INSTITUTO FEDERAL DE SANTA CATARINA

SUYAN MORIEL VIESE MOURA

Uso de chatbot para facilitar a checagem de fake news em redes sociais

São José - SC

Agosto/2024

USO DE CHATBOT PARA FACILITAR A CHECAGEM DE FAKE NEWS EM REDES SOCIAIS

Projeto de trabalho de conclusão de curso apresentado à Coordenadoria do Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina para a aprovação do tema perante banca na disciplina de TCC2.

Orientador: Prof. Cleber Jorge Amaral, Dr.

São José - SC

Agosto/2024

Sempre que te perguntarem se podes fazer um trabalho,
respondas que sim e te ponhas em seguida a aprender como se faz.
T. Roosevelt

RESUMO

As redes sociais surgiram como poderosos meios de disseminação de informações. No entanto, esse cenário também desencadeou o desafio crescente das fake news. O problema em questão reside no fato de que, embora existam processos de fact-checking, as publicações em redes sociais não são necessariamente submetidas a esse processo. Nesse contexto, a abordagem adotada neste projeto foi desenvolver um bot integrado ao Twitter[1], capaz de indicar fontes confiáveis diretamente na plataforma, proporcionando aos usuários acesso conveniente a informações verificadas. A conclusão dos testes revelou não apenas a eficácia do sistema no ambiente controlado, mas também a complexidade inerente ao processo de mitigação de fake news. Diante disso, reiteramos que combater a desinformação é um esforço conjunto e o projeto oferece uma contribuição valiosa ao fornecer informações confiáveis de maneira conveniente aos usuários do Twitter.

Palavras-chave: Bot; Fake News; Redes sociais.

[1] No ano de 2023 o Twitter mudou o nome para X (DAVIS; WARREN, 2023).

ABSTRACT

Social media has emerged as a powerful means of information dissemination. However, this has also brought forth the growing challenge of fake news. The underlying issue is that, despite the existence of fact-checking processes, most posts aren't submitted to those processes. In this context, the approach taken in this project was to develop a bot integrated into Twitter, capable of indicating reliable sources directly on the platform and providing users with convenient access to verified information. The conclusion of the tests revealed not only the system's effectiveness in a controlled environment but also the inherent complexity in mitigating fake news. In light of this, we emphasize that combating misinformation is a collective effort, and the project offers a valuable contribution by providing reliable information conveniently to Twitter users.

Keywords: Bot. Fake News. Social network.

LISTA DE ILUSTRAÇÕES

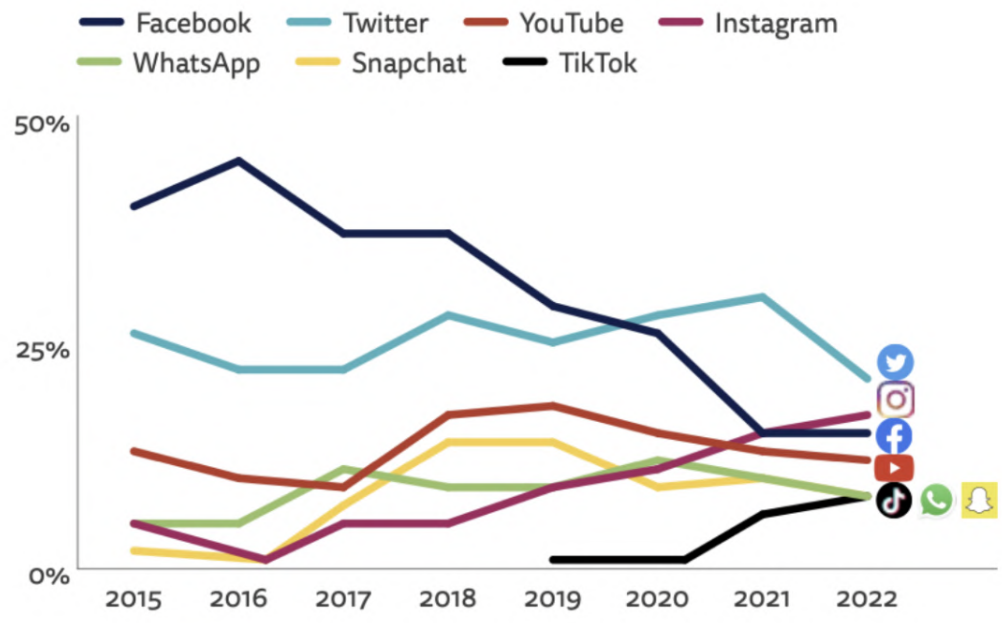
SUMÁRIO

1 INTRODUÇÃO

Cada vez mais as redes sociais estão se tornando parte de nossas vidas, não só como meio de interação social, mas também como meio de obter informações.

No Reino Unido, por exemplo, 39% dos jovens com idades entre 18 e 24 anos optam por obter informações por meio de redes sociais (NEWMAN et al., 2022). Entre as redes sociais, o Twitter destaca-se como a fonte preferida de informações jornalísticas predileta pelos jovens britânicos, como ilustrado na Figura 1.

Figura 1 – A proporção de pessoas com idades entre 18 e 24 anos que utilizaram cada rede social para se informar na última semana no Reino Unido.



Fonte: Newman et al. (2022)

Nos últimos anos, intensificou-se a discussão referente às *fake news*, que são notícias intencionalmente falsas e passíveis de verificação (LAZER et al., 2018). Em redes sociais, como o Twitter, qualquer pessoa pode criar uma ou mais contas para publicar mensagens, nomeadas como tweets, que podem conter qualquer tipo de conteúdo e difundir qualquer tipo de informação. O Twitter também permite que um usuário compartilhe a informação publicada por outro usuário (o retweet), o que facilita o difusão de notícias falsas. No entanto, para descobrir a veracidade de uma informação no Twitter é relativamente complexo e pode demandar horas de pesquisa em fontes confiáveis.

Hoje, há empresas e ferramentas que facilitam a verificação de fatos, que são as agências verificadoras de fatos. Essas agências têm como objetivo verificar a veracidade das

notícias que são divulgadas em massa nas redes sociais, através do processo de *fact-checking.*

Atualmente, o Twitter permite o uso de *bots*, que são programas de computador que se comportam como usuários convencionais (humanos) na rede social. Esses *bots* podem realizar as mesmas operações que usuários humanos, como ler, publicar e responder os tweets nos quais foram mencionados.

Nesse contexto de disseminação de *fake news* em redes sociais usadas como fontes de informações jornalísticas, juntamente com os desafios na obtenção de informações confiáveis, este trabalho teve como objetivo criar um *bot* integrado ao Twitter para facilitar a obtenção de informações confiáveis por parte dos usuários. Esta integração funciona da seguinte forma: ao mencionar o *bot* em um tweet, o usuário indica seu interesse em verificar a veracidade de um conteúdo ou obter informações adicionais. O *bot*, por sua vez, é responsável por realizar as pesquisas fora da rede social, em ambientes externos como motores indexadores de verificações de fato ou Interface de Programação de Aplicativos (API)s especializadas em verificação de fatos, trazendo os links de artigos confiáveis[1] que tratam o tema do tweet para dentro da rede social.

O resultado esperado com essa estratégia é combinar informações confiáveis de fontes especializadas em verificação de fatos fora do Twitter com a comodidade de não precisar sair da rede social para obtê-las. O *bot* realizará pesquisas externas e entregará resultados diretamente aos usuários por meio de respostas nos tweets mencionados. Isso simplificará o acesso a informações confiáveis, ajudará a identificar notícias falsas e oferecerá sugestões de leitura e verificações de fatos de forma rápida e conveniente. Assim, os usuários terão uma experiência segura com a obtenção de informações no Twitter. Além disso, a resposta realizada pelo bot fica acessível aos demais usuários da rede social, auxiliando na divulgação de artigos de checagem de fake news.

Sabe-se que identificar *fake news* pode ser uma tarefa complexa. Sabendo desse grau de complexidade, a proposta desta ferramenta não é acusar a notícia como *fake news*, mas indicar artigos de sites de checagem, de tal forma que fique fácil ao usuário encontrar fontes sobre certa informação divulgada na rede social.

1.1 Objetivo Geral

Desenvolver um bot integrado ao Twitter com o propósito de auxiliar o usuário na obtenção de informações verificadas e confiáveis.

[1] Neste caso, “confiáveis” são artigos oriundos de agencias verificadoras de fato

1.2 Objetivos Específicos

- Implementar um bot para extrair informações de um tweet marcado pelo usuário;
- Implementar um serviço (software) que a partir das informações obtidas pelo bot, seja capaz de obter fontes no Google Fact-Checking Tools (GFCT);
- Projetar uma arquitetura que facilite a integração com múltiplas redes sociais e diferentes serviços de verificação de fatos.
- Realizar testes para validar a eficácia da solução desenvolvida;
- Contribuir para a promoção da veracidade das informações propagadas na plataforma, trazendo maior confiança e credibilidade para seus usuários.

1.3 Organização do texto

No capitulo 2 será apresentada a fundamentação teórica, onde os conceitos fundamentais para o entendimento deste projeto serão apresentados. No capitulo 3 será apresentado o desenvolvimento do projeto, será discutida a arquitetura projetada, os serviços desenvolvidos, as classes criadas, a API implementada e os contêineres criados. Por fim, no quarto capítulo serão apresentadas as conclusões e indicações para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os tópicos essenciais para o entendimento deste projeto. Será explicado o que são as redes sociais, tal como quem as utiliza e para que são utilizadas. Um pouco do Twitter será discutido neste capítulo, tal como também será discutido como as redes sociais se relacionam com as fake news. A seguir serão abordados temas mais técnicos úteis para o entendimento do projeto como um todo.

2.1 Redes Sociais

Uma rede social é uma plataforma web que permite aos indivíduos (ou às vezes grupos e até mesmo empresas) criarem perfis públicos ou semi-públicos dentro de um sistema delimitado. Permitindo assim a interação com outros usuários com os quais compartilham algum tipo de conexão (BOYD; ELLISON, 2007).

As redes sociais desempenham um papel significativo na vida contemporânea. São plataformas digitais que permitem que pessoas e organizações se conectem, compartilhem informações e interajam online. Elas facilitam a comunicação, a interação social e o compartilhamento de mídia, como fotos e vídeos. Exemplos populares de redes sociais incluem Facebook, Twitter, Instagram e LinkedIn.

2.1.1 Seus usuários

Redes sociais são acessíveis a uma ampla gama de usuários. Qualquer pessoa com acesso à Internet pode criar uma conta e participar de redes sociais, incluindo indivíduos de quase todas as idades, empresas, organizações sem fins lucrativos e até mesmo figuras públicas. As redes sociais oferecem a flexibilidade de criar perfis pessoais ou comerciais, permitindo que diferentes tipos de usuários interajam.[1]

2.1.2 Seus propósitos

As redes sociais possuem uma ampla variedade de usos, desde manter contato com familiares e amigos até promover produtos e serviços. Uma das aplicações mais relevantes é o uso das redes sociais para informações jornalísticas. Elas oferecem um ambiente dinâmico e interativo para a disseminação de notícias em tempo real (BOYD; ELLISON, 2007).

