Análise de sentimentos em *tweets*: baseado em stream de dados via Kafka

Enrico V. S. Robazza[1], **Gustavo T. Mastrobuono** [1], **Samuel L. Godoy**[1], **Vinícius M. Garcia**[1]

[1] Instituto de Ciências Matemáticas e de Computação (ICMC) – Universidade de São Paulo (USP)
São Carlos, SP

Repositório: https://gitlab.com/icmc-ssc0158-2021/2021/gcloud10

Resumo. *O Twitter é um site de mensagens curtas no qual os usuários podem postar atualizações (tweets) para seus seguidores. Tornou-se um imenso conjunto de dados dos chamados sentimentos. Neste artigo, apresentamos uma abordagem simples que classifica automaticamente o sentimento de tweets. Os tweets são classificados como positivos, negativos ou neutros em relação a um termo de consulta. Essa abordagem é útil para consumidores que desejam usar a análise de sentimento para buscar por produtos, para empresas que visam monitorar o sentimento público de suas marcas, e para muitas outras aplicações.*

1. Introdução

O twitter é um serviço mundialmente popular, no qual os usuários podem expressar suas opiniões, sentimentos e impressões através de pequenos trechos de texto, denominados *tweets*, com tamanho máximo de 140 caracteres. Os números do Twitter são impressionantes, sendo um dos maiores banco de dados de conteúdos gerados por usuários, contando com cerca de 200 milhões de usuários, postando mais de 400 milhões de *tweets* por dia [Ritter et al., 2011]. Por conterem expressões de usuários reais, o conteúdo presente no Twitter pode ser utilizado como um ”termômetro”para medir tendências do mundo real, como a percepção que as pessoas têm de uma determinada marca ou produto [Jansen et al., 2009], alertas sobre catástrofes e situações que ameaçam a segurança das pessoas, prever a preferência política [Calais Guerra et al., 2011] e outras ilimitadas possibilidades. Ferramentas automatizadas para extrair informações desta base de dados e transformar os milhões de *tweets* em insights que mostrem tendências podem ser grandes aliadas no processo decisório. Desta forma, este trabalho propõe uma ferramenta que lida com um *stream* de tweets e faz uma análise de sentimentos em cada um para extrair o teor do texto: se é algo positivo ou negativo sobre o assunto.

A análise de sentimentos, no contexto de ciências computacionais, trata do conjunto de técnicas e ferramentas para extrair de algum material de origem, geralmente textuais, quais são as opiniões e emoções associadas ao material. Em aplicações como extração de sentimentos de *tweets*, gera-se a preocupação com a informalidade da linguagem, contendo uma quantidade maior do que o normal de gírias e erros ortográficos. Além disso, os conteúdos postados no Twitter não são classificados de acordo com uma temática, mas sim, tratam de uma grande variedade de tópicos, de forma a dificultar a escolha de um corpus textual. A análise de sentimentos neste caso, comporta-se como um problema de classificação e pode ser tratada de duas formas: a primeira delas é analisar a presença de alguma emoção e classificar em positivo, negativo ou neutro. A outra opção

é considerar as seis emoções universais [Ekman, 1982]: raiva, medo, felicidade, tristeza, desgosto e surpresa.

Este trabalho tem um foco muito maior na estrutura da aplicação em nuvem, que tratará o stream de dados (produtor e consumidor) do que no modelo de classificação em si. Deste forma, utilizaremos um modelo simples de Regressão para demonstrar o funcionamento da plataforma, considerando a abordagem de positivo e negativo. Vale também salientar algumas questões que levam a baixas performances de análise - que vão além da escolha do modelo - por desafios intrínsecos à natureza dos *tweets* [Hassan et al., 2013]: (i) os tweets são em sua maioria neutros, não contendo expressões positivas ou negativas; (ii) os tweets são muito curtos e, em muitas das vezes, com informações insuficientes para uma expressão de sentimento; (iii) a representação linguística do tweet diferem de outros corpus textuais, gerando maior complexidade nas etapas de engenharia de características.

