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 os estudantes do segundo ano do curso de Sistemas de Informação. Onde explicitou a necessidade da empresa em ter um sistema de informações capaz de fornecer uma compreensão clara da percepção dos clientes e consumidores do mercado 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 e consumidores sobre as ações da empresa, e 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 vizualização 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 da maior quantidade de interações para a menor.
- E que a organização da lista deverá ser alterada frequentemente devido a quantidade de dados.

Portanto, decidimos utilizar uma Lista Encadeada Genérica Simples, por possuir alocação dinâmica de dados, não ter um limite de inserção de elementos e pelo fato de ser muito 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: nesta etapa se a referência lista ainda estiver com um valor nulo o ponteiro do novo nó deverá apontar para mesma direção da lista e a lista passa a apontar para o novo nó.

Caso o a lista já esteja apontando para um outro nó e não para um valor nulo, o programa deverá checar se o nó que a lista está apontando é maior ou menor do que o novo nó que foi alocado.

Se o novo nó tiver uma quantidade de interações maior que o da lista o novo nó deve ser inserido como o novo primeiro nó da lista. Caso o contrário, o método percorre a lista até encontrar o lugar correto para inserir o novo nó, quando o método encontra o lugar correto, ele 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):

Como a alocação estática 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 perfeita para o nosso projeto.