SPTRINT REVIEW

1. SPRINT PREVISTO/REALIZADO

- Armazenamento dos dados vindos da API Twitter: Foi previsto salvos os dados vindos da API Twitter em formato txt, mas como estes dados seriam consumidos em uma etapa posterior pela API criada, os dados não foram salvos em txt, mas sim em formato JSON que é um formato mais simples de ser lido pela API criada.
- Criação da API: Foi previsto utilizar a ferramenta Frapi para criar a API, mas optei por utilizar node.js e mongoDB para criar a API, que é a forma mais comum de criar API.

2. TECNOLOGIA UTILIZADAS

A seguir são descritas as tecnologias utilizadas em cada etapa (IDE's, linguagens, bibliotecas, etc.)

2.1. AQUISIÇÃO DE DADOS DO TWITTER

- Java: O projeto que faz a aquisição das informações do twitter e salva as listas em arquivos JSON, foi feito utilizando a linguagem Java.
- Twitter API: API que faz a comunicação com o twitter.
- Twitter4j: Biblioteca Java que chama a API do twitter, esta biblioteca tem todos os métodos (de alto nível) necessários para comunicação com o twitter.
- Eclipse Neon: foi utilizada a IDE Eclipse Neon para desenvolver o aplicativo que busca as #tags solicitadas e gera as listas propostas pelo problema.

2.2. CRIAÇÃO DA API

- Node.js: foi utilizada esta tecnologia para criar a APi
- MongoDB: o banco de dados foi criado utilizando esta tecnologia

2.3. CHAMADA DA API CRIADA

- Batch: A chamada da API criada é feita utilizando arquivo .bat

2.4. REPOSITÓRIO

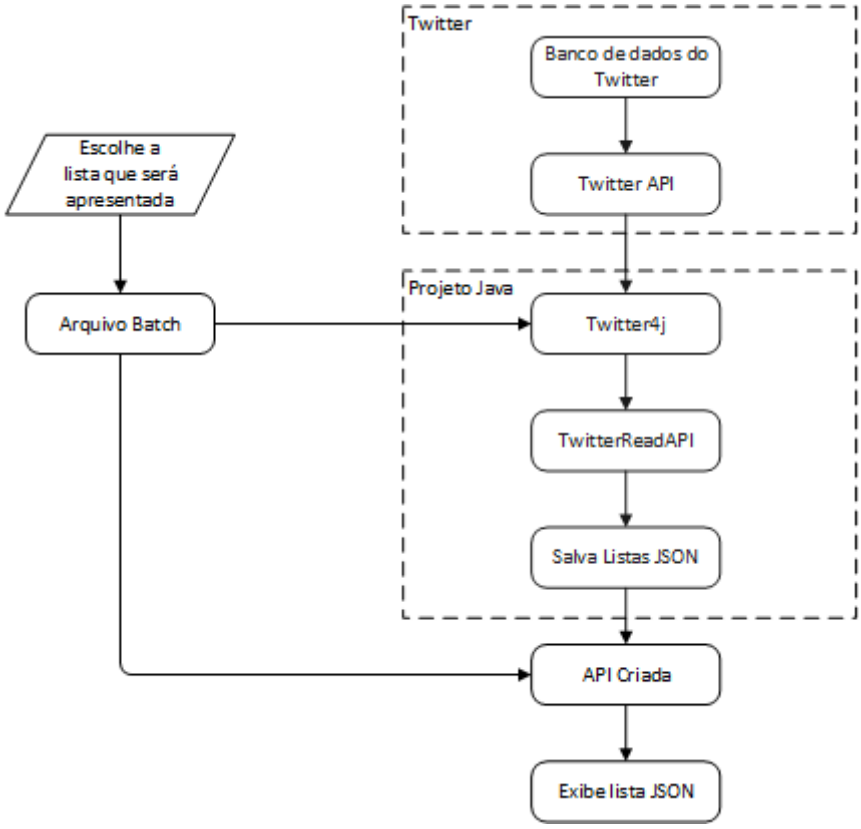
- Github: Todos os arquivos foram armazenados em um repositório github (https://guthub.com/andre1393/twitter)

- SourceTree: Toda interação com o repositório github (commit, pull, etc.) foi feita utilizando a IDE SourceTree.

