

**CENTRO UNIVERSITÁRIO CARIOCA – UNICARIOCA**

**DOUGLAS PEREIRA PESSOA**

**OS BENEFÍCIOS DE UTILIZAR MICROFRONTEND E MICROSERVIÇOS**

**RIO DE JANEIRO**

**2023**

**DOUGLAS PEREIRA PESSOA**

**OS BENEFÍCIOS DE UTILIZAR MICROFRONTEND E MICROSERVIÇOS**

Trabalho de Conclusão de Curso apresentado  
ao Centro Universitário Carioca, como  
requisito parcial para conclusão do curso de  
Ciência da Computação.

Orientador: Prof. Anderson Fernandes P. dos Santos, DSc

**RIO DE JANEIRO**

**2023**

Dedico esse trabalho aos meus pais que sempre me apoiaram nos estudos desde a infância e não duvidaram da minha capacidade de evolução e hoje devolvo a eles o orgulho de me tornar cientista.

## **AGRADECIMENTOS**

À minha família que sempre me apoiou nos meus estudos, investiram e dedicaram tempo nos meus conhecimentos.

Ao meu orientador Prof. Anderson pelos ensinamentos nas aulas presenciais nos últimos períodos que foi de grande valia para meu crescimento como cientista da computação.

Aos meus amigos que contribuíram com ideias para levar a aplicação desenvolvida para frente e o apoio psicológico que atualmente é extremamente necessário.

E em especial os seguintes amigos e colegas de profissão:

**Juciano de Carvalho Barbosa** – Desenvolvedor sênior em Frontend

Agradeço por toda paciência para a explicação da complexa configuração dos módulos dos microfrontends.

**Mariana Cerqueira Leite** – Analista de qualidade pleno

Agradeço por todo o apoio psicológico sobre esse trabalho e agradeço também nos aprendizados para o aumento da qualidade dos meus softwares.

**Roger Tessmer Siqueira** – Desenvolvedor sênior em Java

Agradeço por todo o apoio inicial de como pensar na estrutura da aplicação levando em conta as boas práticas dos microserviços.

## **RESUMO**

A partir desse trabalho será possível compreender conceitos sobre a arquitetura de microsserviços, microfrontend e quais são seus benefícios para as empresas, equipes e desenvolvedores produzirem softwares de robustos, leves e eficientes. Demonstrando também na prática com uma aplicação feita e dividida em módulos, como elas funcionam e quais tecnologias são usadas para desenvolver aplicações nesse tipo de arquitetura. Esse trabalho visa entender os conceitos dos benefícios de se desenvolver nessa arquitetura de forma prática.

### **Palavras-chave:**

Microsserviços, Microfrontends, Modularização, Resiliência, Twitter

## **ABSTRACT**

Through this work, it will be possible to comprehend concepts about microservices and microfrontends architectures and their benefits for companies, teams, and developers to produce robust, lightweight, and efficient software. Additionally, practical demonstrations will be provided using an application built and divided into modules, showcasing how they function and the technologies used to develop applications in this type of architecture. This work aims to understand the concepts and benefits of developing in this architecture in a practical manner.

### **Keywords:**

Microservices, Microfrontends, Modularization, Resilience, Twitter

## LISTA DE ILUSTRAÇÕES

Figura 2.1 - O que são microsserviços?.....	17
Figura 2.2 - What Are Microservices? .....	18
Figura 2.3 - O que é Apache Kafka e como funciona?.....	19
Figura 2.4 - Mensageria. Cada vez mais utilizado e exigido no TI.....	20
Figura 2.5 - O que é micro front end? .....	21
Figura 3.1 - Diagrama de caso de uso .....	24
Figura 3.2 - Diagrama de Sequência .....	26
Figura 3.3 - Modelagem banco de dados – Módulo Mensagens .....	28
Figura 3.4 - Modelagem banco de dados – Módulo Pesquisa .....	29
Figura 3.5 - Modelagem banco de dados – Módulo Notificações .....	30
Figura 4.1 - Microsserviço de mensagens – rota para adicionar mensagem.....	32
Figura 4.2 - Microsserviço de mensagens – rota para listar mensagens.....	33
Figura 4.3 - Microsserviço de mensagens – rota obter mensagem única.....	34
Figura 4.4 - Microsserviço de pesquisa – salvar nome de usuário .....	35
Figura 4.5 - Microsserviço de pesquisa – rota de pesquisa por usuário .....	36
Figura 4.6 - Microsserviço de notificações – rota para notificar usuário no Twitter.....	37
Figura 4.7 - Microfrontend módulo Home – Página inicial .....	38
Figura 4.8 - Microfrontend módulo Home – Página de formulário .....	39
Figura 4.9 - Microfrontend módulo Perfil – Página de listagem de mensagens.....	39
Figura 4.10 - Microfrontend módulo Perfil – Página de mensagem .....	40
Figura 4.11 - Notificação enviada ao Twitter .....	41

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO.....</b>	<b>11</b>
1.1	Objetivo Geral .....	11
1.2	Organização .....	11
<b>2</b>	<b>CONCEITOS DA ARQUITETURA SOFTWARE PARA MICROSERVIÇOS E MICROFRONTEND E ATUAL MERCADO DE APLICAÇÕES DE MENSAGENS ANONIMAS.....</b>	<b>13</b>
<b>2.1</b>	<b>MICROSSERVIÇOS.....</b>	<b>13</b>
2.1.1	Modularização .....	13
2.1.2	Flexibilidade.....	14
2.1.3	Desenvolvimento em sistemas legados .....	14
2.1.4	Time-To-Market (tempo para entrada no mercado).....	14
2.1.5	Tecnologias.....	14
2.1.6	Complexidade.....	15
2.1.7	Refatoração .....	15
2.1.8	Sobrecarga .....	15
<b>2.2</b>	<b>SOA – Service Oriented Architecture .....</b>	<b>15</b>
<b>2.3</b>	<b>MICROFRONTEND.....</b>	<b>16</b>
<b>2.4</b>	<b>COMO FUNCIONA? .....</b>	<b>17</b>
2.4.1	Microserviços .....	17
2.4.2	Microfrontend.....	21
<b>2.5</b>	<b>MERCADO DE MENSAGENS ANÔNIMAS.....</b>	<b>22</b>
2.5.1	NGL .....	23
2.5.2	Tellonym.....	23
<b>3</b>	<b>MODELAGENS .....</b>	<b>24</b>
<b>3.1</b>	<b>Diagrama de caso de uso .....</b>	<b>24</b>
3.1.1	Descrição do diagrama de caso de uso .....	25
<b>3.2</b>	<b>Diagrama de Sequências .....</b>	<b>26</b>
3.2.1	Envio de mensagem.....	26
3.2.2	Pesquisa de conta mencionada.....	27
3.2.3	Carregamento das mensagens .....	27
<b>3.3</b>	<b>Modelo Entidade Relacionamento .....</b>	<b>27</b>
3.3.1	Modelagem para o microserviço de mensagens .....	28
3.3.2	Modelagem para microserviço de pesquisa .....	29
3.3.3	Modelagem para microserviço de notificações.....	30
<b>4</b>	<b>DESENVOLVIMENTO .....</b>	<b>31</b>
<b>4.1</b>	<b>Backend – Microserviços.....</b>	<b>31</b>



