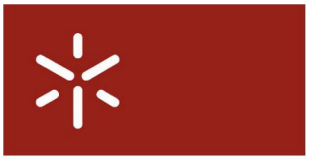
Universidade do Minho

Departamento de Informática



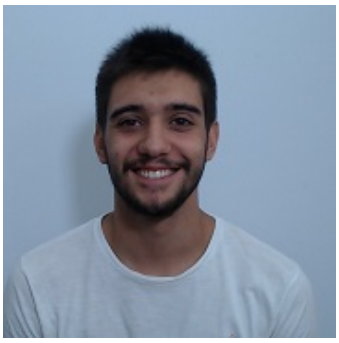
Universidade do Minho

Laboratórios de Informática III

Sistema de Gestão e Consulta de Recomendações de Negócios na Plataforma Yelp

---

Grupo 19- Parte C



André Vieira

Francisco Andrade

Joana Sousa

André Gonçalves Vieira A90166
Francisco Alves Andrade A89513
Joana Castro e Sousa A83614

Maio de 2021

Conteúdo

1 Introdução

Este projecto foi apresentado no presente ano lectivo no âmbito da unidade curricular de Laboratórios de Informática III do curso de Mestrado Integrado em Engenharia Informática.

Nesta primeira fase do projeto a linguagem para a execução do mesmo foi C, proporcionando a oportunidade de desenvolver um projeto em larga escala com o objetivo de criar uma aplicação que simule um sistema de Gestão e Consulta de Recomendações na Plataforma Yelp, mantendo sempre proteção de dados através do encapsulamento dos mesmos.

Ao longo do desenvolvimento deste projeto, consideramos que o maior desafio foi a implementação das estruturas onde seria organizada a informacão, de forma a que o seu acesso fosse rápido e eficiente. Além disso, consideramos um desafio a libertacão da memória total do programa.

Este documento visa apresentar as soluções adotadas para o desenvolvimento deste projeto, bem como explicitar todas as estruturas implementadas.

2 Estrutura Do Projeto

Sendo uma aplicação de um projeto em larga escala, a aplicação aceita a entrada de três ficheiros de texto: um ficheiro com uma listagem de utilizadores, um ficheiro com uma listagem de negócios e um ficheiro final com os dados das reviews.
Implementou-se uma estrutura semelhante para o tratamento de dados dos ficheiros.

2.1 Estrutura de Dados

Essa estrutura tem como objetivo tornar o trabalho prático mais eficiente e por esta razão implementou-se um *SGR* que contém os diversos catálogos. Cada *Catalogo* é uma Hash Table de 38 posições, correspondendo aos 38 possíveis casos que podem dar início ao id, ou seja, as 26 letras do alfabeto com os 10 dígitos e ainda '_' e '-', com apontadores para *AVL's* de forma a tornar mais eficiente a procura de informação. Desta forma, cada AVL é composta por um inteiro denominado size, e apontadores para *"Node"* e dentro de cada um é encontrada a informação relevante para o desenvolvimento do projeto.
Como os dados apresentados nos ficheiros .csv têm diversos dados para processar, adoptamos uma solução que, na nossa ideia, seria mais eficiente, utilizando Hash Tables e árvores binárias AVL. As Hash Tables foram utilizadas para indexar os dados que se pretendiam procurar, de forma mais rápida e as AVL's tornaram o processo mais organizado.