[1] Cada rede social possui seus próprios termos de uso, e um usuário só pode criar uma conta em uma rede social se ele estiver de acordo com as regras internas da mesma.

Redes sociais desempenham um papel importante na disseminação de notícias e informações. Elas permitem que jornalistas compartilhem artigos e atualizações com seu público alcançando um público mais amplo do que os métodos tradicionais de distribuição. Além disso, as redes sociais permitem que os usuários compartilhem notícias, comentem sobre eventos atuais e participem ativamente do ciclo de notícias (BOYD; ELLISON, 2007).

2.1.3 Fake News nas redes sociais

O fenômeno das fake news representa um grande problema nas redes sociais. Como qualquer pessoa pode publicar conteúdo online, as redes sociais se tornaram um terreno fértil para a disseminação de informações tendenciosas e muitas vezes falsas. A facilidade de compartilhamento e a rápida disseminação de notícias nas redes sociais podem levar ao aumento das notícias falsas, especialmente quando não estão acompanhadas de avisos ou verificações de fatos (PENNYCOOK; RAND, 2018).

A disseminação de fake news em redes sociais pode ter sérias consequências para a sociedade. As informações falsas podem influenciar a opinião pública, distorcer a percepção da realidade e afetar decisões importantes, como votos políticos (ALLCOTT; GENTZKOW, 2017) e até mesmo causar sérios danos a saúde pública. Portanto, combater a propagação de notícias falsas nas redes sociais tornou-se uma questão crítica, exigindo esforços de educação e verificação de fatos a fim de reduzir seu impacto prejudicial.

A disseminação de notícias falsas sobre questões de saúde, como a desinformação sobre vacinas, pode ter consequências diretas na saúde pública, colocando em risco a segurança de comunidades inteiras (VOSOUGHI; ROY; ARAL, 2018).

Podemos trazer como exemplo a fake news “vacinas causam autismo”. Em 1998, o médico Andrew Wakefield publicou um estudo em uma revista que investigou 12 crianças. Segundo ele, essas crianças apresentaram comportamentos autistas e graves problemas intestinais, e todas tinham vestígios do vírus do sarampo em seus corpos. O médico sugeriu que esses sintomas poderiam estar relacionados à vacina Tríplice Viral, que protege contra sarampo, rubéola e caxumba, e que havia sido administrada em 11 das 12 crianças do estudo. Embora Wakefield tenha mencionado que essa conexão era apenas uma hipótese, a notícia fez com que muitos pais deixassem de vacinar seus filhos contra essas doenças em todo o mundo. Isso teve sérias consequências para a saúde pública (IMMUNIZE, 2024), vale ressaltar que o problema em questão não foi a pesquisa de Wakefield, e sim a divulgação em massa de uma interpretação errada de sua pesquisa.

Devido à propagação de fake news, em especial aquelas relacionadas à vacinação e ao autismo, observamos o ressurgimento de doenças anteriormente erradicadas, como o sarampo. O Brasil havia conquistado, em 2016, o reconhecimento da ONU pela eliminação

dessa doença (LIMA et al., 2020). No entanto, recentemente, o país enfrentou um novo surto de sarampo que afetou vários estados. A disseminação de informações falsas, impulsionada pelas redes sociais, tem tido um impacto extremamente prejudicial na luta contra essas doenças (LIMA et al., 2020).

2.2 Twitter

O Twitter é uma das redes sociais mais populares do mundo (NEWMAN et al., 2022). Ele permite que os usuários interajam publicamente por meio de mensagens curtas de texto, além de enviar mensagens privadas e compartilhar outros tipos de mídia, como imagens e vídeos.

Recentemente o Twitter trocou de nome, deixou de se chamar Twitter e agora se chama de X (VALINOR, 2023). No entanto neste trabalho ainda será chamado de Twitter.

Dentre as funcionalidades do Twitter, destacam-se:

- **Tweet:** É a forma como uma publicação no Twitter é chamada. Um tweet pode conter até 280 caracteres (TWITTER, 2023).
- **Menções:** No Twitter, um usuário pode mencionar outro usuário em um tweet. O usuário mencionado recebe uma notificação de que foi marcado em uma determinada publicação, permitindo que os usuários convidem outras pessoas para visualizarem uma postagem.
- **Reply:** Permite que um usuário responda a um tweet de outro usuário, possibilitando que uma publicação se divida em vários subtópicos e forme uma thread.
- **Thread:** O Twitter organiza as publicações e suas respectivas respostas em forma de thread, que é uma sequência de publicações agrupadas.
- **Cabeça da thread:** É o tweet que inicia uma thread.

Atualmente os usuários do twitter o utilizam para publicar e consumir diversos tipos de conteúdo, como opiniões pessoais, humor, divulgação de produtos e serviços e informações no geral. Como visto na figura 1 o Twitter se destaca como uma fonte de informação jornalística. Jornalistas e redações de jornais tem perfis com milhões de seguidores. Podemos citar como exemplo a BBC News Brasil (@bbcbrasil) que possui cerca de 3.3 milhões de seguidores, o portal G1 (@g1) com mais de 14 milhões de seguidores ou o jornalista Gleen Greenwald (@ggreenwald) com mais de 2 milhões de seguidores.

2.3 Fact-checking

Com a facilidade da difusão de informações em massa e com a democratização do acesso a informações, qualquer pessoa com acesso a internet pode publicar qualquer conteúdo, sem nenhuma forma de verificação. Portanto, os usuários de redes sociais estão suscetíveis a consumir *fake news*, crendo que sejam verdadeiras.

Um dos jeitos de se combater a desinformação é com informação. Neste caso, faz-se necessário a verificação dos fatos publicados em redes sociais. O processo de verificação de fatos é chamado de *fact-checking*.

O processo de *fact-checking*, tradicionalmente dentro do jornalismo, se relaciona com procedimentos e técnicas para verificação de fatos antes da publicação de uma notícia, isso se chama de verificação interna de fatos e por volta das décadas de 1920 e 1930 tornou-se uma função distinta dentro das redações jornalísticas (GRAVES; AMAZEEN, 2019). No entanto, com o crescimento do acesso à internet e a facilidade de compartilhamento de informações, surgiu a necessidade de uma verificação externa de fatos. A verificação externa de fatos consiste na publicação de uma análise baseada em evidências e fatos sobre a acurácia de um determinado texto público (GRAVES; AMAZEEN, 2019).

Atualmente existem diversas agências que fazem esse serviço de verificação externa de fatos. Para indexar as publicações dessas agências foram desenvolvidas ferramentas como o GFCT (Google Fact Checking Tools), por meio das quais os usuários podem pesquisar e obter uma variedade de publicações dessas agências verificadoras de fatos.

2.4 API

A proposta deste trabalho é integrar de forma automatizada, as informações das agências verificadoras de fatos com as redes sociais.

Para isso, é necessário estabelecer uma comunicação entre as ferramentas indexadoras, como o GFCT, e as redes sociais, como o Twitter. Uma das formas de comunicação entre aplicações são as APIs.

Uma API é um conjunto de regras e protocolos que permitem a comunicação entre diferentes sistemas de software (TANENBAUM, 2015). Ela define os métodos e estruturas de dados que desenvolvedores podem utilizar para interagir com um serviço ou aplicativo específico. Uma API atua como uma ponte que possibilita que programas distintos se comuniquem e compartilhem informações de forma estruturada.

As APIs permitem que um serviço se comunique com outros serviços sem precisar saber como eles foram implementados (FERREIRA, 2021). Elas atuam como uma camada intermediária entre a aplicação e o servidor web, processando dados entre os sistemas.

2.4.1 APIs RESTful

Existem diversos tipos de APIs, porém as API Representational State Transfer (REST) são amplamente reconhecidas como o tipo mais utilizado de API na atualidade (POSTMAN, 2023). Sua popularidade se deve à sua simplicidade, eficiência e facilidade de uso. As APIs REST são altamente escaláveis e podem ser implementadas de forma robusta, tornando-as ideais para atender às demandas crescentes das aplicações modernas (RICHARDSON; AMUNDSEN, 2013).

As APIs REST utilizam o protocolo Hypertext Transfer Protocol (HTTP) e seus métodos como padrão de comunicação. Elas aderem ao conceito de recursos que podem ser lidos, criados, atualizados e excluídos.

As API REST geralmente retornam dados em formatos como JavaScript Object Notation (JSON) ou Extensible Markup Language (XML), tornando a integração e a interpretação dos dados mais simples para os desenvolvedores (TANENBAUM, 2015).

2.4.2 HTTP

O HTTP é um protocolo de comunicação amplamente utilizado. Ele foi projetado para permitir a transferência de informações, especialmente documentos hipertexto, entre um cliente (geralmente um navegador) e um servidor web. O HTTP é baseado em uma arquitetura cliente-servidor, onde o cliente faz solicitações de recursos, como páginas da web, e o servidor responde a essas solicitações, enviando os recursos solicitados de volta ao cliente (TANENBAUM, 2015).

Um aspecto importante do HTTP é que ele é um protocolo sem estado, o que significa que cada solicitação é tratada independentemente, sem conhecimento do estado anterior.

Os *endpoints* representam Uniform Resource Locator (URL)s específicas que correspondem a recursos ou funcionalidades do sistema (TANENBAUM, 2015). Cada *endpoint* é associado a um conjunto de métodos HTTP.

Os principais métodos do protocolo HTTP são:

- **GET**: O método GET é utilizado para recuperar informações de um recurso.
- **POST**: O método POST é utilizado para criar um novo recurso.
- **PUT**: O método PUT é utilizado para atualizar um recurso existente.
- **DELETE**: O método DELETE é utilizado para remover um recurso

2.4.3 JSON

JSON é um formato de dados amplamente utilizado para a troca de informações estruturadas entre sistemas. Ele foi originalmente derivado da notação de objetos JavaScript, mas agora é independente de linguagem e é amplamente adotado em muitos contextos, incluindo desenvolvimento web e integração de sistemas.

```
{
   "nome": "Suyan Moriel Viese Moura",
   "idade": 27,
   "telefones": [
        {
            "numero": "123456789",
            "ddd": "48",
            "habilitado": true
        },
        {
            "numero": "98765431",
            "ddd": "47"
            "habilitado": false
        }
   ]
}
```

O JSON acima representa um objeto, pois os elementos delimitados por chaves representam um objeto. Esse objeto possui os seguintes atributos: nome, idade e telefones. O atributo de nome é um texto, enquanto o atributo de idade é um número, já o atributo de telefones é uma lista.