4.1.1	Microserviço de mensagens .....	31
4.1.2	Microserviço de pesquisa .....	35
4.1.3	Microserviço de notificação.....	37
<b>4.2</b>	<b><i>Frontend</i> – Microfrontend .....</b>	<b>38</b>
4.2.1	Microfrontend – <i>Home</i> .....	38
4.2.2	Microfrontend – Perfil do Usuário .....	39
<b>4.3</b>	<b>Notificação no Twitter .....</b>	<b>41</b>
<b>5</b>	<b><i>CONCLUSÃO</i> .....</b>	<b>42</b>
	<b><i>REFERÊNCIAS</i>.....</b>	<b>43</b>

## LISTA DE ABREVIATURAS E SIMBOLOS

### ABREVIATURAS

API	-	<i>Application Programming Interface</i>
CSS-in-JS	-	<i>CSS in JavaScript</i>
DB	-	<i>Database</i>
HTML	-	<i>Hypertext Markup Language</i>
HTTP	-	<i>Hypertext Transfer Protocol</i>
JSON	-	<i>JavaScript Object Notation</i>
MVP	-	<i>Minimum Viable Product</i>
NGL	-	<i>Not Gonna Lie</i>
NoSQL	-	<i>Non-relational database</i>
REST	-	<i>Representational State Transfer</i>
SGBD	-	<i>Sistema de gerenciamento de banco de dados</i>
SOA	-	<i>Service Oriented Architecture</i>
XML	-	<i>eXtensible Markup Language</i>

## 1 INTRODUÇÃO

Com a evolução tecnológica as aplicações estão cada vez mais robustas, leves e eficientes. Nos tempos atuais as aplicações grandes chamadas monolíticas podem gerar um custo maior para manter ativa, tanto financeiramente quanto de produtividade para quem desenvolve. Para isso em 2009 surgiu o conceito de microsserviços, uma aplicação maior é dividida em diversos pedaços menores, chamados de módulos, esses módulos trabalham de maneiras independentes sem que impactem uns aos outros, caso porventura ocorra alguma falha entre algum deles. A sua popularidade cresceu por volta de 2014 e 2015 quando o desenvolvimento de software baseado em nuvem se tornou atraente, hoje é amplamente utilizado por grandes empresas de software.

Embarcando nesse conceito, os microfrontends vieram para resolver o mesmo problema a diferença é para o *frontend* está na modularização das interfaces do usuário. Para o *frontend* é necessária uma interface hospedeira englobe os outros microfrontends para ter em tela os módulos que são desenvolvidos separadamente.

### 1.1 Objetivo Geral

O objetivo deste trabalho é entender os benefícios do conceito sobre a arquitetura de microsserviços e microfrontends, as vantagens e desafios de se desenvolver com essa tecnologia e ao final executar uma aplicação de maneira prática sobre o que será abordado. A aplicação tem como proposta de um usuário qualquer da internet, envie mensagens em anônimo para outra pessoa, mencionando o nome do usuário dessa do *Instagram* ou *Twitter*, as mensagens serão salvas no banco de dados da aplicação e caso seja a mensagem de destino seja um usuário do *Twitter* o mesmo será notificado via uma conta robô, mencionando o usuário e informando que há uma nova mensagem na plataforma.

### 1.2 Organização

O presente trabalho está organizado em 5 capítulos.

No 2º capítulo será abordado os conceitos da arquitetura de software para os microsserviços e microfrontends.

No 3º capítulo será possível entender a modelagem de dados e modelagem do fluxo para microsserviços.

No 4º capítulo encontra-se como a aplicação foi projetada e desenvolvida para o funcionamento eficiente dos microsserviços

E no último capítulo foi descrito a conclusão de todo o trabalho.

## **2 CONCEITOS DA ARQUITETURA SOFTWARE PARA MICROSERVIÇOS E MICROFRONTEND E ATUAL MERCADO DE APLICAÇÕES DE MENSAGENS ANONIMAS.**

### **2.1 MICROSERVIÇOS**

O conceito arquitetônico sobre os microserviços é caracterizada pela modularização de uma arquitetura monolítica. Cada serviço possui uma funcionalidade específica dentro do sistema completo e essa abordagem de arquitetura é pensada, desenvolvida e distribuída de forma independente, deste modo cada serviço pode ser desenvolvido com tecnologias diferentes.

Um software baseado em arquitetura monolítica tem todas as suas funcionalidades e componentes em um sistema único, por essa razão esse tipo de arquitetura pode gerar um forte acoplamento entre suas funções. O acoplamento entre as funcionalidades do sistema pode resultar em dificuldades de manutenção, evolução e dependendo do tamanho do projeto as suas atualizações poderão ser mais complexas e arriscadas. A escalabilidade e o ciclo de desenvolvimento também são afetados por essa arquitetura, onde a escalabilidade é geralmente limitada podendo resultar em desperdício de recursos e o ciclo de desenvolvimento é feita de forma sequencial, quando qualquer alteração no sistema deverá passar por testes, compilação e implantação seguindo métricas de qualidade de software, assim gerando um desenvolvimento mais demorado.

Diferentemente da arquitetura monolítica, os microserviços trouxeram uma abordagem para um desenvolvimento de software mais benéfica.

#### **2.1.1 Modularização**

Uma das maiores vantagens do desenvolvimento de microserviços é a modularização do software. Como cada serviço é único, obtém-se a independência de desenvolvimento onde diferentes equipes podem trabalhar em projetos distintos sem afetar os trabalhos de outros desenvolvedores. Com a modularização dos microserviços, é possível escalar cada módulo independente (escalabilidade modular) gerando uma redução de custos para a empresa que usufrui dessa arquitetura e quando se trata de falhas, um serviço não impacta nas funcionalidades dos demais serviços resultando em maior resiliência.

### **2.1.2 Flexibilidade**

Os microsserviços podem ser substituídos ou realocados de maneira mais simples que um software baseado na arquitetura monolítica. Quando um microsserviço tem a sua tecnologia limitada para determinado problema, ele poderá ser substituído facilmente por outro microsserviço que tenha a mesma interface.

### **2.1.3 Desenvolvimento em sistemas legados**

Trabalhar em novos projetos permite o desenvolvimento seja mais fácil e ágil, porém em sistemas legados baseados na arquitetura monolítica pode comprometer a produtividade dos desenvolvedores da empresa principalmente aqueles colaboradores que acabaram de ser contratados gerando muita demanda de tempo para o aprendizado do funcionamento do sistema. Na arquitetura dos microsserviços o tempo gasto para entender como o serviço funciona é bem menor, partindo do conceito que os microsserviços têm funcionalidades bem específicas. Os microsserviços podem funcionar em paralelo ao software construído em arquitetura monolítica, realizando tarefas que o sistema legado realmente precisa de substituição.