Comentadas estas questões sobre a análise de sentimentos em si, está na hora de tratar da origem dos dados. Este trabalho utiliza o conceito de *streaming* de eventos, que é a base tecnológico para um fluxo de dados vivo no mundo digital, uma das características mais decisivas para os negócios inteligentes [Zeng, 2019]. É através da datificação do mundo físico, com fluxos de dados contínuos e bem estabelecidos que ”dados vivos”podem ser coordenados e otimizados para o processo decisório. De forma técnica, o *event streaming*, ou fluxo de eventos, é a captura de dados em tempo real, em que os eventos podem vir de banco de dados, sensores, serviços em nuvem ou outras aplicações de software, como é o caso do Twitter. Após capturados, estes fluxos de dados são armazenados em uma forma não-volátil para que posteriormente seja possível recuperá-los e reutilizá-los. Além de poderem ser manipulados, processados e direcionarem a reação de outros sistemas, os fluxos de eventos podem ser direcionados para outras tecnologias conforme for necessário.

Este trabalho utiliza uma plataforma para lidar com fluxos de eventos denominada Apache Kafka [Kafka, 2021], uma plataforma que três capacidades essenciais ao lidar com este fluxo contínuo de dados:

- Ele é capaz de escrever (publicas) e ler (consumir) fluxo de eventos
- É possível armazenar estes fluxos de forma durável e confiável pelo tempo que for preciso
- Processar os eventos conforme eles ocorrem, ou de forma retrospectiva

sendo que todas estas funcionalidades são disponibilizadas de uma forma distribuída, altamente escalável, elástica, resistente a falhas e de forma segura.

O Kafka é um sistema distribuído que opera em uma lógica de cliente-servidor, os quais se comunicam através de um protocolo de rede TCP de alta performance. Os servidores são executados como um cluster, que podem abranger uma vasta região da *cloud* e repositórios de dados. Alguns destes servidores constituem a camada de armazenamento, os chamados *brokers*. Outros fazem o trabalho de importar e exportar os fluxos de eventos, conectando o Kafka com algum outro sistema existente. Já os clientes, permitem que o usuário crie aplicações distribuídas que lêem, escrevem e processam esses fluxos de eventos de forma extremamente eficiente e robusta. O *client-side* está disponível através da biblioteca Kafka Streams para Go, Python (linguagem escolhida neste trabalho), C/C++, APIs REST e muitas outras linguagens.

2. Arquitetura Geral

Eventos são entidades que retratam que algo aconteceu no domínio monitorado, podemos encará-los como uma mensagem que será enviada. Quando escrevemos ou lemos dados do Kafka, fazemos isso na forma de eventos, o qual contém uma chave, seu valor, um timestamp e também pode conter metadados sobre o evento. Os **produtores** são as aplicações que vão publicar (escrever) os eventos no Kafka, em nosso contexto, é o servidor que escuta a API do Twitter e escreve o *tweet* como uma mensagem no Kafka. Já os **consumidores** são as aplicações que se inscrevem para poderem ler e processar estes eventos (mensagens). Ao trabalhar com Kafka, os produtores e consumidores são completamente desacoplados, de forma que podem ser desenvolvidos independentemente.

