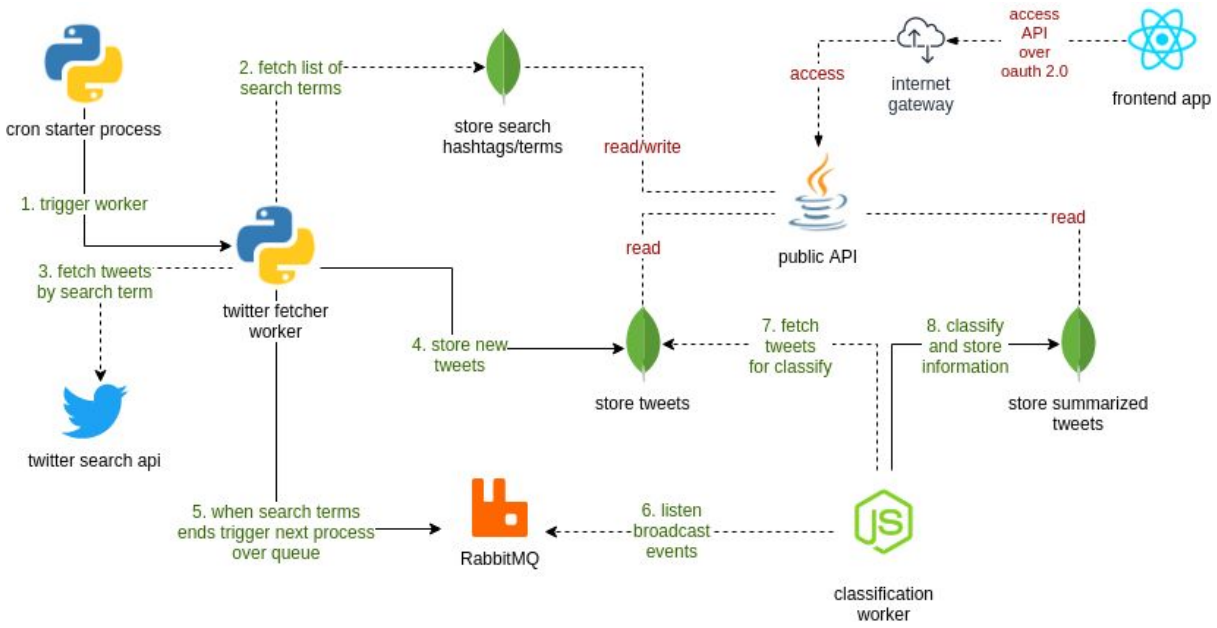
# Etapa case técnico

## Projeto Inicial

O diagrama a seguir foi elaborado pensando em como o processo será elaborado para resolver o desafio proposto pelo teste.




A ideia é utilizar um banco de dados não relacional para guardar as informações pois elas, aparentemente, não precisam de relacionamentos fortes. Cada tweet pode ser abordado como um documento distinto e guardado no banco.

O banco de dados escolhido foi o MongoDB por ser um banco fácil de manipular, ter uma resposta muito rápida e possuir a funcionalidade *map-reduce* que pode ser explorado para fazer a devida classificação e agregação.

MongoDB é um banco de dados que combina muito bem com nodejs pelo fato de os dois trabalharem com linguagem JavaScript tanto para fazer as *queries* como para realizar ações dentro do banco de dados. Por isso, foi selecionado nodejs como ferramenta de sumarização de dados.

Para a realização do resgate e armazenamento dos tweets foi optado como linguagem Python por ser uma linguagem com uma sintaxe simples de fazer manutenção e com suporte ótimo assim como uma comunidade de desenvolvedores madura. A linguagem possui ótimas bibliotecas de apoio como utilitário de *cron* para executar comandos de tempos em tempos e cliente de twitter muito simples.

Para realizar a comunicação entre os dois serviços (python e nodejs) optou-se por utilizar um sistema de mensageria para que um sistema não interfira no processo do outro, tornando assim os dois sistemas independentes e facilitando a manutenção de ambos.

A API que disponibilizará os dados será escrita utilizando *spring* na plataforma Java por ser um *framework* que facilita o desenvolvimento de aplicações de microsserviços e ser mais fácil de trabalhar na atualidade, além de ser mais utilizado no mercado atual, o que facilita na hora de contratação de funcionários para dar futuras manutenções em seu código.

O API gateway ainda não foi definido, mas talvez será utilizado Kong por se tratar de uma ferramenta pronta que possui diversos plugins para facilitar o desenvolvimento da estrutura, ser um utilitário *opensource* além de poder ser instalado no servidor, o que permite que seja agnóstico quanto ao servidor onde o sistema será implementado.

Para entregar a aplicação de *frontend* será utilizado React por estar em maior evidência no mercado atual e possuir um bom suporte e bibliotecas que facilitam o desenvolvimento e manutenção do mesmo. Pode-se utilizar ainda, numa possível próxima versão, o utilitário NextJs para realizar implementações de SSR ─ Server Side Rendering.

Para realizar todos os processos será utilizado *docker* por ter uma natureza de isolamento de dependências de a nível de software. Desta forma o ambiente fica igual tanto nas máquinas dos desenvolvedores quanto nos servidores finais. Ele também possui um isolamento de variáveis de ambientes e chaves criptografadas deixando as aplicações mais seguras. A conteinerização provida pelo *docker* também possui facilidade de escalonamento através de plataformas como *docker swarm*, *kubernetes*, *rancher*, *AWS ECS* entre outras.

Para prover releases semânticos será utilizado stepup, assim as versões do projeto serão gerenciadas sem que o desenvolvedor tenha que ficar pensando no gerenciamento de suas versões e sua lista de alterações ao longo do projeto.

# Referências


- Documentação do twitter search
  https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html
- Python cron tab
  https://pypi.org/project/python-crontab/
- Python twitter client
  https://github.com/bear/python-twitter

- PyMongo
  https://pypi.org/project/pymongo/
- MongoDB map-reduce
  https://docs.mongodb.com/manual/core/map-reduce/
- MondbDB incremental map-reduce
  https://docs.mongodb.com/manual/tutorial/perform-incremental-map-reduce/
- Start spring
  https://start.spring.io/
- SpringData MongoDB
  https://spring.io/projects/spring-data-mongodb
- APIGee
  https://cloud.google.com/apigee/
- Axway
  https://www.axway.com/en
- Docker with multiprocess
  https://docs.docker.com/config/containers/multi-service_container/
- Docker and supervisord
  https://www.dedoimedo.com/computers/docker-supervisord.html
- Supervisord config samples
  https://gist.github.com/robcowie/3833088
- Supervisord documentation
  http://supervisord.org/index.html
- Kong
  https://konghq.com/
- Kong Oauth2.0 plugin
  https://docs.konghq.com/hub/kong-inc/oauth2/
- StepUp
  https://rubygems.org/gems/step-up/
  https://github.com/kawamanza/step-up