As listas são delimitadas com colchetes, e cada elemento de uma lista é separado por vírgula. Na lista de telefones podemos ver que há dois elementos. Portanto a lista de telefones contém dois objetos, representando um telefone. Esse objeto contém três atributos: número, que nesse caso é um texto; ddd, que também é um texto; e habilitado, que é um booleano.

2.5 Google Fact Checking Tools

O Google Fact Checking Tools é um serviço da Google cuja a finalidade é facilitar a divulgação e a busca de verificações externas de fatos. Ele é composto por duas ferramentas que visam facilitar o trabalho de jornalistas e pesquisadores, permitindo a pesquisa e publicação de análises de fatos (fact-checking) (GOOGLE, 2023). Elas são o Google Explorer (foco deste trabalho) e o Google Markup Tool[2]

[2] A explicação sobre essa ferramenta não será aprofundada pois ela não é usada neste projeto.

2.5.1 Google Markup Tool

O Google Markup Tool é uma das ferramentas de verificação de fatos oferecidas pelo Google. Com o Markup Tool, as agências verificadoras de fato podem marcar e destacar informações em suas páginas da web que foram verificadas como incorretas. Essa marcação especial permite que o Google e outros mecanismos de busca identifiquem e exibam claramente as informações desonestas ou errôneas nos resultados de pesquisa.

Quando as informações verificadas incorretas são identificadas e marcadas, os usuários podem tomar decisões mais informadas ao acessar e compartilhar conteúdo online. Essa abordagem contribui para a promoção da transparência e da precisão das informações disponíveis na web, auxiliando no combate à desinformação e à desonestidade online.

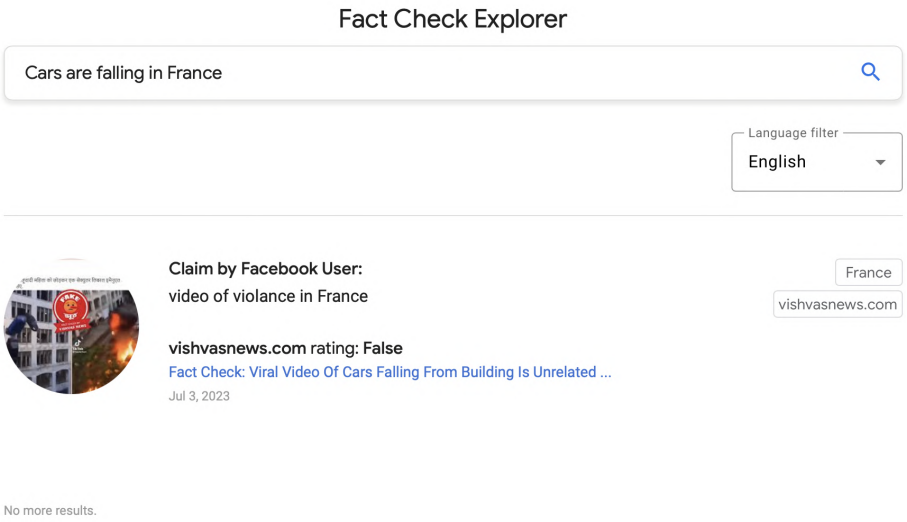
2.5.2 Explorer

O Google Fact-Checking Tools Explorer (GFCTE) é uma ferramenta desenvolvida pela Google com o propósito de auxiliar os usuários na verificação de fatos e na busca por informações verificadas por agências verificadoras de fatos.

A interface se assemelha ao buscador padrão do Google, proporcionando uma experiência familiar aos usuários. No campo de busca, os usuários podem inserir termos, frases ou manchetes que desejam verificar. A semelhança com o buscador tradicional do Google facilita a navegação e utilização para buscas, tornando-a mais acessível (GOOGLE, 2023).

É importante reconhecer que a busca por informações verificadas pode muitas vezes ser um processo inconveniente para os usuários. Atualmente, os usuários precisam sair da rede social em que estão, abrir uma nova aba do navegador e realizar uma pesquisa separada no GFCTE ou em outro buscador de sua escolha. Essa abordagem pode resultar em uma experiência fragmentada e em muitos casos, desencorajar os usuários a verificarem a veracidade das informações antes de compartilhá-las.

Figura 2 – Interface do Explorer



Fonte: Desenvolvida pelo autor

A figura acima é uma *screenshot* da interface do GFCTE [3], no campo de busca, foram colocadas as palavras-chave "Cars are falling in France" (Carros estão caindo na França) e é apresentado um resultado para essa busca. Cada resultado é uma *resposta* a uma potencial notícia falsa.

No resultado, a reivindicação[4] é oriunda de uma publicação no Facebook e diz "video of violance in France" (vídeo de violência na França), ou seja, um usuário no Facebook publicou isso.

Podemos ver também que a organização "vishvasnews.com" avaliou como "falsa" a alegação do "Carros estão caindo na França". Logo abaixo, é possível ver um link o qual redirecionará o usuário a verificação de fatos publicada pela "vishvasnews.com". [5]

[3] Está disponível no link <https://toolbox.google.com/factcheck/explorer>.

[4] Nesse contexto o termo reivindicação se refere a notícia, publicação ou texto que está em análise, uma possível *fake news*.

[5] O contexto dessa fake news é uma onda de protestos que ocorreu após um jovem ser baleado pela policia de paris no dia 23 de julho de 2023. Os franceses em sua maioria estão protestando contra a violência policial, especialmente contra pessoas marginalizadas (MAWAD; KENNEDY; ISAAC, 2023). De acordo com "vishvasnews.com" o vídeo de carros caindo de prédios é uma cena do filme Velozes e Furiosos 8, a qual fora publicada pelo cineasta Justin King.

Todas essas informações exibidas na interface gráfica também são obtidas via API. A imagem a seguir retrata o JSON de retorno para mesma pesquisa feita anteriormente, esta se limita apenas ao primeiro resultado da busca.


```
{
    "claims": [
        {
            "text": "video of violance in France",
            "claimant": "Facebook User",
            "claimDate": "2023-07-03T08:34:09Z",
            "claimReview": [
                {
                    "publisher": {
                        "site": "vishvasnews.com"
                    },
                    "url": "https://www.vishvasnews.com/english/world/fact-check-viral-video-of-cars-falling-from-building-is-unrelated-to-france-violence/",
                    "title": "Fact Check: Viral Video Of Cars Falling From Building Is Unrelated ...",
                    "reviewDate": "2023-07-03T08:34:09Z",
                    "textualRating": "False",
                    "languageCode": "en"
                }
            ]
        }
    ]
}
```


O retorno exposto na imagem acima demonstra que a API da Google retorna uma lista de *claims*[6], cada *claim* é composta pelos seguintes itens

- **text:** O texto da reivindicação que está sendo revisada.
- **claimant:** a fonte da reivindicação.
- **claimDate:** a data na qual a reivindicação foi publicada.
- **claimReview:** é uma lista com a revisão de todas as agências verificadoras de fato.
  - **publisher**: contém os dados da agencia verificadora de fatos que revisou como o nome e o site.
  - **url:** Link [7] da verificação dos fatos.

[6] *Claims* é como a Google chama o objeto que contém as informações relativas a uma verificação de fato especifica.

[7] disponível em <https://encurtador.com.br/xABFI>.

- **title:** o titulo da matéria.
- **textualRating:** o *veredito* que a agencia verificadora de fatos dá em relação àquela reivindicação.
- **languageCode:** O idioma da verificação de fatos.

2.6 Microsserviços

Uma arquitetura de software orientada a microsserviços implica em aplicações que podem ser implementadas, escaladas e testadas de forma independente e que, idealmente, possuem uma única funcionalidade (WASEEM; LIANG; SHAHIN, 2020).

As principais razões para utilizar microsserviços são:

- Reduzir a complexidade de um projeto, utilizando serviços pequenos;
- Escalabilidade;
- Flexibilidade para utilizar diferentes *frameworks* e ferramentas.

A abordagem de microsserviços adotada neste projeto oferece não apenas modularidade e independência, mas também escalabilidade. Ao separar as funcionalidades de um sistema em serviços individuais, é possível expandir o sistema para incorporar diversas tecnologias e plataformas, conforme necessário. Essa flexibilidade permite que o sistema se adapte às demandas em constante evolução do ambiente digital, possibilitando a integração com diferentes redes sociais, ferramentas de verificação de fatos e outras tecnologias relevantes. Esta arquitetura escalável promove um ambiente dinâmico e adaptável, capaz de lidar com os desafios em constante mudança no cenário das informações online (WOLFF, 2016).

2.6.1 Escalabilidade

A escalabilidade é uma característica essencial dos microsserviços, permitindo que os sistemas se adaptem às demandas variáveis do ambiente digital. A escalabilidade pode ser alcançada através da replicação de microsserviços, permitindo que múltiplas instâncias de um serviço sejam implantadas para distribuir a carga e atender a um grande número de solicitações simultâneas. Além disso, a arquitetura de microsserviços facilita a adição ou remoção de serviços conforme necessário, tornando a escalabilidade uma característica essencial para a adaptação eficaz a picos de tráfego e variações na demanda (WOLFF, 2016).

2.6.2 Tecnologias que suportam os microsserviços

As arquiteturas de microsserviços são amplamente suportadas por um conjunto de tecnologias que facilitam o desenvolvimento, implantação e gerenciamento de serviços independentes. Entre essas tecnologias, o Docker tem se destacado como uma das principais ferramentas para a implantação de microsserviços.

O Docker é uma plataforma de código aberto que permite a criação, implantação e execução de aplicativos em contêineres. Os contêineres são unidades leves e portáteis que empacotam todos os componentes necessários para que um aplicativo seja executado, incluindo código, bibliotecas e dependências. Essa abordagem simplifica a implantação de microsserviços, pois os contêineres podem ser criados com ambientes isolados e autossuficientes, garantindo consistência entre diferentes ambientes, como desenvolvimento, teste e produção (DOCKER, 2023).

Uma grande gama de empresas e aplicações utilizam o Docker em seu dia-a-dia. Dentre elas se destacam: Adobe, Paypal e a universidade de Yale (DOCKER, 2023).

2.7 Contêineres e orquestração

Contêineres no geral são dedicados a execução de microsserviços (VAYGHAN et al., 2018). As aplicações distribuídas em microsserviços têm-se tornado cada vez mais complexas e como consequência separadas em vários contêineres diferentes. Por isso faz-se necessário uma forma de *orquestração* destes contêineres para que haja um provisionamento eficiente de recursos (CASALICCHIO, 2016).

Sistemas orquestradores de contêineres permitem uma implantação, escalabilidade e gerenciamento de aplicações em contêineres sem a preocupação com a infra-estrutura na qual eles estão hospedados (GOOGLE, 2024).

Em um ambiente de produção com vários contêineres em execução simultânea, faz-se necessário uma forma eficaz e confiável de garantir que tudo estará operando como o esperado (KUBERNETS, 2023).