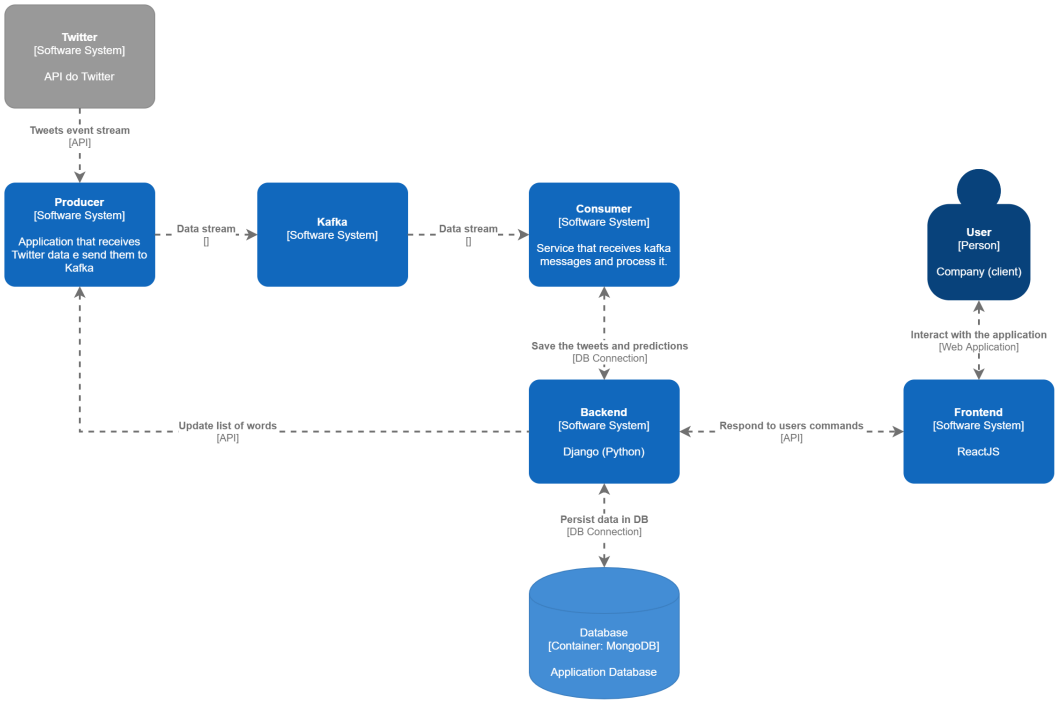

Figura 1. Arquitetura do sistema mostrando os contêineres macro.

Na Figura 1, podemos visualizar o bloco produtor recebendo o fluxo de dados da API do Twitter e encaminhando para o Apache Kafka. Por sua vez, o consumidor recebe o fluxo destes dados e faz os encaminhamentos desejados.

Vale comentar que os eventos são organizados e armazenados no formato de tópicos, como visto na Figura 2, que podem ser visualizados como se fossem uma pasta em um sistema de arquivos e os eventos são os arquivos contidos nesta pasta. Os tópicos podem ter vários produtores e consumidores acessando-o simultaneamente. E uma diferença de outros sistemas tradicionais baseados em mensagens é que os eventos não são apagados após a leitura, mas sim, podem ser lidos quantas vezes for necessário. Os tópicos são particionados e guardados em diferentes *brokers* - camadas de armazenamento - garantindo que tal distribuição no armazenamento confira ao Kafka seu poder de escalabilidade, pois permite que que as aplicações possam ler e escrever dados de diferentes brokers ao mesmo tempo.