### **2.1.4 Time-To-Market (tempo para entrada no mercado)**

Um microsserviço pode ser levado a produção em um tempo consideravelmente reduzido, assim empresas alocam diferentes equipes para desenvolver diferentes funcionalidades do sistema e então produzirem os chamados MVP's. A entrega contínua é extremamente utilizada nesses casos, com as implantações independentes dos serviços, as correções são lançadas em um ritmo mais rápido melhorando a qualidade do produto para mercado.

### **2.1.5 Tecnologias**

Uma das grandes vantagens de trabalhar com microsserviços é a escolha da tecnologia, não há limitações para a escolha da linguagem, *framework* ou bibliotecas para a resolução dos problemas, uma vez que, uma tecnologia não impacta diretamente nos outros serviços. Caso a empresa deseje introduzir uma nova tecnologia em seus serviços,

o risco é menor porque fica restrita apenas aqueles serviços onde a ferramenta foi implementada, reduzindo os problemas com falhas e dando possibilidade de tomada de decisões independentes. Outros grandes benefícios que se obtém a partir desses serviços com diferentes tecnologias são diferentes vagas de trabalho, onde esses colaboradores precisam ser especializados na determinada tecnologia.

Os microsserviços também tem seus desafios, abaixo uma lista deles.

#### **2.1.6 Complexidade**

O desenvolvimento em microsserviços requer um estudo maior sobre os casos, o domínio sobre a arquitetura é valioso onde o desenvolvedor tem o desafio de fazer a composição dos serviços funcionarem de maneira eficiente, levando em conta que cada serviço tem seu próprio banco de dados e infraestrutura adequada para cada aplicação.

#### **2.1.7 Refatoração**

Uma das consequências geradas pela modularização do software é a dificuldade de refatoração. Isso pode exigir do desenvolvedor uma significativa experiência para poder reestruturar o código já existente, identificar os limites e criar testes baseado no novo serviço.

#### **2.1.8 Sobrecarga**

A comunicação entre serviços é realizada através de chamadas de *API* ou mensagens assíncronas. A vantagem de modularizar os serviços, gera a possível desvantagem de sobrecarga, aumentando a latência e o tempo de resposta entre serviços.

### **2.2 SOA – *Service Oriented Architecture***

Essa arquitetura tem conceitos bem parecidos com os microsserviços, visa criar softwares flexíveis, escaláveis e reutilizáveis. A SOA tem o conceito de decomposição de um sistema em serviços autônomos que podem ser combinados e reutilizados para atender diferentes necessidades de negócios. Como por exemplo: muitos sistemas

dependem de autenticação, a SOA vem para desacoplar esse serviço de autenticação e permitindo que ele seja independente para a reutilização de em outros negócios.

### 2.3 MICROFRONTEND

Para este ano de 2023, microfrontend é um padrão arquitetural ainda muito recente na comunidade de desenvolvimento, ele estende o conceito sobre os microsserviços que, supracitado, decompõe partes da aplicação em módulos, nesse caso, decompõe em módulos a interface do usuário.

Uma aplicação monolítica de *frontend* é desenvolvida pela mesma equipe, desde o levantamento de requisitos até a escolha do *framework* ou tecnologia e o desenvolvimento é feito no mesmo diretório, ou seja, todas as páginas, componentes, estilos e funcionalidades contidos no mesmo código-base.

Na arquitetura microfrontend aborda-se a modularização dos componentes e interfaces do usuário, nesse conceito, pode ser trabalhado por diferentes equipes e implantados separadamente, usando tecnologias diferentes. Assim como nos microsserviços, o microfrontend se beneficia das vantagens já citadas, como: escalabilidade modular; reutilização; isolamento e resiliência.

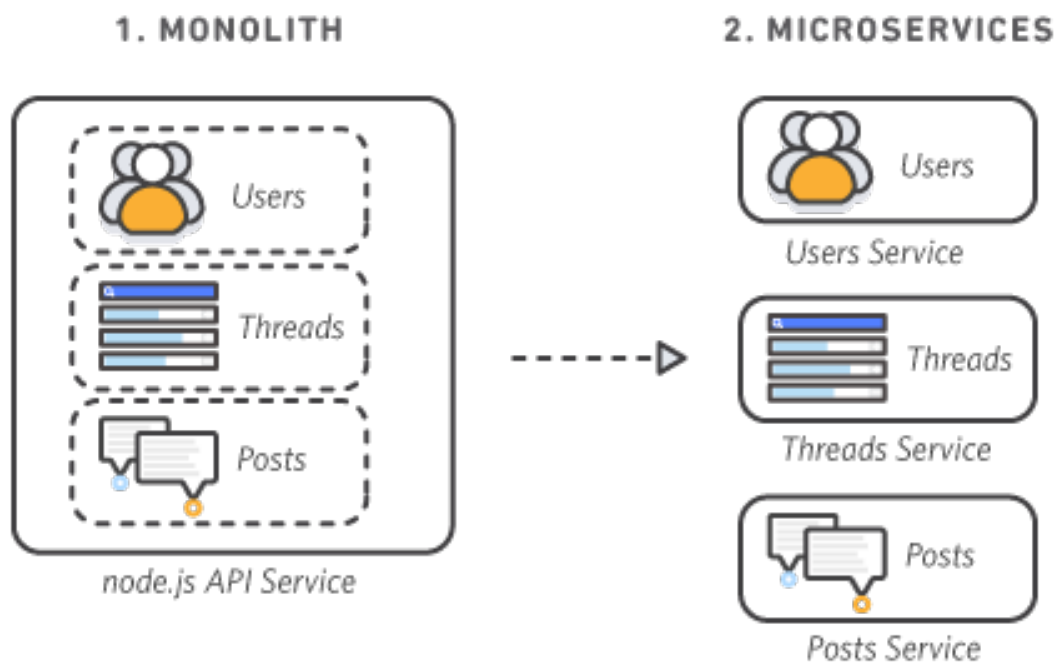


## 2.4 COMO FUNCIONA?

### 2.4.1 Microserviços

Com o conceito de modularização dos serviços é dividir a aplicação em partes menores e deixá-las independentes, leves e escaláveis, abaixo um exemplo de uma aplicação monolítica dividida em serviços de funcionalidade única.