2.7.1 Ferramentas para orquestração

Ferramentas como o Kubernetes ou docker-compose permitem esse gerenciamento automático de múltiplos contêineres e fornecem ao desenvolvedor vantagens como autocorreção (é capaz de reiniciar contêineres que falham) e armazenamento de dados sensíveis (como senhas ou *tokens*) de forma segura (KUBERNETS, 2023).

O docker-compose é uma ferramenta para gerenciar múltiplos contêineres utilizando um único arquivo de configuração (DOCKER, c2024a).

2.7.2 Criação de contêiner com docker

O Docker é capaz de construir um contêiner a partir de um *dockerfile*. O *dockerfile* nada mais é que um arquivo de texto contendo as instruções e comandos usados para construir um contêiner (DOCKER, c2024b). Essencialmente, o *dockerfile* contém as instruções para a criação de um novo contêiner.

2.7.3 Orquestração de contêineres com docker-compose

O docker-compose é uma ferramenta para organizar e executar aplicações distribuídas em multiplos contêineres usando um único arquivo de configuração (DOCKER, c2024a). O docker-compose, similar ao docker, utiliza um arquivo de texto para configuração (DOCKER, c2024c). O arquivo de configuração padrão docker-compose.yml e ele permite a divisão em vários arquivos diferentes para maior organização de projetos grandes (DOCKER, c2024c).

2.8 Agentes

Agentes são softwares capazes de perceber seu ambiente, tomar decisões com base em suas percepções e agir sobre o ambiente para alcançar seus objetivos (WOOLDRIDGE, 2009).

Agentes são utilizados em diversas aplicações, desde sistemas de recomendação em *e-commerce* até sistemas de controle de tráfego aéreo. A utilização de agentes permite a criação de sistemas mais inteligentes e adaptativos, capazes de lidar com ambientes complexos e dinâmicos (JENNINGS, 2000).

Os agentes reativos são uma abordagem simples de agentes na arquitetura de software. Eles reagem diretamente às percepções do ambiente, sem manter um estado interno completo (BROOKS, 1999). Esses agentes são eficientes em termos de processamento e memória, o que os torna adequados para sistemas distribuídos. No entanto, eles podem ser limitados em tarefas mais complexas (WOOLDRIDGE, 2009).

Um agente reativo, por exemplo, pode ser um bot que responde de forma autônoma às menções feitas pelos usuários no Twitter. Além disso, um agente também pode ser responsável por realizar buscas em uma base de dados e interagir com outros agentes, sejam eles artificiais ou humanos. Esses exemplos ilustram a versatilidade e a capacidade de adaptação dos agentes, permitindo que o sistema atenda às demandas específicas de diferentes contextos e interaja de forma eficiente com os usuários e outras entidades envolvidas.

3 DESENVOLVIMENTO

A proposta deste trabalho consiste no desenvolvimento de dois agentes, que de forma coordenada, integrarão um usuário *bot* de uma rede social (neste caso o Twitter) com um serviço indexador de verificações de fato (neste caso o GFCT). O AT é o usuário *bot* integrado ao Twitter, enquanto o AFC é responsável pela comunicação com o serviço indexador.

Uma conta específica no Twitter foi criada e será utilizada pelo AT. Dado que essa conta é destinada a ser utilizada por uma aplicação, ela é caracterizada como um *bot*. O papel principal do AT é monitorar as menções ao *bot* e responder aos tweets nos quais foi mencionado, fornecendo fontes e informações verificadas.

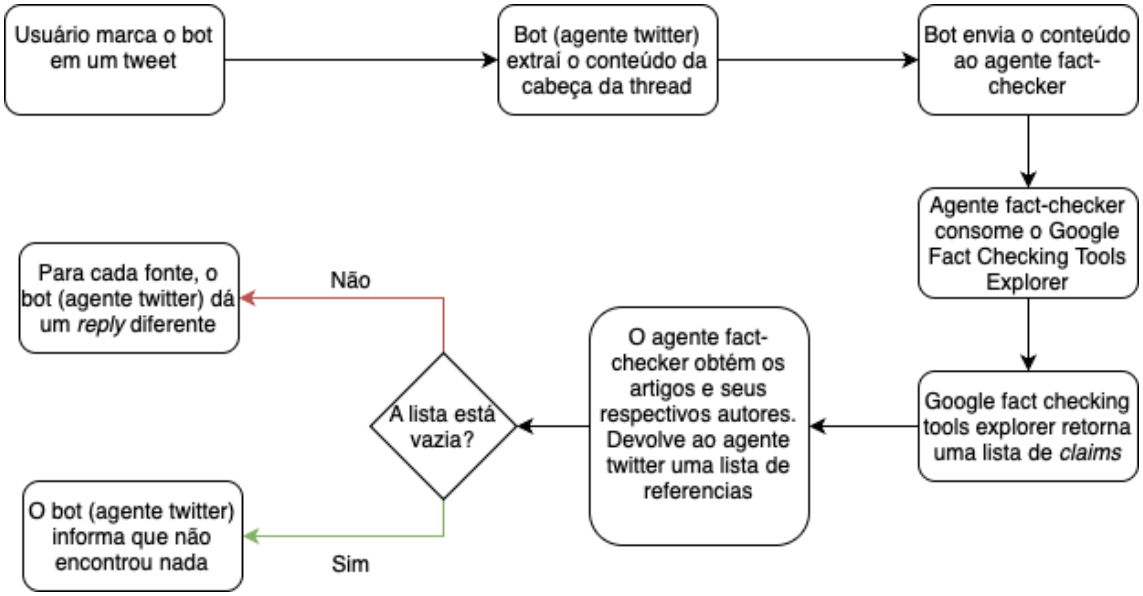
Essa arquitetura foi projetada com a finalidade de proporcionar uma abordagem eficaz na disseminação de informações verificadas e fontes de confiança. Para atingir esse objetivo, foi concebida a integração de agentes autônomos que operam como microsserviços independentes, cada um desempenhando tarefas específicas.

Essa estrutura oferece uma solução sólida para garantir a precisão das informações e a disponibilidade de fontes confiáveis em um ambiente altamente dinâmico e interconectado. O fluxograma a seguir demonstra como o sistema se comporta.

A proposta de integração de um *bot* diretamente na rede social tem o potencial de resolver essa falta de comodidade. Ao oferecer aos usuários a capacidade de verificar informações sem sair da plataforma. Essa abordagem reduz significativamente as barreiras para a verificação de fatos.

Somado a isso, uma vez que um usuário acionou o *bot*, as fontes obtidas pelo *bot* permanecerão disponíveis na rede social. As informações verificadas permanecerão na rede social para que outros usuários possam vê-las, sem nem sequer precisar acionar o *bot*.

Figura 3 – Fluxograma do sistema



Fonte: Desenvolvida pelo autor

3.1 Modularidade da arquitetura

O projeto adota uma arquitetura baseada em microsserviços, nos quais foram implementados dois agentes, cada um encarregado de executar uma função específica. [1]

Cada agente está integrado a um sistema externo, sendo um deles conectado ao Twitter e o outro ao GFCT. Esses agentes são chamados, respectivamente, de "Agente Twitter" (AT) e "Agente Fact Checker" (AFC).

Esses agentes se comunicam por meio de API REST. O AFC consome a API do GFCT e também oferece sua própria API, enquanto o AT consome a API do Twitter e a API do AFC. Essa arquitetura permite a troca eficiente de informações entre os agentes e os sistemas externos, facilitando a verificação de fatos e a resposta a consultas dos usuários.

Cada agente, atuando como um módulo independente, possui versatilidade de ser substituído por outro agente similar, desde que sejam mantidos os parâmetros de entrada e saída esperados dentro da estrutura estabelecida. Isso implica que a arquitetura é adaptável o suficiente para permitir integrações com várias outras redes sociais, da mesma forma que é capaz de se conectar a diversos serviços indexadores. Essa flexibilidade proporciona uma base robusta para a expansão do sistema, seja para incluir novas redes sociais ou para incorporar serviços adicionais de verificação de fatos.

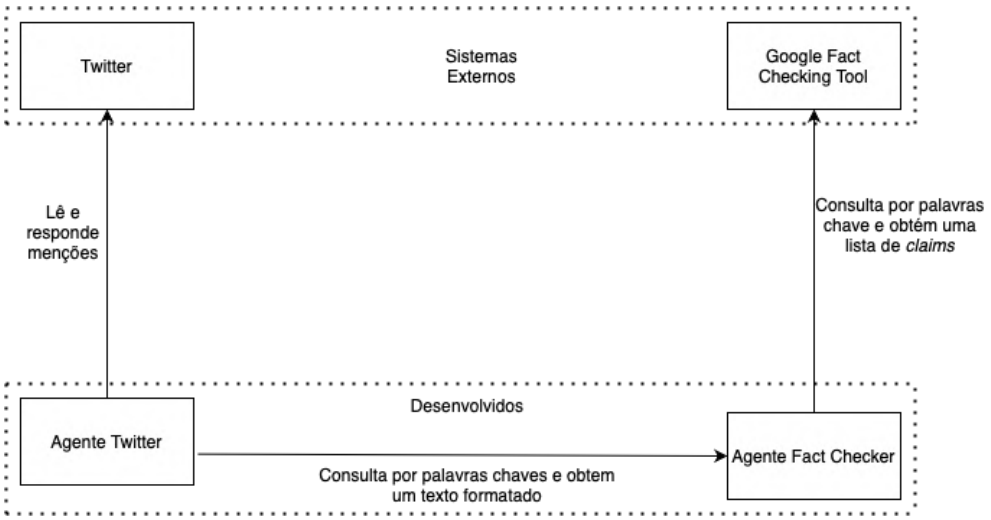
Na figura a seguir podemos ver um diagrama de comunicação entre as entidades do sistema. O AT se comunica com o Twitter e também com o AFC. Essa lógica de comunicação é inevitável, uma vez que o AT deve ser integrado a uma rede social, ele deve

[1] Os termos microsserviços e agentes serão utilizados intercambiavelmente pois cada agente será responsável por um microsserviço.

necessariamente se comunicar com uma rede social. E também como este agente precisa realizar buscas em fontes indexadoras, no caso o GFCT, ele precisa se comunicar com o AFC. O AFC por sua vez precisa se comunicar com a fonte indexadora de fatos, o GFCT.

Os sistemas externos expostos no diagrama abaixo demonstram sistemas que já existem, no caso o Twitter e o GFCT. Os desenvolvidos são os microsserviços criados neste projeto.

Figura 4 – Arquitetura do sistema



Fonte: Desenvolvida pelo autor

3.1.1 Padronização da arquitetura

Para garantir uma operação eficaz e flexível do sistema, foi estabelecido um padrão de comunicação consistente entre os dois agentes envolvidos no projeto. Essa abordagem é fundamental para simplificar o desenvolvimento de futuras integrações e manter a integridade das operações.