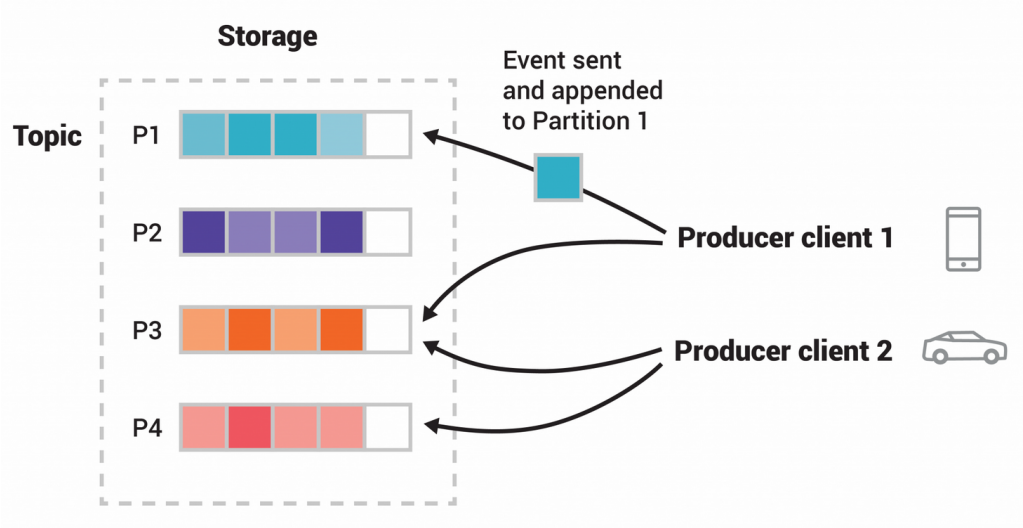

Figura 2. Estruturação do armazenamento de eventos dentro de tópicos. Imagem retirada do site oficial do Apache Kafka [Kafka, 2021].

3. Producer

A função do produtor é escutar a API do Twitter para receber os tweets desejados. Para isto, ele conta com dois blocos principais: o PrintingListener e o TermChecker, como mostrado na Figura 3. O PrintingListener é responsável por receber os tweets que contenham uma ou mais palavras presentes na track de palavras desejadas. Já o TermChecker é executado a cada 15 segundos e tem como objetivo buscar, via API, a lista de palavras atualizadas no backend da aplicação.

Para que a aplicação pudesse funcionar de acordo com os casos de uso que mapeamos, forkamos a biblioteca **twittermonitor** para permitir a escuta de lista de palavras, ao invés, de palavras individuais.

4. Consumer

O consumer é o serviço que está inscrito no Kafka para ler e processar o stream de eventos nos tópicos que o consumer está interessado. Ele possui três grandes blocos: o KafKaConsumer, o Translator e o Predictor.

O KafkaConsumer basicamente lê o fluxo de eventos do tópico desejado no Kafka e envia cada evento deste fluxo para o Translator. Este, por sua vez, traduz cada uma dos tweets para o Português brasileiro. Ele é necessário uma vez que o modelo foi treinado utilizando-se uma base de dados de comentários sobre séries e filmes do IMDb [Ali et al., 2019] que estavam neste idioma. Assim, para que a classificação fosse feita corretamente, cada um dos tweets precisa passar pela etapa de traduação para o idioma alvo.

Seguindo o fluxo que pode ser visualizado na Figura 4, o próximo passo do fluxo de eventos é o Predictor. Uma vez que os tweets já estão traduzidos para o idioma alvo, eles podem ser direcionados para a classificação. O Predictor faz uma verificação para ver se já existe algum modelo treinado disponível (um .joblib), caso ainda não exista, ele faz a etapa de treinamento com a base do IMDb. Se já existir, ele utiliza este modelo para