*Figura 2.1 - O que são microserviços?*

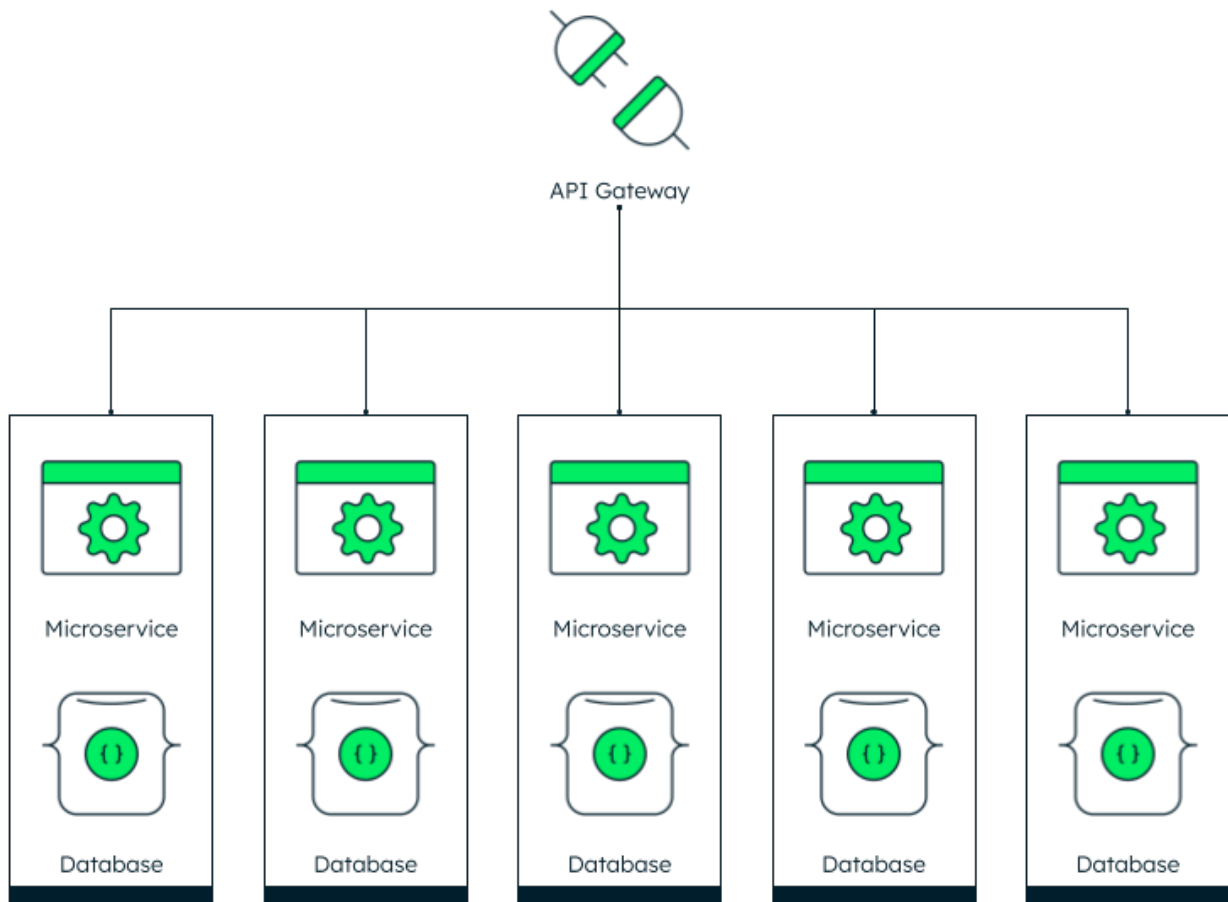


Fonte: AWS, 2023

Os módulos podem ser chamados independentemente por outros clientes ou internamente em outros serviços, essas chamadas podem ser via *API RESTful* ou por mensagens assíncronas, usando tecnologias como *RabbitMQ*, *IBM MQ* ou *Apache Kafka*. O desenvolvedor tem o desafio de pensar na tecnologia mais adequada para a conversa entre os microserviços de forma eficiente.

Para chamadas *API RESTful* vindas do cliente, geralmente é aplicado o *API Gateway* que é um componente central nas arquiteturas de microsserviços. Ele é uma camada de abstração facilitando o roteamento, segurança e gerenciamento de solicitações.

Figura 2.2 - What Are Microservices?



Fonte: A Full Guide | MongoDB, 2023

#### 2.4.1.1 *API RESTful*

*API RESTful* (ou apenas *API REST*) é uma interface de programação para aplicações que está baseada na arquitetura *REST*, é um conjunto de diretrizes e padrões que é amplamente utilizado nas aplicações nos tempos modernos. As operações são realizadas através de requisições *HTTP* e seus diferentes tipos de formatos como *JSON*, *XML* ou apenas *HTML*. Uma *API RESTful* é projetada para ser *stateless*, ou seja, toda nova

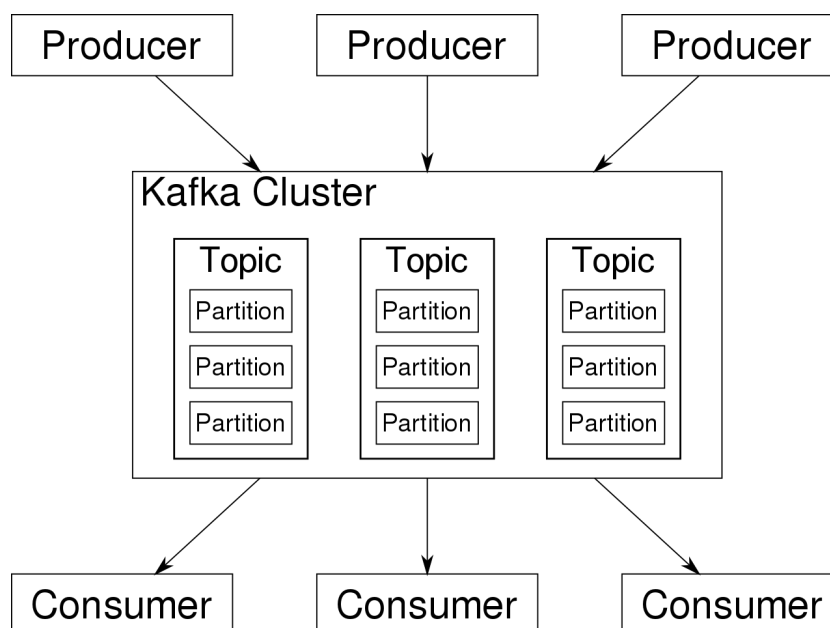
solicitação é tratada de forma independente, sem levar em conta os estados anteriores de outras requisições.

Por outro lado, trabalhando com a comunicação com mensagens assíncronas, ganha-se uma maior flexibilidade, escalabilidade, durabilidade e tolerância a falhas. Uma vez que, esses serviços de mensageria configurados de uma determinada maneira, recebem mensagens dos serviços e encaminham para outros e caso o destino não esteja funcionando, ele mantém guardado até o serviço ficar ativo novamente e consumir essa mensagem. Abaixo, uma breve explicação de como funciona o *Apache Kafka* e o *RabbitMQ*.