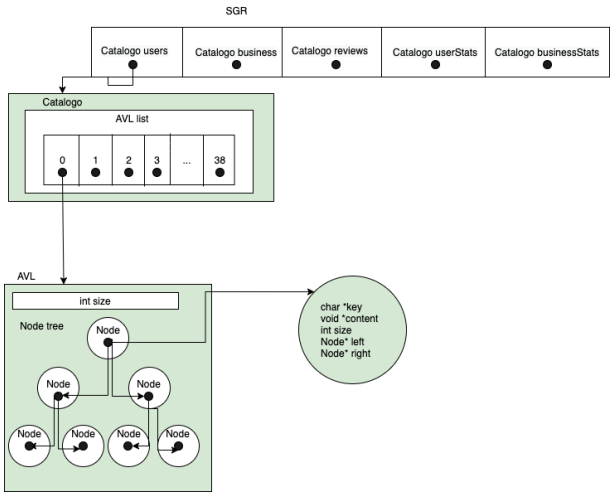


Figura 1: Estrutura de Catalogo

É de notar, que a nossa AVL contém diversos nodos e, em cada um, contém a informação que será posteriormente necessária para o desenvolvimento do nosso projeto, pois estruturamos, de forma a que a procura de informação se tornasse mais prática e sistemática. Podemos observar a informação contida em cada nodo na imagem apresentada de seguida:

```
/* Estrutura de uma AVL- tree Balanceada de P

struct avl
{
    int size;
    Node tree;
};

/* Estrutura de um nodo de uma AVL com a info

struct node
{
    char *key;
    void *content;
    int size;
    Node left;
    Node right;
};
```

Figura 2: Estrutura da AVL e Node

3 Módulos

Como um dos objetivos era o da proteção dos dados utilizando a técnica de encapsulamento, o projeto foi implementado em diversos módulos.
Um projeto de larga escala como o que foi apresentado, necessita de diversos módulos para que tais dados importantes sejam bem estruturados e protegidos.

- ***avl.h:*** módulo que contém todas funções responsáveis pelo processamento da informação recebida e dos dados em memória, disponibilizando o espaço necessário para todas as estruturas de dados.
- ***business.h:*** módulo onde estão todas as informações que serão inseridas nas AVL's relativamente ao ficheiro business.
- ***catalogo.h:*** módulo onde está implementada a hash table e todas as funções necessárias para armazenar informação relativa a cada ficheiro.
- ***controller.h:*** módulo que trata da interpretação dos comandos fornecidos pelo utilizador.
- ***parser.h:*** módulo que tem como objetivo a leitura e divisão da informação contida em cada ficheiro.
- ***review.h:*** módulo onde estão todas as informações que serão inseridas nas AVL's relativamente ao ficheiro reviews.
- ***sgr.h:*** módulo com a estrutura inicial do projeto.
- ***stats.h:*** módulo que interliga informações dos diferentes catálogos de dados (user, business e review) para criar outros novos catálogo (userStats, cityStats e businessStats).
- ***user.h:*** módulo onde estão todas as informações que serão inseridas nas AVL's relativamente ao ficheiro users.
- ***view.h:*** módulo que apresenta todas as funcionalidades no terminal para o uso do utilizador.

4 Tarefas/Queries:

4.1 Query 1:

Esta primeira interação corresponde à leitura dos três ficheiros e criação das respetivas estruturas de dados. Toda esta interação utiliza os módulos e métodos criados passando pelo parse de cada ficheiro para posteriormente executar a sua leitura e colocar os dados na estrutura.

4.2 Query 2:

A segunda Query pede uma lista de nomes e o número total de negócios iniciados por uma dada letra. Para tal, inicializamos uma *table* com uma coluna, onde serão armazenados os nomes dos negócios, e procuramos em todas as posições do catálogo, através da travessia das árvores balanceadas, comparando o primeiro elemento do campo *name* do *business* e colocamos o seu nome na *table*. O campo *header* da *table* exibe o número total de negócios cujo nome inicia pela a letra pretendida.

4.3 Query 3:

Nesta Query pretende-se, através do id do negócio, fornecer as informações do mesmo, nomeadamente o nome, a cidade, o estado, as stars e o número total reviews. Começamos por inicializar uma *table* com 5 colunas, onde armazenaremos as informações relativas ao negócio em questão, sendo estas o seu nome, a cidade e o estado onde se localiza, o seu número médio de stars e o número total de stars. Posteriormente, acedemos ao catálogo *BusinessStats* e retiramos as informaçoes relativas ao negócio desejado.