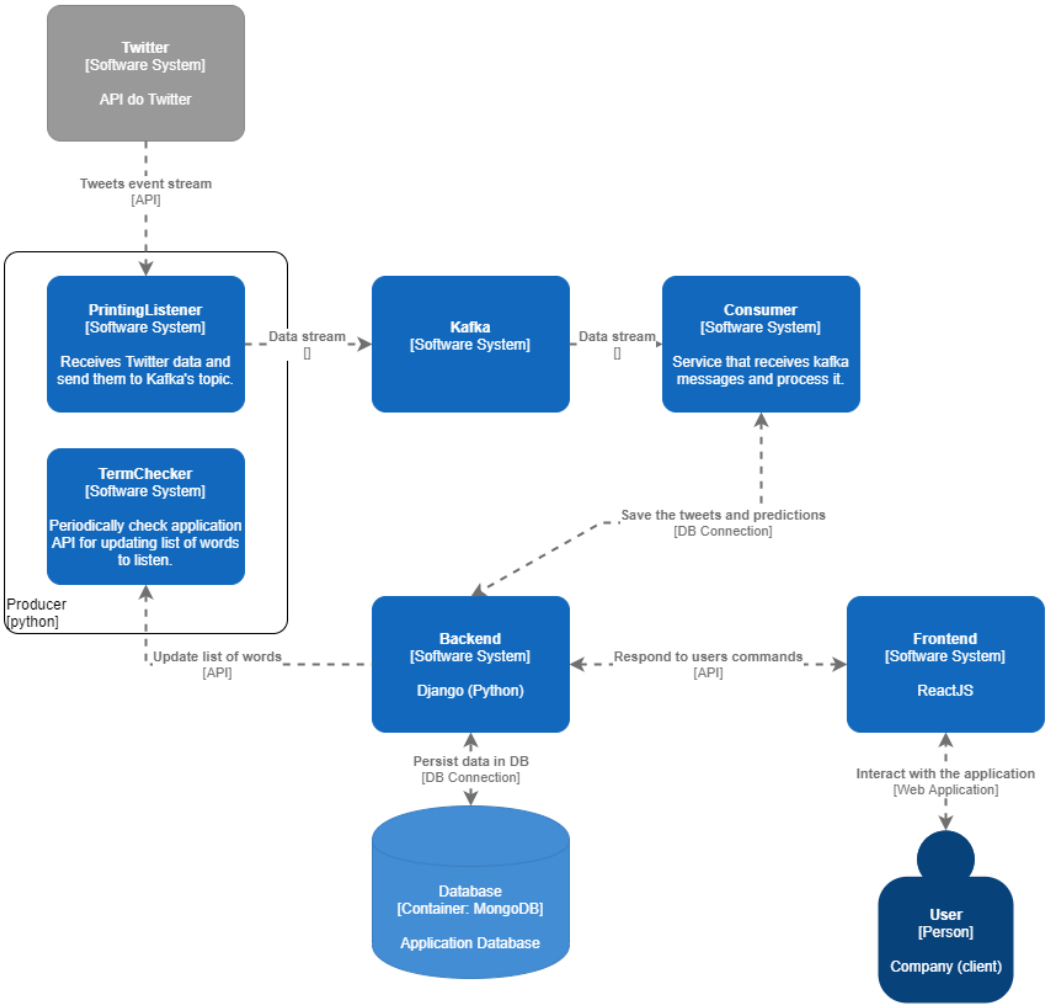

Figura 3. Visualização dois dois blocos que compõe o producer.

fazer o predict em cima dos dados que já chegaram. Este predict vai retornar se foi um tweet positivo ou negativo.

Tanto o tweet quanto a classificação realizada sobre o mesmo serão enviados para o backend da aplicação, via API, que persistirá estas informações no banco de dados.

5. Web

O acesso do usuário é feito através de uma aplicação Web implementada em Django (back-end) e React (front-end), com suporte de um banco de dados MongoDB. Esta estruturação traz algumas facilidades na hora de estruturar a lógica de autenticação e perfilamento, de forma que toda a parte de cadastro, alteração de senhas, login e autenticação foram implementadas.

Através da aplicação React, o usuário consegue interagir com o sistema que faz a análise de tweets através de uma API REST, pela seguinte arquitetura:

- Ao fazer um GET (GroupView), o front-end obtém a lista de Grupos que foram previamente cadastrados. Estes grupos são mostrados na tela de Grupos para que o usuário possa selecionar um grupo para visualizar.