Neste contexto, temos dois agentes fundamentais: AT, que está conectado à rede social (no caso o Twitter), e o AFC, responsável pela integração com o serviço de indexação de verificação de fatos. O AFC é responsável por realizar consultas em uma fonte indexadora específica (GFCT).

Para viabilizar essa comunicação, o AFC possui sua própria API, que oferece um método (no caso o GET) para receber um texto como entrada, o qual é passado na própria URL. Esse texto serve como base para realizar buscas no GFCT. Em resposta, a API do AFC retorna uma lista de respostas prontas, em forma de JSON, sendo que cada resposta é um texto formatado de acordo com as fontes identificadas no GFCT.

Por sua vez, o agente se responsabiliza pela interação com os usuários da rede social. Ele recebe mensagens e consultas dos usuários, executando uma requisições ao AFC.

Posteriormente, o AT entrega aos usuários as informações obtidas a partir das respostas fornecidas pelo AFC.

Esse processo é possível graças à conformidade de ambos os agentes com o padrão de comunicação estabelecido, garantindo uma operação coordenada do sistema.

3.2 Agente fact-checker

O AFC é integrado a fonte indexadora de fatos (GFCT). Sua função dentro deste projeto é receber um texto qualquer do AT, realizar consultas no GFCT e devolver ao AT respostas prontas com base nas fontes obtidas.

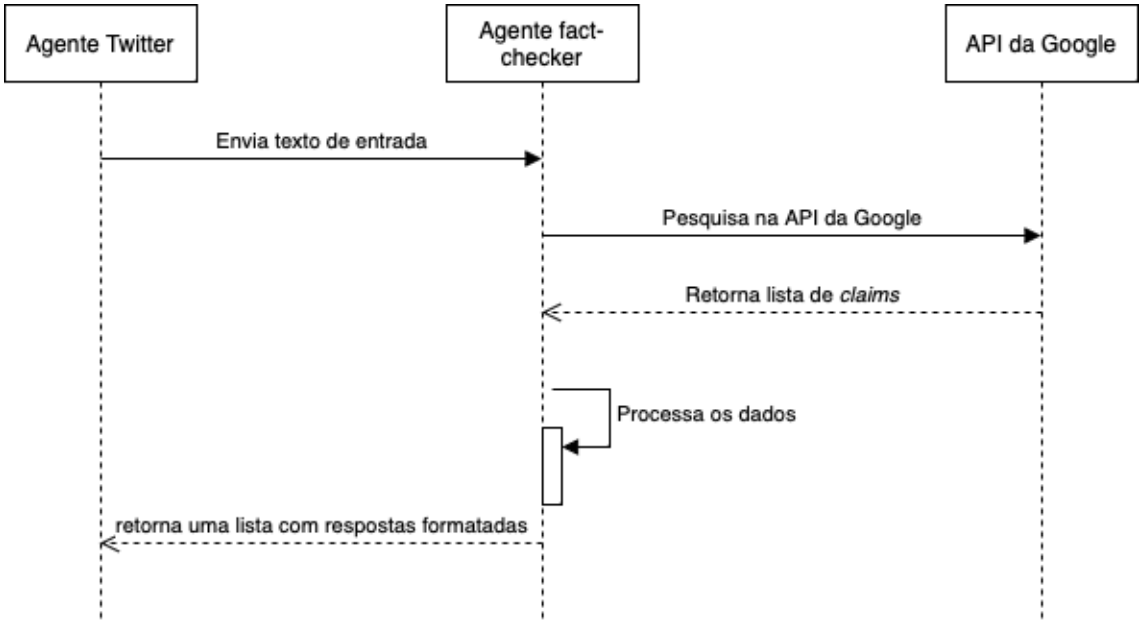
Uma vez que o AFC não mantém nenhum estado interno, ele é considerado um agente reativo. O AFC não mantém nenhum estado interno pois sua função é simples: realizar consultas, preparar respostas com base nas fontes obtidas e devolver uma lista de resposta.

Dessa forma, este microsserviço encapsula a complexidade de realizar buscas e a complexidade de formatar as respostas.

Podemos ver na figura 5 o papel do AFC.

Na figura 5 o AT envia um texto para o AFC. O AFC envia esse texto ao GFCT. O GFCT retorna a lista de *claims*. O AFC processa estes dados, extraindo os campos necessários do JSON (nome de quem publicou, titulo da matéria que está sendo avaliada,

Figura 5 – Diagrama de forma simplificada do funcionamento do agente fact-checker



Fonte: Desenvolvida pelo autor

textualRating e a URL da verificação externa dos fatos), com essas informações o AFC converte tudo para um texto e devolve para o AT uma lista de textos, prontos para serem postados.

3.2.1 API

Para implementação de sua API foi usado a linguagem Java e o *framework* Spark, este *framework* permite a implementação de forma simples de uma API REST (SPARK, 2021).

A seguir será explicado o principal método da API, o checkPretty que é utilizado para consultar o GFCT e retornar textos prontos para o AT replicar no Twitter.

O objetivo do método *checkPretty* é retornar uma lista de textos que representam os resultados da consulta à API do GFCT de forma clara e organizada.

Este método recebe um texto, oriundo de um tweet, como parâmetro de entrada. Ele utiliza esse texto para realizar suas consultas internamente.

O texto de entrada é enviado ao GFCT, o GFCT por sua vez retorna uma lista de *claims* (ver figura 3). O AFC, por sua vez, extrai a fonte (url) e agência verificadora (*publisher name*) e então formata suas respostas.

Neste caso, usando o JSON 2.2 como exemplo, o AFC geraria o texto exemplificado na imagem a seguir.

Fonte: Desenvolvida pelo autor

Graças ao framework Spark a implementação de um endpoint é simples como podemos ver no trecho de código abaixo.

```
1  Spark.get("checkPretty", (request, response) -> {
2      try {
3          String s = request.queryString();
4          logger.info("Received request to check claims: " + s);
5
6          JSONObject json = new JSONObject();
7          ToolBoxResponse toolBoxResponse = controller.getClaims(s);
8
9          json.put("data",
10                 toolBoxResponse != null && toolBoxResponse.claims != null ?
11                         toolBoxResponse
12                                 .claims
13                                 .stream()
14                                 .map(Claim::prettyClaim)
15                                 .collect(Collectors.toList())
16                         : new JSONArray()
17         );
18
19
20         return json.toString();
21     } catch (Throwable t) {
```

```
22          logger.error(String.valueOf(t));
23          throw new WebServiceException("Failed to retrieve data");
24      }
25  });
```

Basta passar o nome da rota utilizada como primeiro parâmetro, neste caso "checkPretty", e como segundo parâmetro uma função que receberá o *request* e *response* como argumentos. Podemos ver que é um código relativamente simples. Na linha 3 obtêm-se o texto de entrada, logo na linha 7 é executado uma consulta ao GFCT e da linha 9 a linha 17 é feito o mapeamento de *claim* para texto. Na linha 20, finalmente, é devolvido a lista de respostas com base nas consultas do GFCT, esse *return* da linha 20 faz com que o método retorne código 200, caso haja quaisquer exceções o método retornará apenas erro 500. Essa API possui apenas este método, o checkPretty.

3.3 Agente Twitter

Este agente é responsável por controlar as ações do bot. Ele lê e responde aos tweets nos quais foi mencionado.

Periodicamente, ele consome a API do Twitter e obtém uma lista de menções. Esse *endpoint* do Twitter devolve os tweets, nos quais o nosso usuário do bot fora mencionado, a partir de um ID (identificador de um tweet) especifico, ou seja, o AT salva no seu banco de dados o ID do último tweet processado, e todos os tweets a partir dele ainda não foram lidos. Para cada tweet não lido, ele buscará a cabeça da *thread* e os colocará numa fila.

Para cada elemento da fila, o AT extrai o conteúdo do tweet e o envia para o AFC. O AFC, por sua vez, devolve uma lista de respostas. Para cada resposta, o AT replica o tweet no qual fora marcado.

O AT armazena todas as menções em seu banco de dados interno a fim de saber quais tweets ele já respondeu e quais ele precisa responder. Para desenvolvê-lo foi usado a linguagem Python com a biblioteca Tweepy. O Tweepy implementa uma fácil integração com a API do Twitter, de forma que se torna simples fazer ações como publicar um tweet, responder a uma menção, enviar um *direct*[2] e até mesmo bloquear um usuário (TWEEPY, 2021).

3.3.1 Estrutura de classes

Nesta seção serão discutidas as principais classes desse microsserviço e suas funcionalidades.

3.3.1.1 FactCheckingClient

Classe responsável por se comunicar com o AFC, utiliza a biblioteca “requests” da linguagem python. Seu construtor espera como parâmetros uma lista de palavras, as quais serão transformadas, internamente, em parâmetros de *query* [3].

3.3.1.2 TweetHandler

Essa classe é responsável por lidar com tweets. Ela é capaz de: extrair o conteúdo de um tweet, extrair metadados de um link, obter ID do tweet e replicar um tweet.

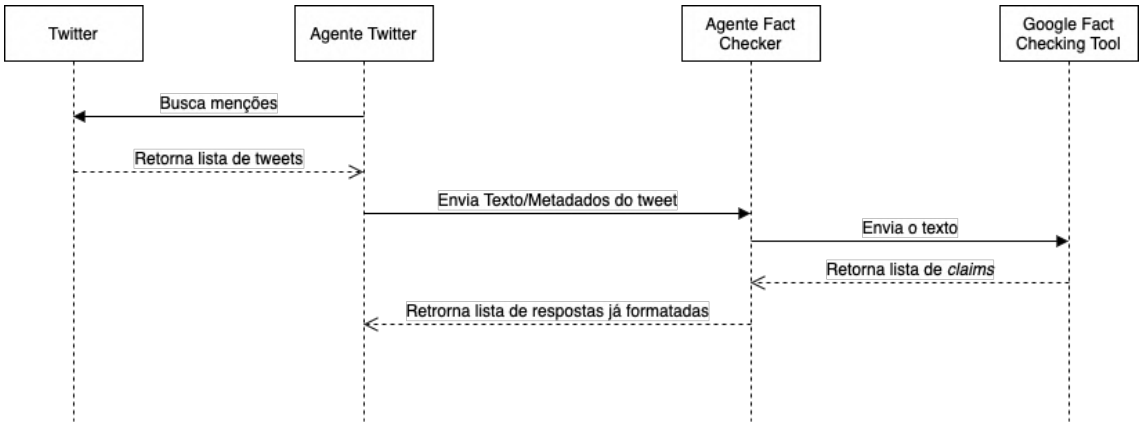
O AT, em seu *loop* principal, fica constantemente checando por novas menções a partir do ID de sua última menção. Quando há uma nova menção, ele extrai o conteúdo da cabeça thread. Caso o conteúdo seja apenas um link, ele extrai os metadados do link.

