Mini-Relatório

Buzzmonitor - Desafio - Estágio Backend

Giovanni Rosário - giovanni.rosario@hotmail.com

Screencast: youtu.be/rvRT6zz0ljo
Repositório da API: github.com/giovannirosario/BZMintern

Populando o banco ElasticSearch

A partir do dataset fornecido no link
https://gist.github.com/fabiosl/bfef293c110d3513b334f4134bff8ca2/raw/cb1ff2de0899f1c9a6c11c006ba24ff56e9f0571/dataset.json

Utilizei as ferramentas JQ stedolan.github.io/jq/ e JSON PY ES github.com/xros/jsonpyes para popular o banco.
No sublime, retirei a anotação do array e as virgulas que separavam cada documento.
No terminal
(cat data.json | jq -c .) > file
para desfazer o 'pretty' e
jsonpyes --data file --bulk http://localhost:9200 --import --index post_test --type post --check
para deixar no padrão da API bulk e upar no ES.

OBS: Após perceber que o mapping criado automaticamente havia mapeado o campo 'date' como "text", recriei o index mapeando 'date' para long.

Implementando a API

Foi utilizada a API Java High Level REST Client fornecida por www.elastic.co e documentação completa no link:
https://snapshots.elastic.co/javadoc/org/elasticsearch/client/elasticsearch-rest-high-level-client/7.0.0-alpha1-SNAPSHOT/index.html

Utilizando essa API, foi implementado uma classe DAO (Data Acess Object) que constrói a query a partir dos parametros fornecidos, executa uma search request no banco ElasticSearch e então retorna um objeto SearchHits, contendo todos os documentos do banco (posts, comentários e replies) com o nome de usuário e rede social fornecidos.
E uma classe Controller, onde foi implementado uma função mapeada para a rota GET /bzm_api/getPosts que recebe obrigatoriamente os QueryParams 'username' e 'social_net' e os parâmetros opcionais 'order' (ordenação cronológica do resultado, podendo ser ASC - Ascendente - ou DESC - Descendente -) e 'limit' (Número máximo de resultados).

Cumprimento dos requisitos

** A API deve retornar os dados de maneira cronológica e escalável;*

Os dados retornados são ordenados de forma cronológica a partir da variável ‘date’ e tratados individualmente, possibilitando escalabilidade.

** A API deve ser escrita em Java;*

A API foi escrita em Java, utilizando o framework Spring Boot e a ferramenta Gradle para controle de dependências e automação de compilação.

** A API deve retornar objetos JSON;*

A API retorna uma lista de objetos JSON;

** A API deve ser acessada via protocolo HTTP;*

A API é acessada através da rota http://localhost:8080/bzm_api

** O banco utilizado deve ser Elasticsearch*

Foi utilizado o banco ElasticSearch para armazenar o data set.