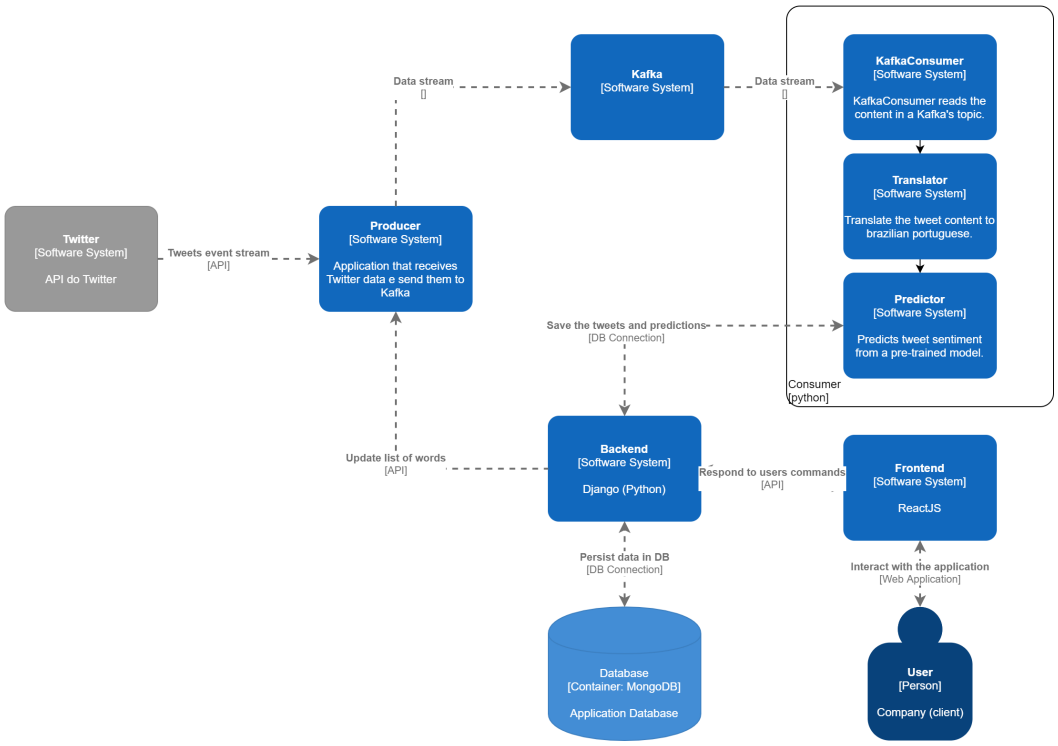

Figura 4. Visualização arquitetural do funcionamento do consumer.

- Ao fazer um POST (GroupView), o usuário pode criar novos grupos de palavras. Ao receber a requisição pela API, o back-end cria um novo grupo no banco de dados.
- Ao fazer um POST, o usuário envia para a API (TrackView) uma palavra que tem interesse que seja rastreada. Ao receber a requisição pela API, o back-end persiste essa informação no banco de dados.
- Através da mesma API (TrackView), o producer faz um polling obtendo os valores que devem ser rastreados no Twitter.
- Através da API (TweetView), o consumer (Predict Service) envia, através de um POST, o resultado da avaliação feita sobre um determinado tweet. O back-end então armazena esta informação no banco de dados.
- O front-end pode acessar, por meio de um GET (TweetView), os tweets e a avaliação feita sobre eles pelo Predict Service. Ao receber a requisição pela API, o back-end busca os tweets atualmente armazenados no banco de dados e retorna o valor encontrado.