4.4 Query 4:

A Query 4 pretende, recebendo um id de um *user*, fornecer uma lista de todos os negócios que tenham uma *review* feita por este. Começamos por inicializar uma *table* com duas colunas, onde serão guardados o id e o nome do negócio que teve a *review* do *user* fornecido. Posteriormente, percorremos o catálogo de reviews e as suas árvores balanceadas e comparamos o id do *user* que fez a *review* com o id do *user* proporcionado. Se este corresponder, guardamos o id do negócio na primeira coluna da *table* e o nome do mesmo na segunda.

4.5 Query 5:

Nesta Query, o objetivo é, recebendo um determinado número de stars e uma cidade, retornar uma lista de negócios com um número, igual ou superior, de stars na cidade fornecida. Para este fim, inicializamos uma *table* com duas colunas, onde serão armazenados o id e o nome do negócio. Posteriormente, procuramos no catálogo de *cityStats* a cidade que o utilizador pretende, e percorremos a respetiva árvore balanceada comparando o número de stars de cada negócio com o número fornecido. Sempre que o número de estrelas for maior ou igual ao pretendido, armazenamos na *table* o id e o nome do negócio.

4.6 Query 6:

Na sexta Query, dado um número inteiro n, indica uma lista com os n melhores negócios, de acordo com o número médio de stars, em cada cidade. Primeiramente, inicializamos uma *table* com 5 colunas onde serão armazenados a cidade, o número que ocupa no top, o id,o nome e o número médio de stars do negócio, respetivamente. De seguida, percorremos o catalogo de *cityStats* e colocamos os primeiros n negócios encontrados nas árvores balanceadas, uma vez que este catálogo está organizado pelo número médio de stars.

4.7 Query 7:

A Query 7 tem como finalidade fornecer uma lista de ids de *users*, e também o número total destes, que tenham feito *reviews* em mais de um estado. Começamos por inicializar uma *table* de uma linha com uma coluna, e de seguida usamos uma função auxiliar, que percorre uma AVL de *Users*, para inserir na table o id dos utilizadores que tenham feito uma *Review* em mais do que um estado.

4.8 Query 8:

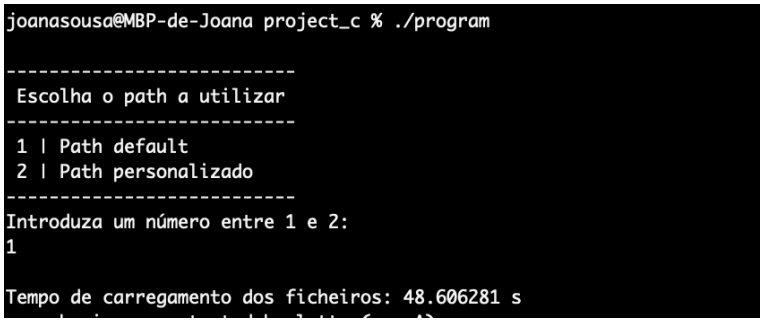
Esta Query pretende, dado um número inteiro n e uma categoria, determinar uma lista com os top n negócios que pertencem à categoria fornecida. Para tal, inicializamos uma *table* com 3 colunas, onde iremos armazenar o id, o nome e o número de stars do negócio e criamos um array de uma nova *struct* (*Q8*) que servirá para armazenar o id, nome e número médio de estrelas dos *Businesses* e será ordenado pelo número de estrelas destes. Depois, inserimos os n *Businesses* com mais estrelas e inserimos na table.

