# Processamento de linguagem natural

Os requisitos foram lidos e separados em uma lista de tarefas. A lista de tarefas foi pensada para englobar todos os pontos necessário para a criação da solução e organizar a ordem que as tarefas seriam criadas.

A biblioteca de processamento de linguagem natural foi desenvolvida separando todas as responsabilidades em seus respectivos módulos e classes.

## Mapeamento das tarefas

* Criar o projeto e versionamento
* Implementar a leitura de arquivos
* Implementar o processamento do texto para remover stop-words
  + Implementar configuração das stop-words
  + Implementar para caso case sensitively
  + Implementar para caso case insensitively
* Implementar o processamento do texto por palavras isoladas
  + Implementar testes
  + Implementar criação do dicionário de palavras
  + Implementar remover stop-words
  + Implementar processamento dos vetores de palavras
* Implementar o processamento do texto por palavras em sequência
  + Implementar testes
  + Implementar criação do dicionário de palavras
  + Implementar remover stop-words
  + Implementar processamento dos vetores de palavras
* Implementar a exibição da saída

## 

## Hierarquia do projeto

A biblioteca foi criada visando a seguinte hierarquia de módulos e classes:

* \_\_main\_\_.py: entry point do projeto. Processa os argumentos passados pela linha de comando
* src: código do projeto
  + \_\_init\_\_.py: entry point do código do projeto
  + dictionaries: módulos dos PLN
    - utils: módulos de códigos diversos para resolver problemas pontuais na criação dos dicionários
    - isolated\_words.py: classe responsável pelo processamento baseado em um dicionário de palavras separadas
    - sequel\_words.py: classe responsável pelo processamento baseado em um dicionário de palavras em sequência
    - stop\_words.py
  + resources: recursos do projeto
  + file\_service.py: classe responsável por gerenciar os arquivos de recursos necessários para a execução das classes de PLN
* tests: código dos testes do projeto

## 

## Decisões do projeto:

### Stop Words

A eliminação das stop words é um recurso opcional da biblioteca. Podemos configurar as stop words de forma dinâmica permitindo alterar entre línguas mais facilmente melhorando a flexibilidade da aplicação. O padrão do arquivo de stop words foi definido consultando o modelo já utilizado pela Lucine que é uma biblioteca muito robusta e utilizada por vários motores de busca consolidados no mercado como o ElasticSearch.

Formato:

{

"wordSet": {

"initArgs": {"ignoreCase": true},

"managedList": []

}

}

### TDD

O desenvolvimento do projeto foi efetuado utilizando a metodologia do TDD, dessa forma foi muito mais simples definir as funções realmente necessárias para a biblioteca e nos casos de mudança detectar de forma rápida os erros cometidos.

## 

## Execução por linha de comando

Os caminhos exemplificados nesta seção consideram o caminho da pasta raiz do projeto.

O comando para a execução dos processadores de linguagem natural é:

|  |
| --- |
| python .\\_\_main\_\_.py .\src\resources\texto1.txt .\src\resources\texto2.txt |

Também podemos adicionar a configuração do arquivo de stop words com a flag -c:

|  |
| --- |
| python .\\_\_main\_\_.py -c .\src\resources\stop\_words.json .\src\resources\texto1.txt .\src\resources\texto2.txt |

Para executar a bateria de testes, foi utilizada a ferramenta pytest, basta executar o seguinte comando:

|  |
| --- |
| pytest .\tests\ |