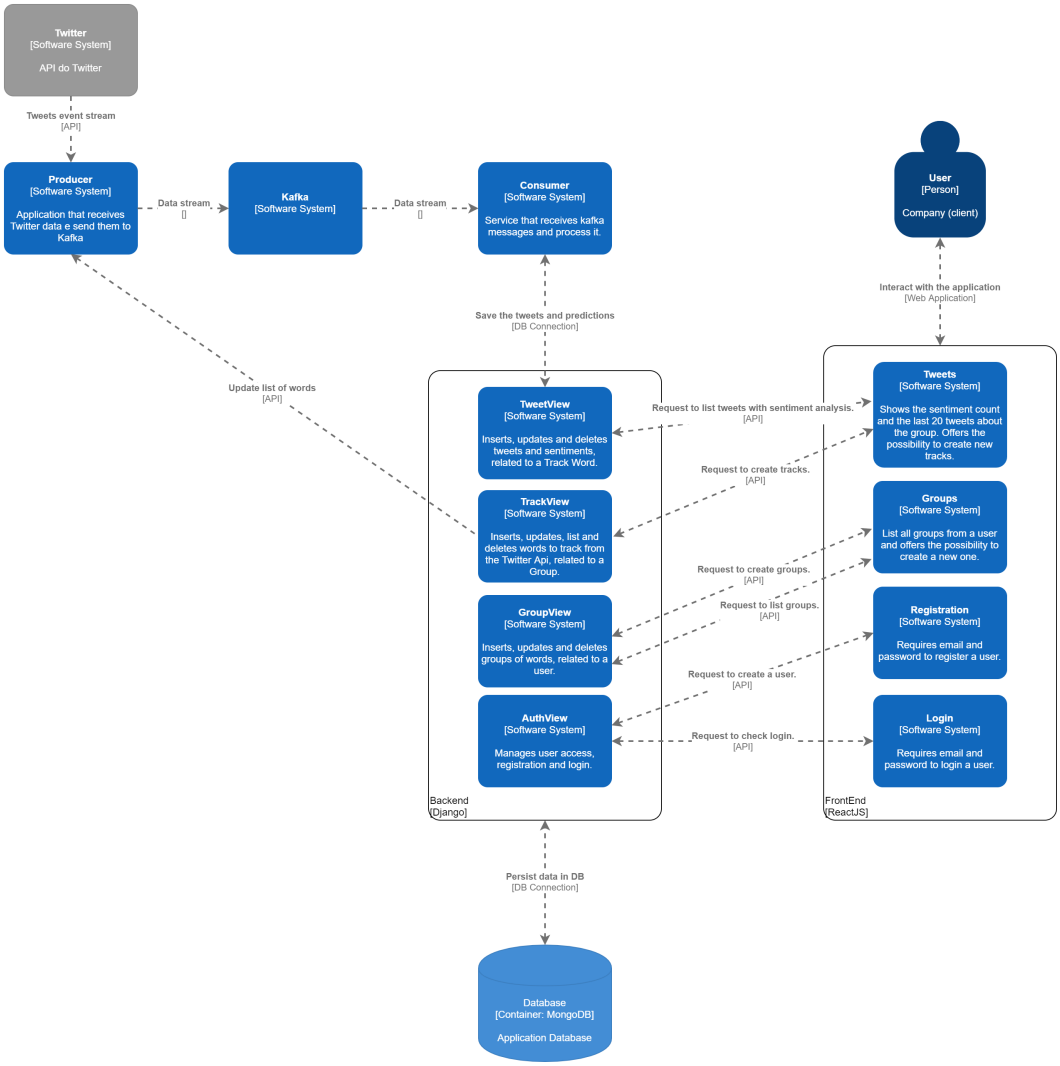

Figura 5. Visualização arquitetural do funcionamento da aplicação Web

Referências

Ali, N. M., Abd El Hamid, M. M., and Youssif, A. (2019). Sentiment analysis for movies reviews dataset using deep learning models. *International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol*, 9.

Calais Guerra, P. H., Veloso, A., Meira Jr, W., and Almeida, V. (2011). From bias to opinion: a transfer-learning approach to real-time sentiment analysis. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158.

Ekman, P. (1982). Emotion in the human face, vol. second.

Hassan, A., Abbasi, A., and Zeng, D. (2013). Twitter sentiment analysis: A bootstrap ensemble framework. In *2013 international conference on social computing*, pages 357–364. IEEE.

Jansen, B. J., Zhang, M., Sobel, K., and Chowdury, A. (2009). Twitter power: Tweets as electronic word of mouth. *Journal of the American society for information science*

and technology, 60(11):2169–2188.

Kafka, A. (2021). Apache kafka. *A distributed streaming platform (acessado em Julho de 2021). https://kafka.apache.org/intro*.

Ritter, A., Clark, S., Etzioni, O., et al. (2011). Named entity recognition in tweets: an experimental study. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 1524–1534.

Zeng, M. (2019). *Alibaba: Estratégia de sucesso*. M. Books.