4.9 Query 9:

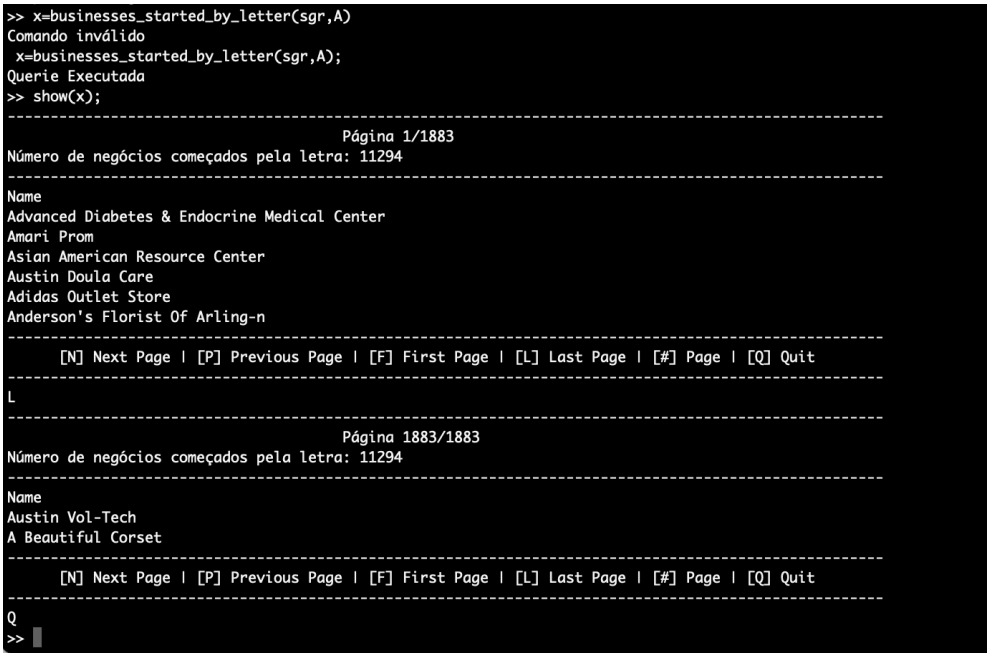
A última Query tem como objetivo, através de uma palavra, determinar o id das *reviews* que contenham a palavra pretendida no campo *text* da review. Como foi feito para as Queries anteriores, criamos uma *table* com uma coluna, onde será armazenado o id das *Reviews* e, após isto, pesquisamos na AVL as *Reviews*, com uma função auxiliar, e inserimos os id's das que apresentaram a palavra pretendida.

5 Interação com o utilizador

De seguida, iremos observar as várias funcionalidades que o nosso projeto consegue desenvolver face ao que era requerido no enunciado. Primeiramente, quando inicializamos o programa com o ficheiro executável, é questionado ao utilizador que path deseja utilizar para fazer o carregamento dos ficheiro como base para a resolução de todas as funcionalidades.



De acordo com a figura abaixo apresentada, conseguimos entender que a funcionalidade de colocar o valor das queries numa variável e de seguida fazer a apresentação da mesma através da funcionalidade ***"show"*** é bem executada. Além disso, a ***"paginação"*** requerida também foi efetuada com sucesso, tendo a opção de voltar atrás, seguir para a página seguinte, visualizar a primeira ou a última página, ter a informação em que página se encontra e por fim, sair da visualização da querie para posteriormente executar mais funcionalidades.



A imagem seguinte mostra que as ferramentas relativas ao carregamento tanto ***"de um ficheiro CSV"*** como ***"para um ficheiro CSV"*** estão funcionais.

```
>> toCSV(x,|,../Files);
Loaded to CSV
>> y= fromCSV(../Files,|);
Loaded from CSV
```

Relativamente à projeção de colunas e à indexação ficarem alocadas em variáveis aleatórias, recorrendo novamente ao ***"show"*** conseguimos visualizar que está a ser corretamente indexada.