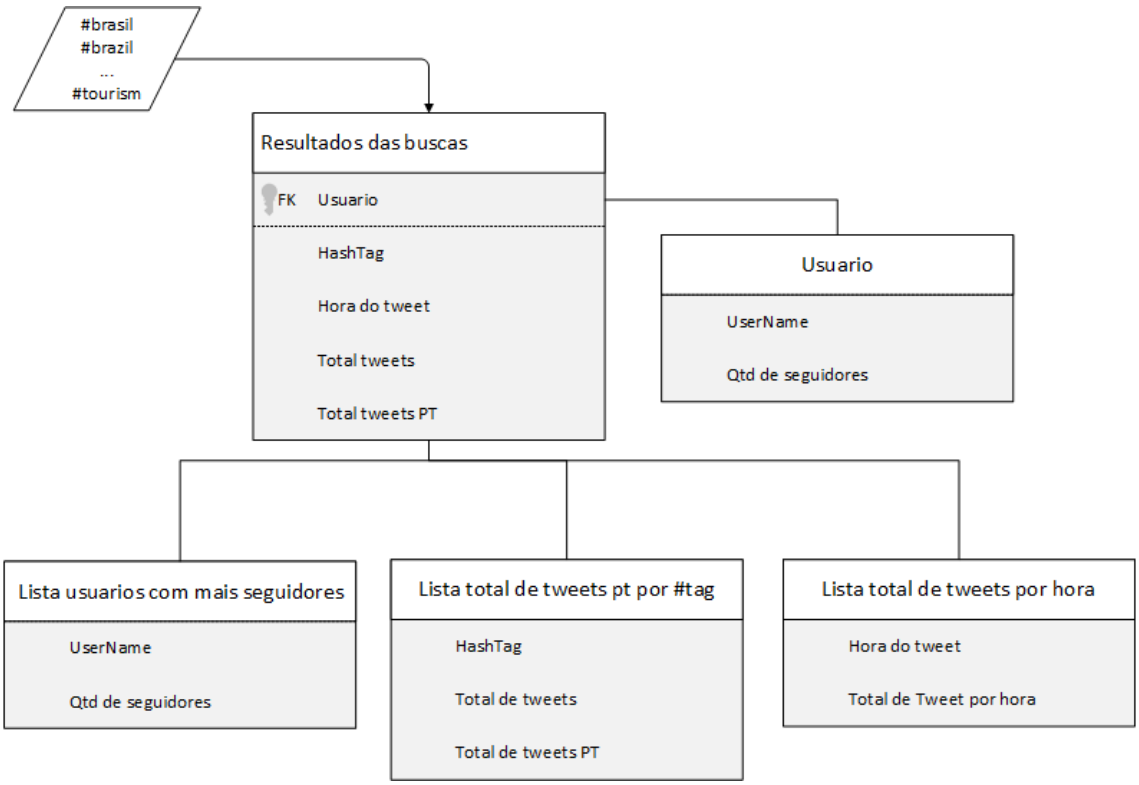
2.5. MONTAGEM DO AMBIENTE

- Docker: A montagem do Ambiente foi feita utilizando a ferramenta Docker e os arquivos imagem foram adicionados no github.

3. DIAGRAMA DA SOLUÇÃO



4. MODELAGEM DE DADOS



5. DIFICULDADES ENCONTRADAS

- Docker: Quando terminei de criar os arquivos .bat (última etapa) não tinha mais tempo de criar os arquivos Docker. Não sei se terá algum problema para rodar em outra máquina.
- Tive dificuldades em apresentar os resultados das listas de forma mais organizada, tive que apresentar em formato json, chamando a página localhost:3000...
- Fiquei na dúvida se a modelagem dos dados era para ser feita realmente dessa forma, tentei mostrar principalmente os dados que são gerados nas listas.

6. REFERÊNCIAS UTILIZADAS

Docker: https://docs.docker.com

Twitter API: https://dev.twitter.com/docs

Batch: http://www.instructables.com/id/Very-Basic-Batch-Tutorial/

Batch: http://batchscript.blogspot.com.br/2013/01/tutorial-batch.html

API: https://www.codementor.io/olatundegaruba/nodejs-restful-apis-in-10-minutes-q0sgsfhbd

API: https://www.youtube.com/watch?v=OG9iug9J6e8