[2] Directs são mensagens privadas no Twitter.

[3] São variáveis que são passadas na própria url, exemplo <https://exemplo.com/exemplo?variavelDemonstracao=1> neste caso é passado uma variável chamada de variavelDemonstracao e seu valor é igual a 1.

Uma vez que o conteúdo foi extraído, ele instancia um FactCheckClient, que processa os dados extraídos. O FactCheckClient repassa esse conteúdo ao AFC e o AFC retorna ao FactCheckClient uma lista de respostas. O TweetHandler usa essas respostas para replicar as menções. Todas as ações são registradas em banco de dados.

Figura 6 – Diagrama de interação entre as 4 entidades do sistema

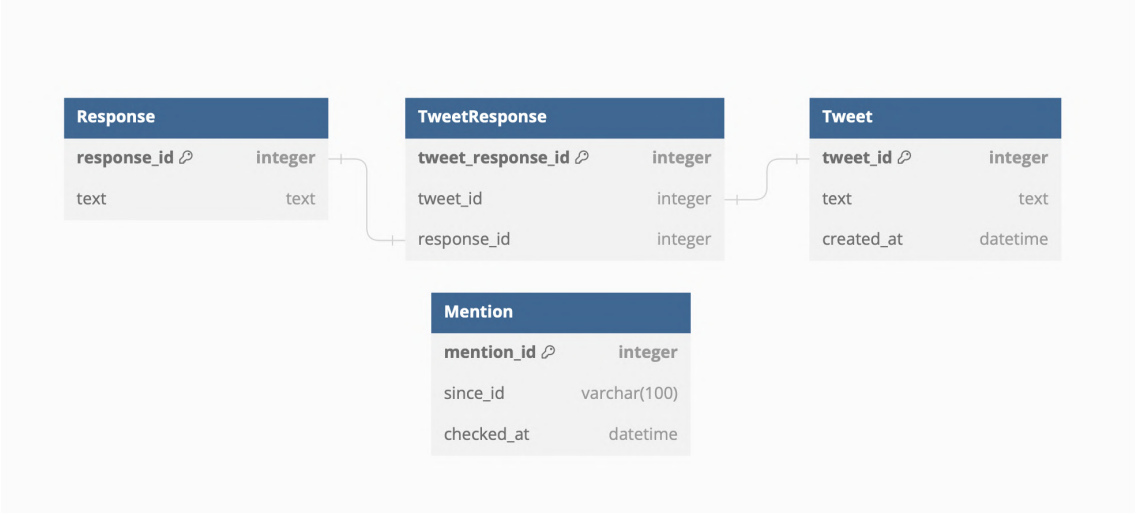


Fonte: Desenvolvida pelo autor

3.3.2 Banco de dados

Para esta aplicação, foi desenvolvido um banco de dados que é usado pelo AT, este banco tem como principal funcionalidade impedir que as mensagens sejam tratadas mais de uma vez.

Figura 7 – Arquitetura do banco



Fonte: Desenvolvida pelo autor

3.3.2.1 Mention

A tabela Mention tem como função indexar todas as menções já tratadas. A coluna "mention_id" é referente ao id da menção tratada. A coluna menção "since_id" é o id da última menção lida pelo AT e a coluna "checked_at" é o instante no qual aquela menção foi processada. Como a API do Twitter permite buscar menções a partir de um ID, essa tabela otimiza o tempo de busca e impede que haja processamentos desnecessários e repetidos. Ela não se relaciona com as outras tabelas pois a sua função é armazenar o último tweet lido. Inicialmente a ideia era atrelar a menção a uma resposta para servir como um cachê, todavia, isso seria ineficaz pois conforme o tempo se passa, novas fontes surgem, ou seja, o cachê seria desatualizado.

3.3.2.2 Tweet, Response e TweetResponse

A tabela Tweet armazenará um tweet enquanto a Response armazenará a resposta dada naquele tweet. Por fim a tabela TweetReponse simplesmente relaciona a Tweet com a Response. Estas tabelas existem para guardar todas as interações do AT. Ou seja, o AT pode obter todos os tweets nos quais fora mencionado e todas as respostas que já deu.

3.4 Implementação

Na implementação da aplicação é utilizado o Docker para criação de contêineres e o docker-compose para orquestração dos contêineres criados.

3.4.1 Criação do contêiner do AFC

A seguir está o código usado para criação do contêiner do AFC.

```
1 FROM openjdk:8-jre
2 WORKDIR /app
3 COPY target/FactCheckToolbox-1.1.jar /app/FactCheckToolbox-1.1.jar
4 COPY target/lib /app/lib
5 EXPOSE 8080
6 CMD ["java", "-cp", ".:FactCheckToolbox-1.1.jar:lib/*", "FactCheckToolbox"]
```

Como dito anteriormente, o Dockerfile contém as instruções para criação de um contêiner. Neste Dockerfile é definido a versão do JDK java (java 8), copiado o FactCheck-Toolbox para dentro do contêiner juntamente das dependências, a seguir a porta 8080 é exposta e por fim é executado o jar.

3.4.2 Criação do contêiner do AT

A seguir está o código usado para a criação do contêiner do AT

```
1 FROM python:3.8-slim
2 WORKDIR /app
3 COPY . /app
4 RUN pip install --no-cache-dir -r requirements.txt
5 ENV NAME agenttwitter
6 CMD ["python", "./main.py"]
```

Similar ao anterior, é definido o interpretador de python que será usado pelo projeto, copiado os dados do projeto para dentro do contêiner, instalado as dependências, definido o nome do ambiente e por fim executa-se o main.

3.4.3 docker-compose

Já o docker-compose permite configurar múltiplos contêineres utilizando um único arquivo.

```
1 version: "3"
2
3 services:
4   agentefactchecker:
```

```
5     image: factchecktool:17
6     ports:
7       - "8080:8080"
8
9   agentetwitter:
10    image: agenttwitter:16
11    environment:
12      DB_USER: admin
13      DB_PASS: password
14      DB_HOST: mysql-service
15      DB_NAME: tcc
16      CONSUMER_KEY: xdYnOeyNldGj9KSYwXOnSDIw3
17      CONSUMER_SECRET: 9yhNiiZXECvrJz9t4JstFBYmJA1ayA2EqfvF0JSEXXfWUZb8Iu
18      ACCESS_TOKEN: 1495378006994571267-Q72Pnf5uVJin7ep6DhtYyoTiTxDjui
19      ACCESS_TOKENS_SECRET: Zh7OTyR12Zaa3oVabHP7NH7YAvViJGTWYKUjiGVfRH5bS
20      MY_USER_ID: 23447520
21
22  mysql-service:
23    image: mysql
24    environment:
25      MYSQL_ROOT_PASSWORD: rootpassword
26      MYSQL_DATABASE: tcc
27      MYSQL_USER: admin
28      MYSQL_PASSWORD: password
29    ports:
30      - "3306:3306"
31    volumes:
32      - ./dump.sql:/docker-entrypoint-initdb.d/dump.sql
```

Já no docker-compose acima é instanciado os serviços deste projeto, o AFC, o AT e o banco de dados em MySQL. O AT recebe diversos parâmetros, são eles:

- DB_USER: Usuário do banco de dados
- DB_PASS: Senha do banco de dados
- DB_HOST: *Host* do banco de dados. Neste caso, como o banco de dados é um outro contêiner, basta passar o nome do serviço.
- DB_NAME: Nome do banco de dados.

Já os parâmetros CONSUMER_KEY, CONSUMER_SECRET, ACCESS_TOKEN, ACCESS_TOKENS_SECRET e MY_USER_ID são parâmetros de autenticação para a API do Tweepy.

4 TESTES E RESULTADOS

4.1 Testes

A realização de testes desempenha um papel crucial neste projeto. Dada a complexidade da arquitetura, que envolve a interação de dois agentes autônomos, a integração com serviços externos e a manipulação de dados em tempo real, é fundamental garantir a confiabilidade e o desempenho do sistema. Os testes permitem identificar e corrigir possíveis falhas de comunicação entre os agentes, validar a precisão das informações fornecidas pelo AFC e garantir que o AT responda de maneira adequada às consultas dos usuários.

4.1.1 Metodologia

A metodologia usada para testar o sistema foi a mais direta e próxima ao cenário de uso real possível. Foi simulado uma comunicação de um usuário com o bot.

4.1.2 Implementação dos agentes

Para a criação dos agentes foi utilizado o docker e para a orquestração de todos os contêineres foi utilizado o docker-compose. O uso do docker-compose facilitou a execução simultânea dos serviços pois ele permite a execução de múltiplos microsserviços de forma ordenada e simples. Foi necessário apenas um arquivo de configuração e uma linha de comando para executar o projeto.

4.1.3 Limitações e cenário de testes

Infelizmente não foi possível testar os agentes com mais de uma conta devido a indisponibilidade de múltiplas contas no Twitter. Portanto o teste foi feito do agente respondendo a si mesmo. Embora pareça estranho, esse teste é capaz de validar a arquitetura uma vez que todos os passos do sistema são atendidos, já que é uma simulação do cenário de uso projetado para este sistema. Os pontos validados foram:

- O AT ser capaz de ler os tweets nos quais foi mencionado.
- O AT ser capaz de extrair os dados dos tweets nos quais foi mencionado.
- O AT ser capaz de enviar o texto ao AFC.
- O AFC ser capaz de receber a requisição do AT.
- O AFC ser capaz de obter o texto de entrada.