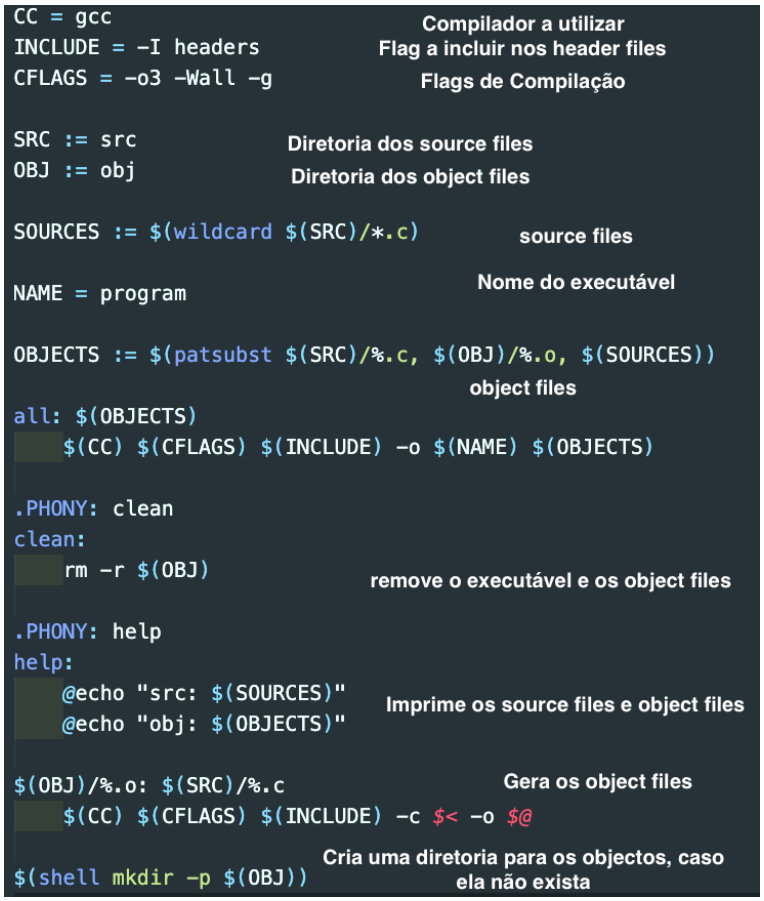
```
>> y=proj(x,0);
Projeção Executada
>> z=x[345][0];
Indexação Executada
>> show(z);
--------------------------------------------------------------------------------------------------------
                                    Página 1/1

--------------------------------------------------------------------------------------------------------
Name
AMC Company
--------------------------------------------------------------------------------------------------------
      [N] Next Page | [P] Previous Page | [F] First Page | [L] Last Page | [#] Page | [Q] Quit
--------------------------------------------------------------------------------------------------------
```

Por último, para sair do programa, utilizamos a ferramenta ***"quit"*** que depois de dar free à memória alocada, fecha assim o programa.

```
>> x= businesses_started_by_letter(sgr, A);
Querie Executada
>> quit;
A fechar o programa;
```

6 Makefile

De modo a uma rápida compilação e interação de todos os módulos, foi utilizada a seguinte 'makefile':



7 Conclusão

Terminado este primeiro projecto da unidade curricular de Laboratórios de Informática III, podemos afirmar que foi algo desafiante e enriquecedor para o nosso grupo. Este foi o projecto que até ao momento gerou mais dificuldades a todos os elementos do grupo. Ao longo da realização do mesmo fomos confrontados com situações novas que puxaram pelo nosso espírito de autonomia.

De acordo com os dados obtidos, é verificado que o maior problema nesta implementação é o tempo de execução do carregamento dos ficheiros csv, uma vez carregados, as queries selecionadas são executadas de imediato.

De uma forma global, acreditamos que o objetivo do trabalho foi concretizado, apesar do tempo de processamento de dados ter margem para melhoria, sempre pondo em prática todos os conceitos aprendidos nas aulas tais como o encapsulamento dos dados. Todas as dificuldades foram, no entanto, ultrapassadas tanto com a aplicação de matéria lecionada em cadeiras anteriores como com o trabalho em equipa do grupo.

Além disso, sentimos que conseguimos aprofundar os nossos conhecimentos relativamente à linguagem imperativa C, estando assim ansiosos para a próxima etapa desta Unidade Curricular em que iremos por em prática e aprofundar os nossos conhecimentos da linguagem de Java.