#### 2.4.1.2 Apache Kafka

*Apache Kafka* (ou apenas *Kafka*) é um serviço de mensageria de código aberto desenvolvido e mantido pela *Apache Software Foundation*, construído nas linguagens de programação Java e Scala. O *Kafka* funciona através do modelo publicação / assinatura, onde as mensagens são publicadas em tópicos e os consumidores (nosso caso, os microserviços) se inscrevem nesses tópicos para receber as mensagens.

*Figura 2.3 - O que é Apache Kafka e como funciona?*

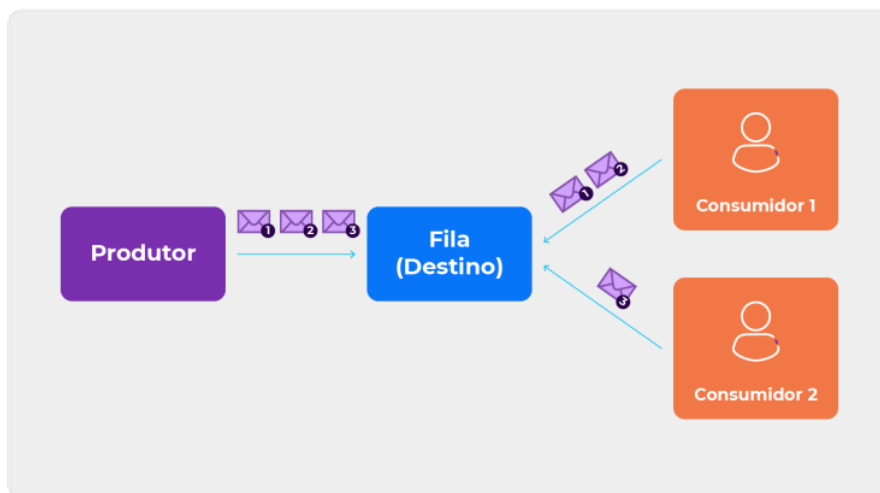


Fonte: Luby Software do seu jeito, 2023

### 2.4.1.3 RabbitMQ

O *RabbitMQ* desenvolvido em na linguagem de programação *Erlang*, mantido pela *VMware*. A sua proposta é parecida com do *Kafka*, a diferença está em como ele lida com as mensagens que utiliza o modelo de filas, onde as mensagens são enviadas para filas específicas e os consumidores se conectam a essas filas para receber as mensagens.

*Figura 2.4 - Mensageria. Cada vez mais utilizado e exigido no TI.*

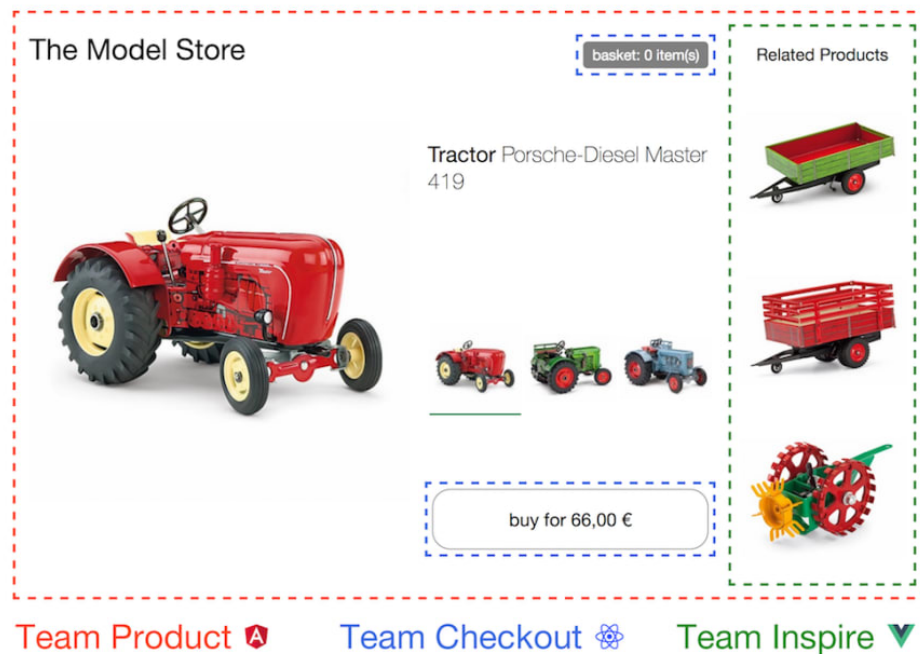


Fonte: LinkedIn - Luciano Rocha

### 2.4.2 Microfrontend

Diferente do microserviços, o microfrontend necessita de uma aplicação hospedeira que consuma os outros módulos para entregar para o cliente a aplicação funcional como um todo.

*Figura 2.5 - O que é micro front end?*



Fonte: João Pedro Resende | dev.to

Acima, um exemplo na figura, o time de inspirações desenvolve a interface que entrega ao usuário da loja virtual produtos relacionados, o time de checkout desenvolve a interface que resolve toda a questão financeira do que está sendo vendido e por fim o time de produto que desenvolve a aplicação hospedeira, que consome todos os outros componentes produzidos por outros times.

#### 2.4.2.1 - Module Federation

O *Module Federation* é uma ferramenta introduzida no *Webpack 5* (empacotador de módulos de código-fonte aberto para *JavaScript*), ele traz o benefício de carregamento sob demanda fazendo com que a aplicação seja leve para ser carregada no lado do cliente. Essa ferramenta utiliza a tecnologia de carregamento assíncrono para buscar e

compartilhar os módulos necessários em tempo de execução permitindo assim que uma aplicação carregue módulos remotamente de outro software e utilizando como se fossem seus próprios componentes.

## 2.5 MERCADO DE MENSAGENS ANÔNIMAS

Os aplicativos de mensagens anônimas estão se tornando cada vez mais populares entre os usuários da internet. Eles oferecem aos usuários a viabilidade de enviar mensagens em anonimato para outros usuários da rede, ou seja, sua identidade não será revelada para outros usuários.

A utilidade desses aplicativos está no envio da mensagem como de feedbacks, críticas construtivas, declarações e entre outros e de forma honesta sem ter medo de represálias. Todavia, os aplicativos também podem ser usados para disseminar mensagens de ódio, *bullying* ou informações falsas, para isso algumas dessas aplicações contam com funcionalidades para bloquear esse tipo de conteúdo. Temos como exemplo o aplicativo *NGL* que tem como funcionalidade a inteligência artificial para poder analisar os conteúdos publicados na plataforma. Todos os aplicativos deixam claro que o anonimato não deixará impune aqueles que violarem suas regras e diretrizes, isso pode contar com o banimento do usuário na plataforma ou enquadrar como crime, podendo rastrear sua localidade e obter informações necessárias para o caso.

Cada aplicativo tem suas particularidades e funcionalidade únicas, mas todos têm a característica fundamental do anonimato. Porém, para o usuário receber mensagens é obrigatório que ele crie uma conta com e-mail e senha ou acessando o aplicativo via contas Google ou Twitter, há alguns também que requer apenas a criação do perfil, inserindo apenas um nome de usuário único.