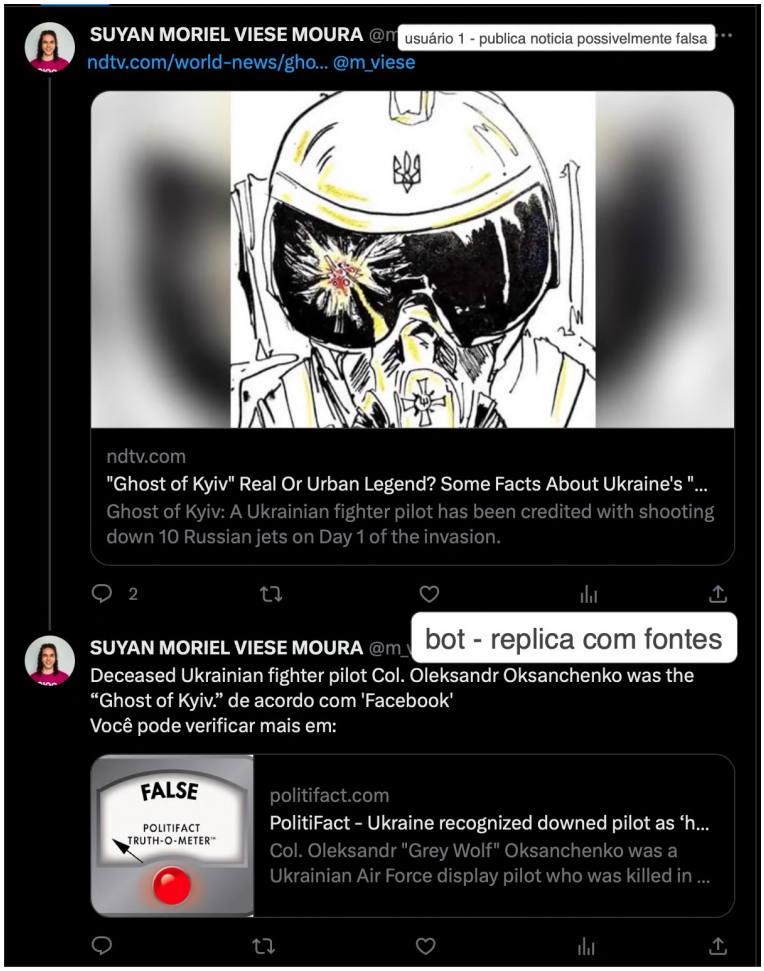
- O AFC ser capaz de enviar esse texto ao GFCT.
- A API do GFCT ser capaz de receber e processar os dados oriundos do AFC.
- O AFC ser capaz de processar o retorno do GFCT.
- O AFC ser capaz de converter as *claims* em texto.
- O AFC ser capaz de devolver os textos ao AT.
- O AT ser capaz de replicar o tweet no qual foi mencionado com as respostas do AFC.
- O AT ser capaz de salvar os dados do tweet no qual foi mencionado, tal como todos os dados envolvidos no evento.
- O AT ser capaz de manter um estado interno. Ou seja, o AT não reprocessar um tweet já processado.

4.1.4 Resultado dos testes

A figura a seguir é o primeiro teste realizado. Devido as limitações citadas anteriormente, a mesma conta será usada para emular um usuário normal e para ação do bot.

No primeiro tweet, o usuário publicou uma notícia (um link) e a seguir marcou o bot. Imediatamente podemos ver o bot replicando ao tweet no qual fora mencionado com uma fonte.[1]

Figura 8 – Teste com link



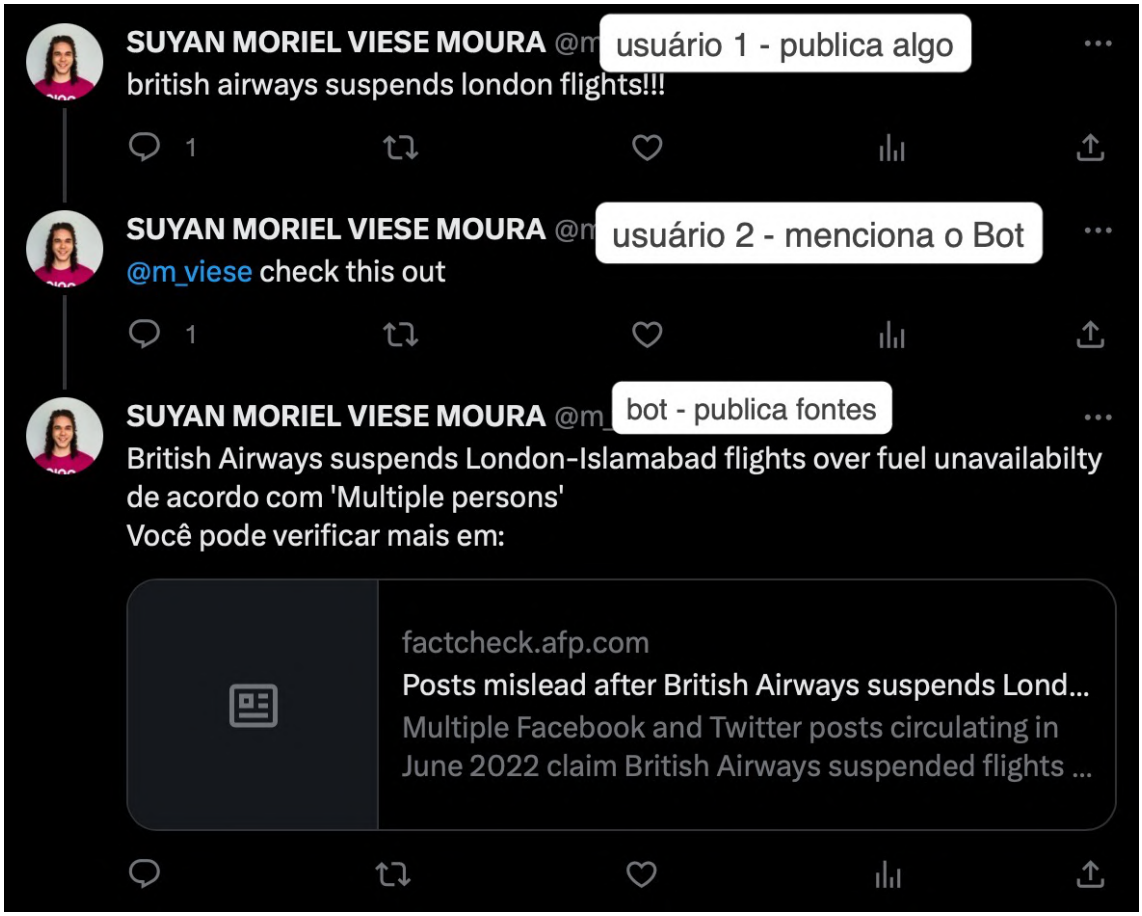
Fonte: Desenvolvida pelo autor

Nesse teste todo fluxo foi validado. O bot foi capaz de ler o tweet no qual ele foi mencionado e foi capaz de extrair os dados do tweet no qual foi mencionado. Após extrair esses dados, ele os enviou ao AFC o qual realizou buscas no GFCT. O GFCT por sua vez devolveu ao AFC uma lista com apenas uma *claim*. O AFC converteu essa *claim* em

[1] O contexto da notícia sobre “O fantasma de Kiev” (tradução livre) é sobre um suposto piloto ucraniano que abateu mais de 40 aviões russos durante a guerra da Ucrânia. A forças áreas ucranianas desmentiram a notícia, não se trata de um piloto mas sim de um batalhão (PETER, 2022).

um texto e devolveu isso em forma de lista (com um elemento) ao AT. o AT por sua vez replicou o tweet inicial com o texto formatado contendo a fonte.

Figura 9 – Teste de extrair conteúdo da cabeça da thread

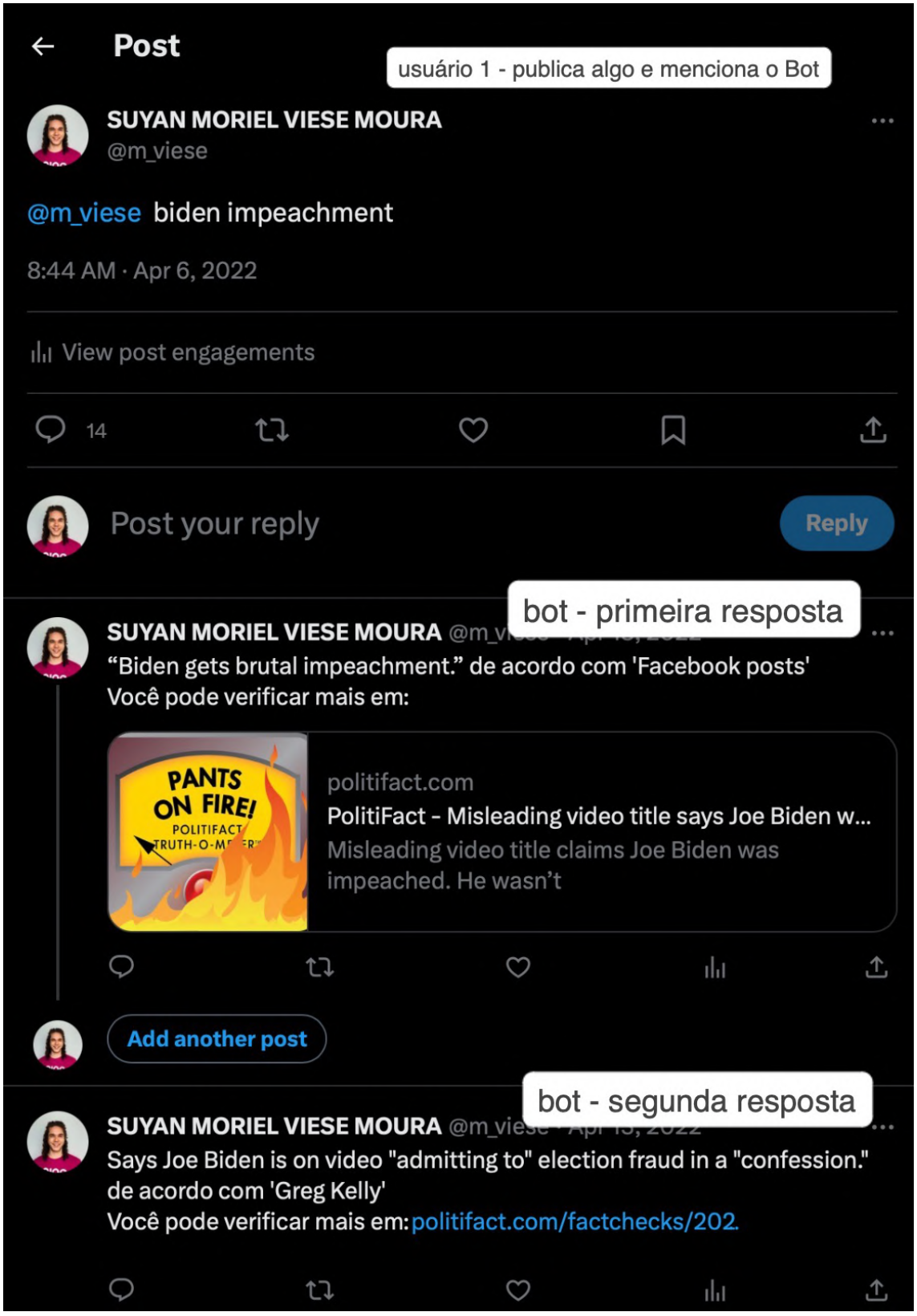


Fonte: Desenvolvida pelo autor

Já na figura acima, o usuário 1 publica um tweet. O usuário 2, desconfiado, menciona o bot na thread a fim de obter mais informações. O bot por sua vez, realiza todo fluxo planejado e replica com fontes. [2]

[2] O contexto da imagem é a alegação circulante de que a British Airways suspendeu voos entre Londres e Islamabad devido à falta de combustível na capital do Paquistão após aumentos nos preços dos combustíveis. No entanto, tanto a empresa quanto a Autoridade de Aviação Civil do Paquistão desmentiram essa alegação, afirmando que a suspensão não estava relacionada à disponibilidade de combustível e que não havia escassez no Aeroporto Internacional de Islamabad (AFP Fact Check, 2022).

