repositorio do projeto: **https://github.com/fabriciosouza21/itexto**

Ferramentas utilizadas

Para realização da atividade de backend foi utilizado o framework spring boot 2.5.5, com o java 11, o gerenciado de dependência utilizado foi maven, as dependências do projeto foram :

- spring-boot-starter-data-jpa
- spring-boot-starter-test
- spring-boot-starter-web
- spring-boot-starter
- mysql-connector-java

Para realização dos testes unitários e de integração foram utilizadas as bibliotecas que já estão embarcadas no start do spring boot. Para realizar os testes da api foi utilizado o Postman, para o desenvolvimento foi utilizado a ide Spring Tool Suite 4.

O projeto foi Projetado utilizando a arquitetura em camadas simplificado na imagem abaixo:

Foram utilizados o padrão DTO para realizar a transferência de dados entre camadas de serviço e os controladores.

Quais dificuldades enfrentou em sua realização

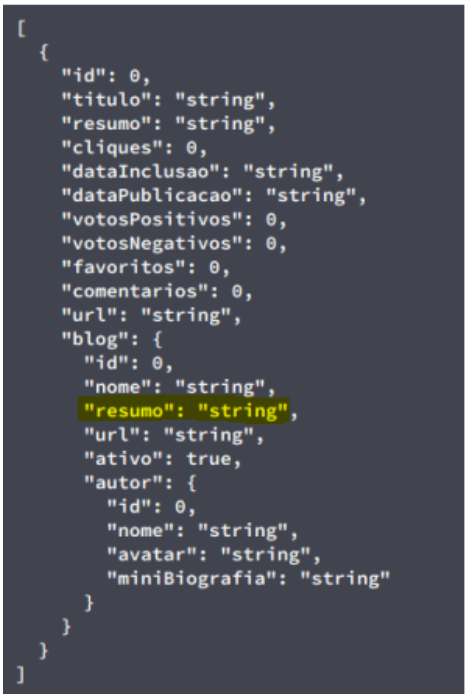
As dificuldades encontradas estão relacionadas ao spring data, sendo mais explícito ao criar query, tenho os fundamentos de banco de dados, porém com a falta de prática acabei tendo que pesquisa na documentação do spring data:
https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods
Foi necessário recorrer a documentação em dois pontos, primeiro ponto foi para gerar a query que fazia pesquisa textual por similaridade (like) posts cujos campos título ou resumo correspondiam ao valor fornecido. O segundo ponto foi para ordenar os dados obtidos da pesquisa por similaridade textual.

Alguma decisão de design que você tenha tomado que tenha gerado um resultado diferente do proposto pelo desafio técnico.

O acesso do banco estava restringindo o acesso a tabela de autor, por esse motivo foi adicionado somente o id do autor que está presente na tabela de Site, a estrutura de dados ficou semelhante a figura abaixo.

```
[
  {
    "id": 0,
    "titulo": "string",
    "resumo": "string",
    "cliques": 0,
    "dataInclusao": "string",
    "dataPublicacao": "string",
    "votosPositivos": 0,
    "votosNegativos": 0,
    "favoritos": 0,
    "comentarios": 0,
    "url": "string",
    "blog": {
      "id": 0,
      "nome": "string",
      "url": "string",
      "ativo": true,
      "autorID": 0
    }
  }
]
```

Além disso houve uma pequena mudança na chave de blog que representa os dados da tabela site, na estrutura enviando havia um atributo que não existia “resumo”.



```
[
  {
    "id": 0,
    "titulo": "string",
    "resumo": "string",
    "cliques": 0,
    "dataInclusao": "string",
    "dataPublicacao": "string",
    "votosPositivos": 0,
    "votosNegativos": 0,
    "favoritos": 0,
    "comentarios": 0,
    "url": "string",
    "blog": {
      "id": 0,
      "nome": "string",
      "resumo": "string",
      "url": "string",
      "ativo": true,
      "autor": {
        "id": 0,
        "nome": "string",
        "avatar": "string",
        "miniBiografia": "string"
      }
    }
  }
]
```

O que é necessário para que possamos compilar/executar seu projeto.

MÉTODO 1
Pré-requisitos backend-java-zip maven: Java 11 e Postman (ou a API Client que preferir)
Entrar na pasta abaixo
- cd backend

Executar o projeto
- ./mvnw spring-boot:run

MÉTODO 2
Pré-requisitos backend-java maven: Java 11 e Postman (ou a API Client que preferir)
Clonar repositório
- git clone https://github.com/fabriciosouza21/itexto

Entrar na pasta abaixo
- cd itexto-backend

Executar o projeto
- ./mvnw spring-boot:run

MÉTODO 3 (docker**)**
Pré-requisitos backend-java docker: Docker e Postman (ou a API Client que preferir)

#Executar docker
docker run -d -p 8080:8080 --rm fabricio21777/itextospring

MÉTODO 4 (Executar ambos os projetos(front e back) com docker compose
Clonar repositório
- git clone https://github.com/fabio21777/itexto

#Executar o projeto
- docker-compose up -d