# Analisador de Hierarquia de Palavras

Você deve criar uma aplicação de linha de comando que carrega uma árvore hierárquica de palavras, onde cada nível da árvore representa uma profundidade específica. A aplicação deve analisar uma frase fornecida pelo usuário, identificar a profundidade associada a uma palavra mencionada na frase, e então exibir os itens mais próximos dessa profundidade.

## Descrição da hierarquia

Você deve criar uma estrutura hierárquica de palavras onde cada palavra ou grupo de palavras pode ter subcategorias, semelhante a uma árvore de classificação. A estrutura deve ser representada em um arquivo json na pasta `dicts/.`

**Exemplo de Estrutura:**

- Animais
    - Mamíferos
        - Carnívoros
            - Felinos
                - Leões
                - Tigres
                - Jaguars
                - Leopardos
        - Herbívoros
            - Equídeos
                - Cavalos
                - Zebras
                - Asnos
        - Bovídeos
            - Bois
            - Búfalos
            - Antílopes
            - Cabras
        - Primatas
            - Gorilas
            - Chimpanzés
            - Orangotangos
    - Aves
        - 1.2.1. Rapinas
            - Águias
            - Falcões
            - Corujas
            - Milhafres
        - Pássaros
            - Canários
            - Papagaios

- Pardais
- Rouxinóis

# Desenvolvimento da CLI

## Comando para analisar uma frase

***syntax:*** java -jar cli.jar analyze –depth <n> –verbose (optional) “{phrase}”
Analisa a frase fornecida e exibe uma tabela com a contagem de palavras no nível de profundidade especificado.

Parâmetros:

- –depth <n>: Nível de profundidade da árvore para o qual exibir a contagem
- “{phrase}” texto a ser analisado
- –verbose: Caso seja informado deve exibir uma tabela no stdout com as seguintes métricas:
    - Tempo de carregamento dos parâmetros (ms)
    - Tempo de verificação da frase (ms)

# Exemplo de execução

Exemplo 1: Possui uma correspondência e está utilizando todos os parâmetros.
**comando**: java -jar cli.jar analyze --depth 2 "Eu amo papagaios" --verbose
***output***: Aves = 1 (Uma ave foi mencionada)

| | |
|---|---|
| Tempo de carregamento dos parâmetros | 50ms |
| Tempo de verificação da frase | 10ms |

Exemplo 2: Possui duas correspondências.
***comando***: java -jar cli.jar analyze --depth 3 "Eu vi gorilas e papagaios"
**output**: Pássaros = 1; Primatas = 1;

Exemplo 3: Não possui correspondência.
***comando***: java -jar cli.jar analyze --depth 5 "Eu tenho preferência por animais carnívoros"
**output**: 0;
Na frase não existe nenhum filho do nível 5 e nem o nível 5 possui os termos especificados.

# Requisitos técnicos

- **Requisitos Técnicos**
- **Linguagens:** Java 17+, JavaScript
- **Frameworks:** Spring Boot, Next.js
- **Bibliotecas Essenciais:** Google Cloud SDK (Java), Google Pub/Sub
- **Habilidades Necessárias*:***
    - Desenvolvimento de Rest APIs
    - Experiência com desenvolvimento de microserviços

    - Testes end-to-end e testes unitários com JUnit e Mockito
- **Diferenciais*:***
    - Experiência com GCP
    - Conhecimento em Java 8
    - Familiaridade com Payara e Jasper Reports
- ***Arquivos de dados***: Salve as estruturas que criar dentro da pasta dicts/
- ***Runtime***: O de sua preferência
- ***Testes:*** Você deve criar testes unitários. Todos devem estar rodando e, ao menos um, deve ter a análise de um texto de mais de 5000 caracteres.

## O que avaliamos?

- ***Git***: Como você organizou seu repositório, commits, branchs e clareza do README.md
- ***Arquitetura***: Como você organizou os componentes do seu código. Eles são extensíveis?
- ***Requisitos***: O projeto implementa tudo que o teste foi solicitado?
- ***Performance***: Quais técnicas de otimização você utilizou? Qual foi a eficiência do carregamento dos parâmetros?

## Desenvolvimento do frontend

Você deve desenvolver uma interface web usando Next.js e TypeScript que permita ao usuário criar uma hierarquia de palavras semelhante à descrita no teste original. A aplicação deve:

1. Permitir que o usuário adicione múltiplos níveis.
2. Exibir a hierarquia de palavras de forma visual.
3. Ter um botão "Salvar" que gera um arquivo JSON contendo a hierarquia criada.
4. Permitir o download do arquivo JSON gerado.

A estrutura do arquivo JSON gerado deve seguir o formato esperado que a CLI irá ler, conforme descrito no teste original.

## Requisitos Técnicos

***Tecnologia:*** TypeScript com Next.js
***Estrutura do Projeto:*** A hierarquia de palavras deve ser representada em estado dentro do React e ser convertida para JSON no momento de salvar.

## O que avaliamos?

- ***Git***: Como você organizou seu repositório, commits, branchs e clareza do README.md
- ***Arquitetura***: Como você organizou os componentes do seu código. Eles são extensíveis?

- ***UX/UI***: A interface é intuitiva e fácil de usar?
- ***Funcionalidades:*** Todas as funcionalidades descritas foram implementadas corretamente?
- ***Código:*** O código é limpo, organizado e documentado?