A forma de como o usuário responde as mensagens nos aplicativos, eles têm formatos parecidos. Alguns são diretamente feitos para o Instagram publicando uma imagem no stories com a pergunta e a resposta, outros o usuário pode responder dentro do próprio aplicativo. O *TypeAnoni.Me* permitirá que o usuário responda de ambas as formas, assim facilitando uma maior interação entre seus usuários. Abaixo, exemplos de aplicações do ramo.

### **2.5.1 NGL**

O *NGL* é um aplicativo construído para *iOS* (sistema operacional da Apple) e *Android* que visa receber mensagens em anonimato, mas requer que o usuário crie um perfil para receber esses feedbacks e o usuário pode postar sua resposta nos *Stories* do *Instagram* gerando mais engajamento.

### **2.5.2 Tellonym**

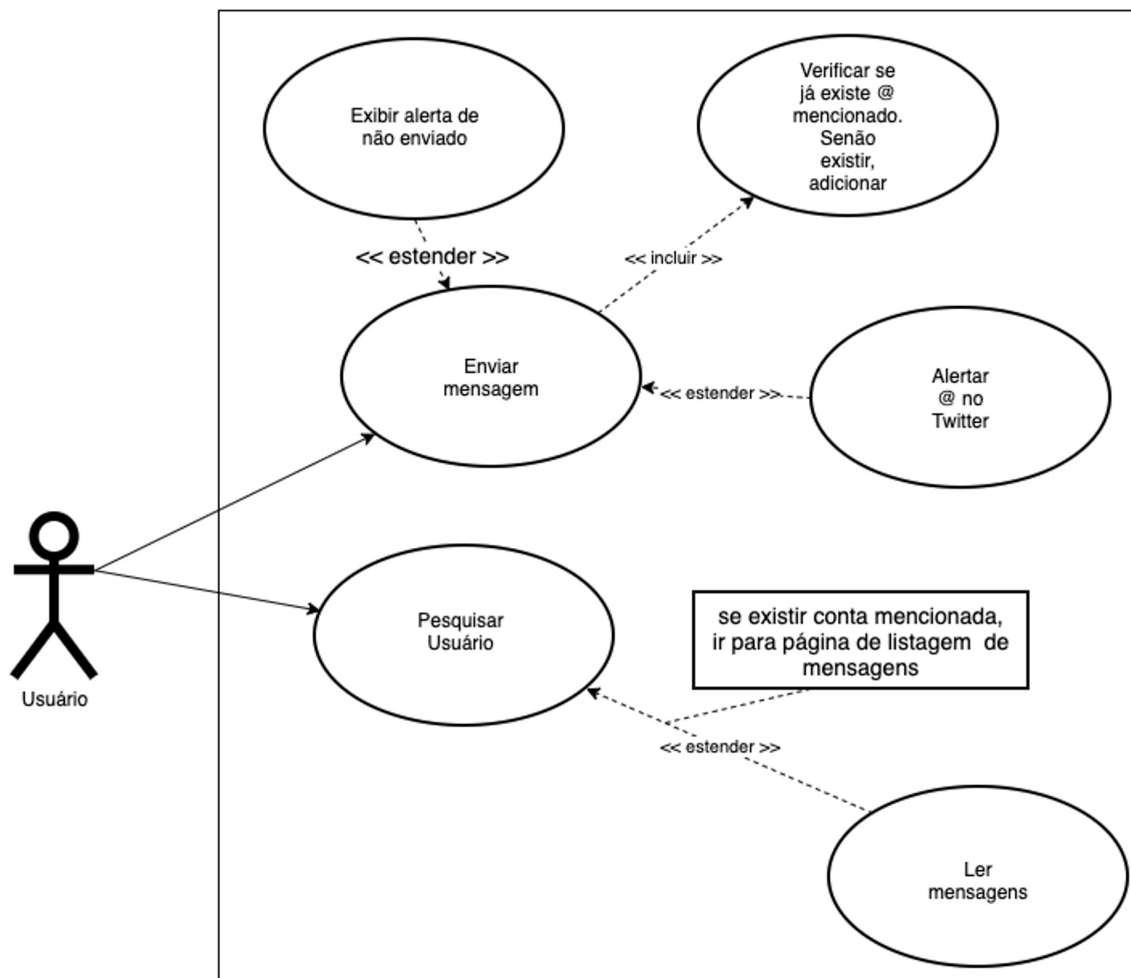
O *Tellonym* é uma aplicação construída para *iOS* e *Android* que também recebe mensagens anônimas, como diferencial do *NGL* esse aplicativo é uma rede social, assim seus usuários podem seguir uns aos outros e enviar mensagens, mesmo com perfil criado, em anonimato.

### 3 MODELAGENS

#### 3.1 Diagrama de caso de uso

Para o desenvolvimento da aplicação, foi criada uma modelagem de caso de uso. Abaixo a figura com a modelagem representada

*Figura 3.1 - Diagrama de caso de uso*



Autor: autoria própria.



