FACULDADE DE INFORMÁTICA E ADMINISTRAÇÃO PAULISTA
BACHAREL EM SISTEMAS DE INFORMAÇÃO

ERIC DOWSLEY FIGUEIRA
GUSTAVO ALVES ROCHA GADINI
KAUÊ CESAR SANTANA
RAFAEL DORIA MARQUES
VITOR SANTOS ALVES

CHALLENGE KRAFTHEINZ

SPRINT 1 - CÓDIGOS DE ALTA PERFORMANCE

SÃO PAULO
2023

ERIC DOWSLEY FIGUEIRA
GUSTAVO ALVES ROCHA GADINI
KAUÊ CESAR SANTANA
RAFAEL DORIA MARQUES
VITOR SANTOS ALVES

CHALLENGE KRAFTHEINZ

SPRINT 1 - CÓDIGOS DE ALTA PERFORMANCE

Entrega do primeiro challenge sprint apresentado a disciplina de Códigos de Alta Performance da Faculdade de Informática e Administração Paulista.

Orientador (a): Patrícia Magna

SÃO PAULO
2023

SUMÁRIO

1. Descrição do Projeto:

A KraftHeinz, uma das principais empresas do ramo alimentício, lançou o challenge 2023 aos estudantes do segundo ano do curso de Sistemas de Informação. Onde explicitou a necessidade da empresa em ter um sistema capaz de fornecer uma compreensão clara da percepção dos clientes e consumidores em relação à sua agenda ESG.

A agenda ESG é uma abreviação para os termos em inglês Environmental, Social and Governance, que significam, respectivamente, Meio Ambiente, Social e Governança. Esses termos se referem a um conjunto de práticas e políticas que as empresas devem adotar para gerenciar seus impactos ambientais e sociais, bem como para garantir uma governança corporativa responsável e ética.

Atualmente, a KraftHeinz não possui um sistema unificado que armazene os dados da visão dos clientes sobre as ações da empresa. A solução que será desenvolvida deverá suprir essa demanda.

Em termos técnicos, o sistema deverá conter funções de coleta, seleção e armazenamento seguro de dados sobre críticas, sugestões e opniões gerais de consumidores sobre a empresa e seus produtos. Além disso, o programa deverá gerar consultas e vizualizações em formato de dashboards que permitam explorar essas informações.

2. Especificação de listas lineares:

Ao analisar e discutir sobre o desafio, levantamos alguns requisitos do nosso sistema de vizualização de dados:

- Ele deverá receber uma grande quantidade de dados, afim de tornar nossa aplicação mais realista.
- Deverá focar nos temas que possuem o maior número de interações (críticas, sugestões ou opniões), ou seja, a lista deverá ordenar do tema com maior quantidade de interações para o menor.
- E que a organização da lista deverá ser alterada frequentemente devido a quantidade de dados.

Portanto, decidimos utilizar em nosso projeto uma Lista Encadeada Genérica Simples, por possuir alocação dinâmica de dados, não ter um limite de inserção de elementos e pelo fato de ser eficiente quando a lista sofre muitas alterações.

3. Definição de atributos e operações:

3.1 Atributos:

Como escolhemos uma lista encadeada vamos precisar de uma classe interna do tipo NO. Essa classe deverá ter um atributo do tipo inteiro para armazenar a quantidade de interações de cada tema e um atributo do tipo NO para servir de ponteiro para o próximo nó. Além disso, a classe Lista deve ter um atributo do tipo NO que irá receber um valor nulo, esse atributo funcionará como uma referência da lista.

3.2 Operações:

Nossa classe Lista se baseará em dois métodos, o Inserir e o Apresentar. O método inserir tem as seguintes operações:

- Alocar novo nó: nesta operação será criado um objeto do tipo NO, esse objeto também vai receber o valor da quantidade de interações.
- Achar a posição em que o nó será posicionado: primeiramente, verifica-se se a lista está vazia ou se o valor a ser inserido é maior do que o valor do primeiro nó da lista. Se qualquer uma das duas condições for verdadeira, o novo nó deve ser inserido como o novo primeiro nó da lista. Nesse caso, o novo nó é colocado na frente do nó atual e a variável lista é atualizada para que aponte para o novo primeiro nó.

Caso contrário, o novo nó deve ser inserido em algum lugar no meio da lista. O método percorre a lista até encontrar o lugar correto para inserir o novo nó. Enquanto ele faz isso, o método compara o valor do novo nó com o valor do próximo nó, avançando um nó por vez até encontrar o lugar correto. Quando encontra o lugar correto, o método atualiza as referências dos nós envolvidos: o novo nó é inserido entre o nó atual e o nó seguinte, e o nó atual passa a apontar para o novo nó.

Já no método Apresentar, é criado uma váriavel auxiliar do tipo NO apenas para percorrer a lista e imprimir os respectivos nós e suas quantidades de interações.

4. Comparação de implementações:

4.1 Sequencial (alocação estática):

Algumas características da alocação estática que seriam úteis para o nosso projeto seriam a simplicidade, já que a memória é alocada em tempo de compilação e permanece fixa durante toda a execução do programa. E sua velocidade de acesso, já que a memória é alocada diretamente na área de dados do programa.

Enquanto aos pontos negativos, como essa implementação impõe um número limitado de elementos e é ineficiente quando a lista sofre muitas alterações, ela não seria funcional para nosso projeto, já que serão armazenados um número indeterminado de dados fazendo com que a lista se altere constantemente.

4.2 Encadeada (alocação dinâmica):

Como a alocação dinâmica de dados não tem um limite de inserção de elementos e é muito eficiente quando a lista sofre muitas alterações, essa seria a implementação ideal para o nosso projeto.

Todavia, além dessa implementação ser mais complexa, podendo levar a erros de programação, como vazamentos de memória e fragmentação, ela também é mais lenta que a alocação estática, já que o acesso aos dados é feito através de ponteiros e envolve a alocação e a desalocação de memória.