Figura 10 – Teste múltiplas respostas



Fonte: Desenvolvida pelo autor

O AFC devolve uma lista de respostas prontas, uma resposta para cada fonte

obtida. Como podemos ver, ele obteve duas fontes e portanto, publicou duas respostas[3]. Quando ele não encontra nada o AFC informa que não obteve nenhuma fonte.

[3] O contexto da notícia era sobre o possível impeachment do presidente americano Joe Biden. Na verdade se trata apenas da câmara dos EUA que aprovou o inquerito para o impeachment do mesmo (G1, 2023)

5 CONCLUSÃO

Este trabalho teve como objetivo desenvolver um robô capaz de ler os tweets nos quais foi mencionado, consultar o seu conteúdo no GFCT e responder ao tweet com sugestões de leitura relacionadas ao conteúdo da postagem.

Uma arquitetura orientada a microsserviços foi proposta de tal forma que o projeto foi organizado em agentes permitindo uma modularização do projeto em si.

Um agente é responsável por se comunicar com o Twitter, obtendo menções e respondendo aos tweets. O outro agente é responsável por consultar o GFCT.

Dentro do AT foi proposto um banco de dados o qual permitirá uma otimização do sistema, não só armazenando as menções que já foram tratadas, mas também salvando tweets e as respostas obtidas pelo AFC. Porém, essa funcionalidade foi descartada uma vez que podem surgir mais análises de fatos de agências verificadoras, de forma que a resposta a um tweet não necessariamente será sempre a mesma.

Ao longo do processo de desenvolvimento desse trabalho houve mudanças nas diretrizes internas do twitter. O Twitter passou a cobrar pelo uso de suas APIs inviabilizando mais testes. Além disso, durante a fase de testes percebeu-se uma escassez de informações em português. Somado a essa escassez, nem sempre as fontes encontradas nem sempre são tão coerentes com o tweet em questão.

Percebeu-se inúmeras limitações do GFCT, seu algoritmo de busca não demonstra ser tão eficaz quanto o buscador Google, além de ficar evidente que o GFCT precisa que as agencias verificadoras de fato se ajustem para que sejam indexadas por ele, o que o torna de certa forma ineficiente, especialmente quando buscamos por algo em português.

Apesar destas limitações a ideia por trás do projeto provou-se eficaz, uma vez que o Twitter adotou um sistema similar de indicar fontes, opiniões contrárias a narrativas e até mesmo rotulamento de tweets.

5.1 Trabalhos futuros

Devido a arquitetura projetada para o bot, é possível integrá-lo a outras redes sociais e a outras fontes indexadoras. Tendo em vista que a API do Twitter é paga e é inviável de utilizá-la no momento, uma sugestão de trabalho futuro seria integrar o bot a uma outra rede social. Outra possibilidade seria utilizar outras fontes indexadoras de verificações de fatos, uma vez que o GFCT mostrou-se extremamente limitado. A arquitetura desenvolvida nesse projeto, apesar de simples, é versátil e permitirá inúmeros projetos que tenham ideias similares.

REFERÊNCIAS

AFP Fact Check. *British Airways flight suspension between London and Islamabad*. 2022. <https://factcheck.afp.com/doc.afp.com.32C39UY>. Citado na página 43.

ALLCOTT, H.; GENTZKOW, M. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, American Economic Association, v. 31, n. 2, p. 211–236, 2017. Disponível em: <https://www.aeaweb.org/articles?id=10.1257%2Fjep.31.2.211>. Citado na página 16.

BOYD danah; ELLISON, N. B. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, v. 13, n. 1, p. 210–230, 2007. Citado 2 vezes nas páginas 15 e 16.

BROOKS, R. A. *Cambrian Intelligence: The Early History of the New AI*. [S.l.]: MIT Press, 1999. Citado na página 26.

CASALICCHIO, E. Autonomic orchestration of containers: Problem definition and research challenges. In: *VALUETOOLS*. [S.l.: s.n.], 2016. Citado na página 25.

DAVIS, W.; WARREN, T. *Twitter is now x as the Little Blue Bird disappears*. 2023. <https://www.theverge.com/2023/7/24/23804973/twitter-x-logo-brand-replacement>. Citado na página 3.

DOCKER. *Docker Compose overview — docs.docker.com*. c2024. <https://docs.docker.com/compose/>. [Accessed 16-01-2024]. Citado 2 vezes nas páginas 25 e 26.

DOCKER. *Dockerfile reference — docs.docker.com*. c2024. <https://docs.docker.com/engine/reference/builder/>. [Accessed 16-01-2024]. Citado na página 26.

DOCKER. *Understand the Compose file — docs.docker.com*. c2024. <https://docs.docker.com/compose/compose-yaml-file/>. [Accessed 16-01-2024]. Citado na página 26.

DOCKER, I. *Docker - Build, Ship, and Run Any App, Anywhere*. 2023. Acessado em 05 de setembro de 2023. Disponível em: <https://www.docker.com/>. Citado na página 25.

FERREIRA, A. G. *Interface de programação de aplicações (API) e web services*. [S.l.: s.n.], 2021. Citado na página 18.

G1. *Congresso dos EUA inicia investigação formal sobre possível impeachment de Biden*. 2023. Disponível em: <https://g1.globo.com/mundo/noticia/2023/12/13/congresso-dos-eua-inicia-investigacao-formal-sobre-possivel-impeachment-de-biden.ghtml>. Citado na página 45.

GOOGLE. *Fact Check Tools*. 2023. Disponível em: <https://toolbox.google.com/factcheck/about#fce-goal>. Citado 2 vezes nas páginas 20 e 21.

GOOGLE. *What Is container orchestration Google Cloud*. 2024. <https://cloud.google.com/discover/what-is-container-orchestration>. [Accessed 08-01-2024]. Citado na página 25.

GRAVES, L.; AMAZEEN, M. A. *Fact-Checking as Idea and Practice in Journalism*. Oxford University Press, 2019. Disponível em: <https://oxfordre.com/communication/view/10.1093/acrefore/9780190228613.001.0001/acrefore-9780190228613-e-808>. Citado na página 18.

IMMUNIZE, C. Vacinação e autismo: a maior fake news sobre vacinação. 2024. Disponível em: <https://centroimmunize.com.br/vacinacao-e-autismo-a-maior-fake-news-sobre-vacinacao/#:~:text=O%20estudo%20que%20mostrou%20a,espectro%20do%20autismo%20>. Citado na página 16.

JENNINGS, N. R. On agent-based software engineering. *Artificial Intelligence*, v. 117, n. 2, p. 277–296, 2000. Citado na página 26.

KUBERNETS. *Visão Geral — kubernetes.io*. 2023. <https://kubernetes.io/pt-br/docs/concepts/overview/>. [Accessed 08-01-2024]. Citado na página 25.

LAZER, D. M. J. et al. The science of fake news. *Science*, v. 359, n. 6380, p. 1094–1096, 2018. Disponível em: <https://www.science.org/doi/abs/10.1126/science.aao2998>. Citado na página 11.

LIMA, G. T. et al. Os impactos da mudança do perfil epidemiológico do sarampo no brasil/ the impacts of changing the epidemiologic profile of measles in brazil. *Brazilian Journal of Health Review*, v. 3, n. 3, p. 5973–5981, Jun. 2020. Disponível em: <https://ojs.brazilianjournals.com.br/ojs/index.php/BJHR/article/view/11258>. Citado na página 17.

MAWAD, D.; KENNEDY, N.; ISAAC, L. *França prende mais de 1,3 mil após protestos: o que você precisa saber sobre as manifestações*. 2023. Disponível em: <https://www.cnnbrasil.com.br/internacional/protestos-na-franca-o-que-voce-precisa-saber-sobre-as-manifestacoes/>. Citado na página 22.

NEWMAN, N. et al. *Reuters Institute Digital News Report 2022*. [S.l.]: Reuters Institute for the Study of Journalism, 2022. Citado 2 vezes nas páginas 11 e 17.

PENNYCOOK, G.; RAND, D. G. The implied truth effect: Attaching warnings to a subset of fake news stories increases perceived accuracy of stories without warnings. *Management Science*, v. 68, n. 11, p. 1553–1570, 2018. Citado na página 16.

PETER, L. Como nasceu a lenda do piloto ucraniano 'fantasma de kiev'. *BBC*, maio 2022. Disponível em: <https://www.bbc.com/portuguese/internacional-61292664>. Citado na página 41.

POSTMAN, I. *2023 State of the API Report*. 2023. Accessed: 2024-08-28. Disponível em: <https://www.postman.com/state-of-api/>. Citado na página 19.

RICHARDSON, L.; AMUNDSEN, M. *RESTful Web APIs: Services for a Changing World*. [S.l.]: O'Reilly Media, 2013. Citado na página 19.

SPARK. *SparkJava*. 2021. <https://sparkjava.com>. Accessed: April 28, 2023. Citado na página 31.

TANENBAUM, A. S. *Modern Operating Systems*. [S.l.]: Pearson, 2015. ISBN 978-0133591620. Citado 2 vezes nas páginas 18 e 19.

TWEEPY. *Tweepy*. 2021. <https://www.tweepy.org>. Accessed: April 28, 2023. Citado na página 34.

TWITTER. *Counting characters*. 2023. Disponível em: <https://developer.twitter.com/en/docs/counting-characters>. Citado na página 17.

VALINOR, R. Twitter muda de nome e agora é x. *Remessa Online Blog*, July 24 2023. Disponível em: <https://www.remessaonline.com.br/blog/twitter-muda-de-nome/#:~:text=O%20dono%20do%20Twitter%2C%20Elon,serÃą%20potencializada%20por%20inteligÃłncia%20artificial.> Citado na página 17.

VAYGHAN, L. A. et al. Deploying microservice based applications with kubernetes: Experiments and lessons learned. In: IEEE. *2018 IEEE 11th international conference on cloud computing (CLOUD)*. [S.l.], 2018. p. 970–973. Citado na página 25.

VOSOUGHI, S.; ROY, D.; ARAL, S. The spread of true and false news online. *Science*, v. 359, n. 6380, p. 1146–1151, 2018. Citado na página 16.

WASEEM, M.; LIANG, P.; SHAHIN, M. A systematic mapping study on microservices architecture in devops. *Journal of Systems and Software*, v. 170, p. 110798, 2020. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121220302053>. Citado na página 24.

WOLFF, E. *Microservices: Flexible Software Architecture*. [S.l.]: Addison-Wesley Professional, 2016. ISBN 0134650408, 9780134650409. Citado na página 24.

WOOLDRIDGE, M. *An Introduction to MultiAgent Systems*. 2. ed. [S.l.]: John Wiley & Sons, 2009. Citado na página 26.