### **3.1.1 Descrição do diagrama de caso de uso**

#### **3.1.1.1 Atores**

- Usuário
- Sistema

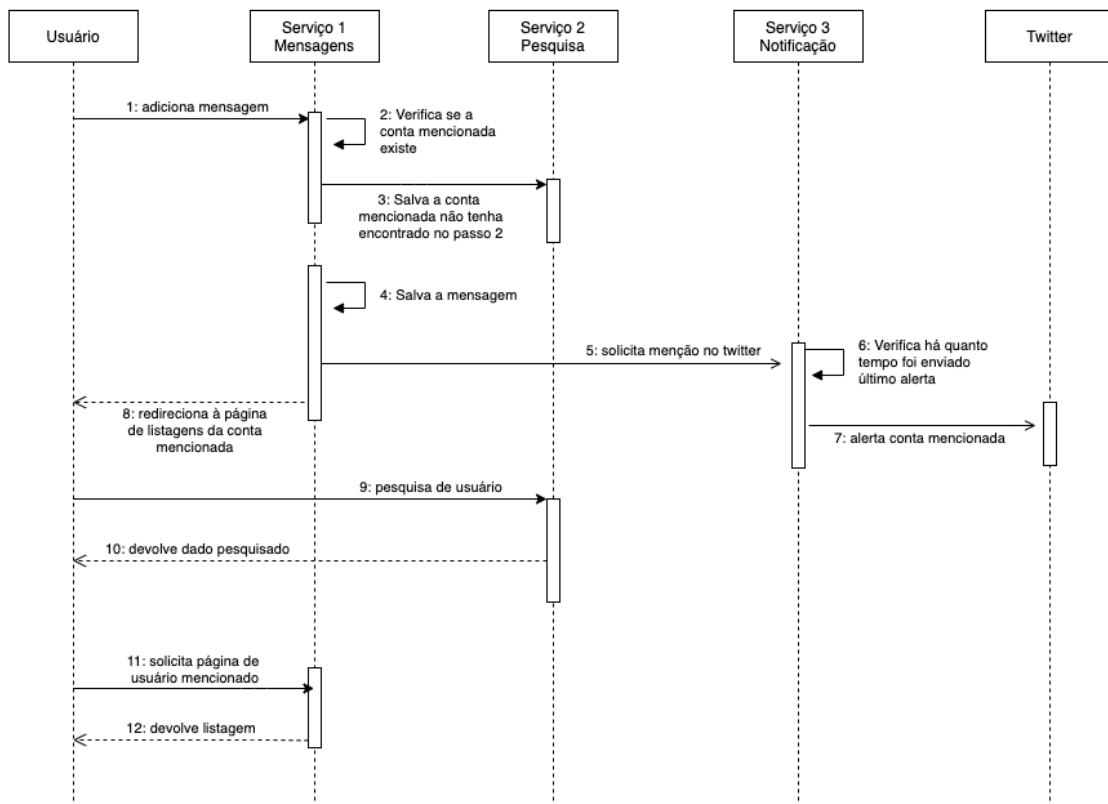
#### **3.1.1.2 Caso de Uso**

- Enviar mensagem.
- Exibir alerta de mensagem não enviada.
- Verificar se já existe conta mencionada, senão existir salvar no banco de dados.
- Alertar conta mencionada no Twitter.
- Pesquisar usuário.
- Ler mensagens.

### 3.2 Diagrama de Sequências

Para compreender melhor o fluxo das operações do sistema, abaixo uma figura representativa das sequências realizadas pelo software.

Figura 3.2 - Diagrama de Sequência



Fonte: autoria própria.

Abaixo, a descrição do fluxo das operações.

#### 3.2.1 Envio de mensagem

- Usuário abre a plataforma e adiciona no formulário a mensagem desejada e envia para o microserviço de mensagens.
- Sistema verifica se a conta mencionada pelo usuário já existe, caso não exista, o microserviço de mensagens chama o microserviço de pesquisa para salvar o nome do usuário mencionado.

- Salva a mensagem e chama o microserviço de notificação no Twitter.
- O microserviço de notificação do Twitter verifica se aquela conta foi mencionada foi notificada na rede social dentro de 24 horas, caso tenha sido acima desse tempo, será notificado novamente, caso contrário será ignorado.
- Usuário é redirecionado para a página da conta mencionada, listando todas as outras mensagens.

### **3.2.2 Pesquisa de conta mencionada**

- Usuário realiza uma pesquisa sobre uma conta mencionada
- O microserviço de pesquisas devolve a usuário da conta, se existir.

### **3.2.3 Carregamento das mensagens**

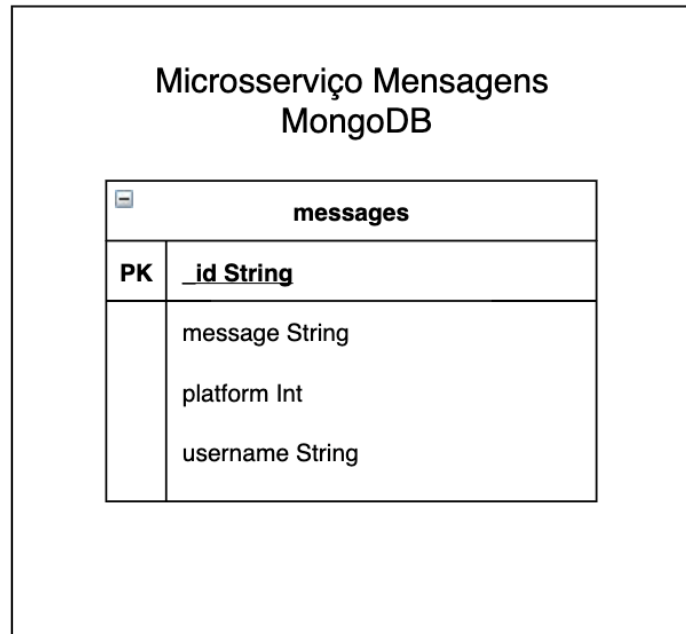
- Usuário abre a página das mensagens da conta pesquisada
- O microserviço de mensagens devolve uma lista de mensagens caso exista.

## **3.3 Modelo Entidade Relacionamento**

Para a modelagem do banco de dados, foi projetado com banco de dados *MongoDB*, e também será utilizado o *ElasticSearch*: um mecanismo de busca e análise de dados em tempo real.

### 3.3.1 Modelagem para o microserviço de mensagens

Figura 3.3 - Modelagem banco de dados – Módulo Mensagens

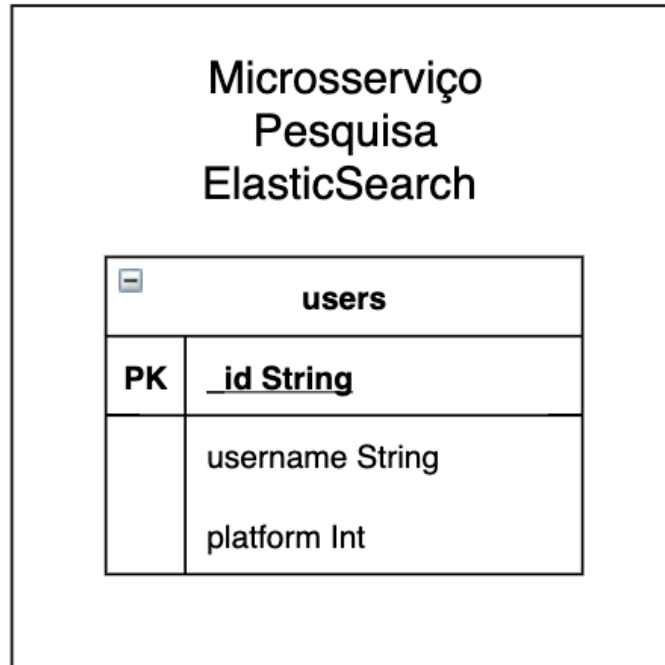


Fonte: autoria própria.

- *\_id* – Chave primária em texto, gerado aleatoriamente pelo SGBD do *MongoDB*.
- *Message* – propriedade em texto, que referencia a mensagem enviada pelo usuário
- *Platform* – propriedade em inteiro, que referencia para qual plataforma o usuário deseja mencionar. Para 0, Instagram. Para 1, Twitter.
- *Username* – propriedade em texto, que referencia o nome de usuário que será mencionado na plataforma.

