualquer eventualidade .

LEIC'S SCHEDULE MANAGEMENT SYSTEM

Este é um sistema de gestão de horários dos estudantes da Licenciatura em Engenharia de Informática e Computação. O programa é Terminal-based concebido para POSIX Systems (Mac OS X, Linux) .

Oferece features como listagem (completa e parcial) dos dados disponíveis, bem como ordenação a critério do utilizador. É também possível adicionar ou remover estudantes de turmas através de pedidos feitos pelos mesmos, sendo estes aceites ou não de acordo com critérios definidos.



Estruturas de dados e Algoritmos

Foram maioritariamente usadas classes da STL de C++ como std::vector, std::map e std::set para armazenar dados. Estes dois últimos implementam uma **Self Balacing Binary Search Tree.** Foram também usados algoritmos como std::sort, Merge Sort e Bubble Sort para ordenação.

Sr. No.	Algorithm	Time Complexity
1.	Search	O(log n)
2.	Insert	O(log n)
3.	Delete	O(log n)

Self Bacing Binary Search Tree

Function	Time Complexity
sort(v.begin( ), v.end( ))	Theta(nlog(n))
reverse(v.begin( ), v.end( ))	O(n)
v.push_back(x)	O(1)
v.pop_back(x)	O(1)
v.size()	O(1)
v.clear()	O(n)
v.erase()	O(n)

std::vector

Time Complexity

- Best Case - O(N log N)
- Average Case- O(N log N)
- Worse Case- O(N log N)

std::sort

Funcionalidades Implementadas

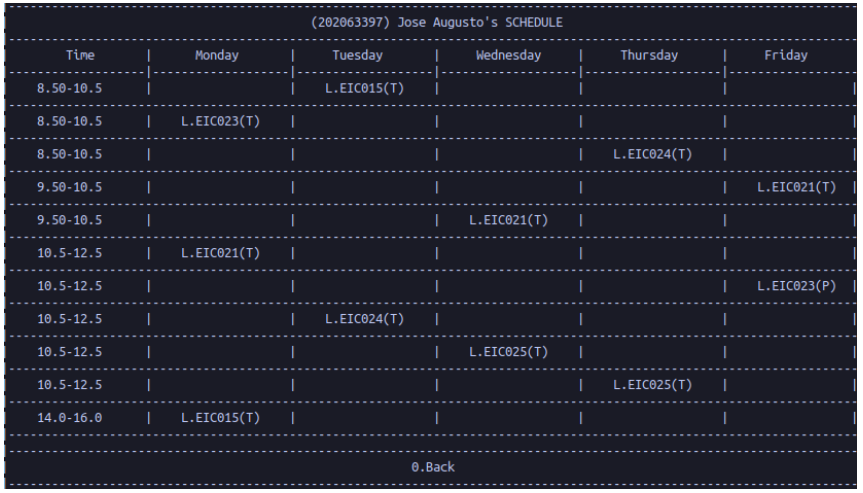
- Listings
  - Students
    - Classes
      - Sort in UC order
      - Sort in Lexicographical order
    - Years
      - Filter by Year
        - Sort in Lexicographical order
    - Curricular Units
      - Filter by Year
      - Sort by Year
      - Sort by nº of UCs
  - Classes
    - Sort by Year
  - UCs
  - Schedules
    - Student
      - Search by Student's code
      - Search by Student's name
    - Class
      - Filter by Class
- Resquests
  - Students
    - Add
    - Remove
    - Replace

Feature de destaque

Geração dinâmica de horários

Para melhorar a experiência de utilizador sentimos a necessidade de criar uma função que gera horários sob demanda, de acordo com o número de aulas que qualquer aluno, turma ou unidade curricular tiver. Juntamente, criamos uma função que permite, antes da impressão do horário, que os slots estejam ordenados adequadamente.

Com performance em mente decidimos implementar o algoritmo Merge Sort (com melhor complexidade temporal)



(202063397) Jose Augusto's SCHEDULE

Time	Monday	Tuesday	Wednesday	Thursday	Friday
8.50-10.5		L.EIC015(T)			
8.50-10.5	L.EIC023(T)				
8.50-10.5				L.EIC024(T)	
9.50-10.5					L.EIC021(T)
9.50-10.5			L.EIC021(T)		
10.5-12.5	L.EIC021(T)				
10.5-12.5					L.EIC023(P)
10.5-12.5		L.EIC024(T)			
10.5-12.5			L.EIC025(T)		
10.5-12.5				L.EIC025(T)	
14.0-16.0	L.EIC015(T)				

0.Back

Dificuldades

- Importação de dados dos ficheiros .csv
- A nossa solução inicial do problema exigia o uso de estruturas de dados para cada objeto criado. No entanto, isto causou vários problemas com memória o que implicou durante o desenvolvimento da aplicação uma grande mudança de filosofia, levando assim a um gasto de tempo e trabalho superior ao esperado.

Esforço Individual

- Ntsay Zacarias – Listagens , interface do utilizador, algoritmos de ordenação e pesquisa
- Eriton Naife – Pedidos, organização de classes, documentação com Doxygen