### 3.3.2 Modelagem para microsserviço de pesquisa

Figura 3.4 - Modelagem banco de dados – Módulo Pesquisa

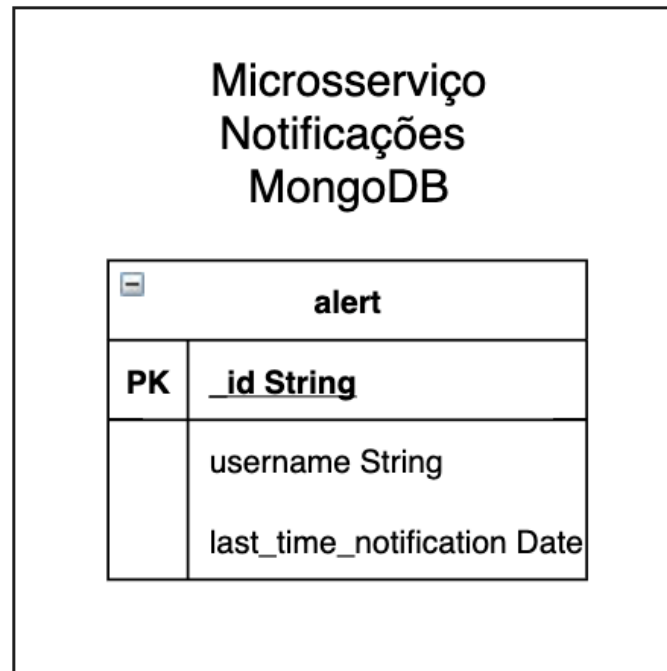


Fonte: autoria própria

- *\_id* – Chave primária em texto, gerado aleatoriamente pelo mecanismo do *ElasticSearch*.
- *Platform* – propriedade em inteiro, que referencia para qual plataforma o usuário deseja mencionar. Para 0, Instagram. Para 1, Twitter.
- *Username* – propriedade em texto, que referencia o nome de usuário que será mencionado na plataforma.

### 3.3.3 Modelagem para microsserviço de notificações

Figura 3.5 - Modelagem banco de dados – Módulo Notificações



Fonte: autoria própria

- *\_id* – Chave primária em texto, gerado aleatoriamente pelo SGBD do *MongoDB*.
- *Username* – propriedade em texto, que referencia o nome de usuário que será mencionado na plataforma.
- *Last Time Notification* – Propriedade em data, que referencia a última vez que o usuário foi notificado no Twitter.

## 4 DESENVOLVIMENTO

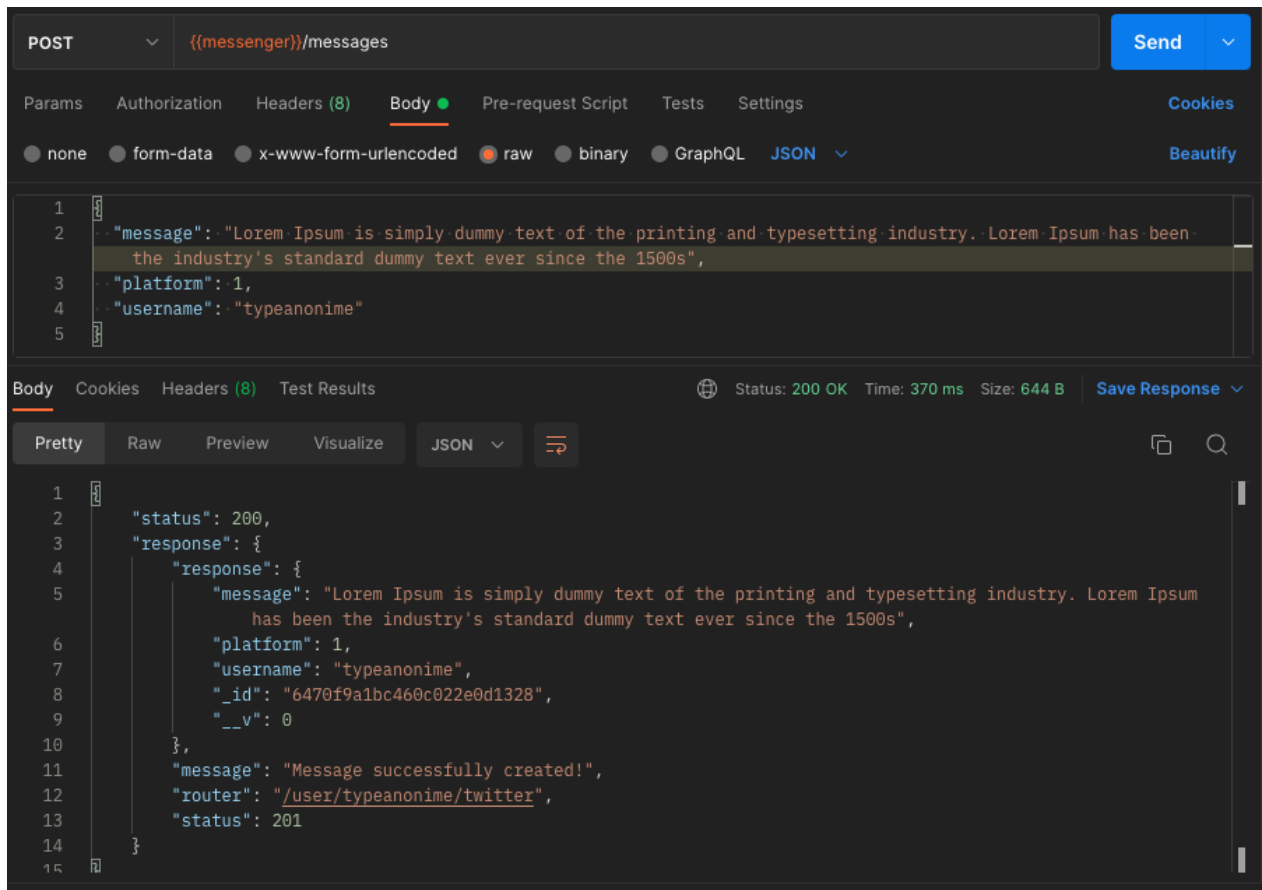
O *TypeAnoni.Me* foi projetado e pensado de forma simples de demonstração de como os microsserviços e microfrontend funcionam na prática. Essa aplicação foi construída com 3 microsserviços e 2 microfrontend (sem contar o hospedeiro). A linguagem utilizada para o desenvolvimento desses softwares foi o *TypeScript*, uma linguagem moderna e evolutiva do *JavaScript*. Para banco de dados foi utilizado um banco *NoSQL*, o *MongoDB* – um banco de dados não relacional. A motivação para o uso desse banco está empregada no seu não-relacionamento pelo fato que os microsserviços tem apenas uma entidade cada. Todas as requisições entre os microsserviços e microfrontend foram feitas via *API RESTful*.

### 4.1 Backend – Microsserviços

#### 4.1.1 Microsserviço de mensagens

Esse módulo foi desenvolvido com o *framework* chamado *Express*, ele oferece recursos facilitados para desenvolvimento de *API RESTful* e sua funcionalidade visa criar e ler mensagens enviadas pelos usuários. Abaixo, as três rotas criadas para esse microsserviço.

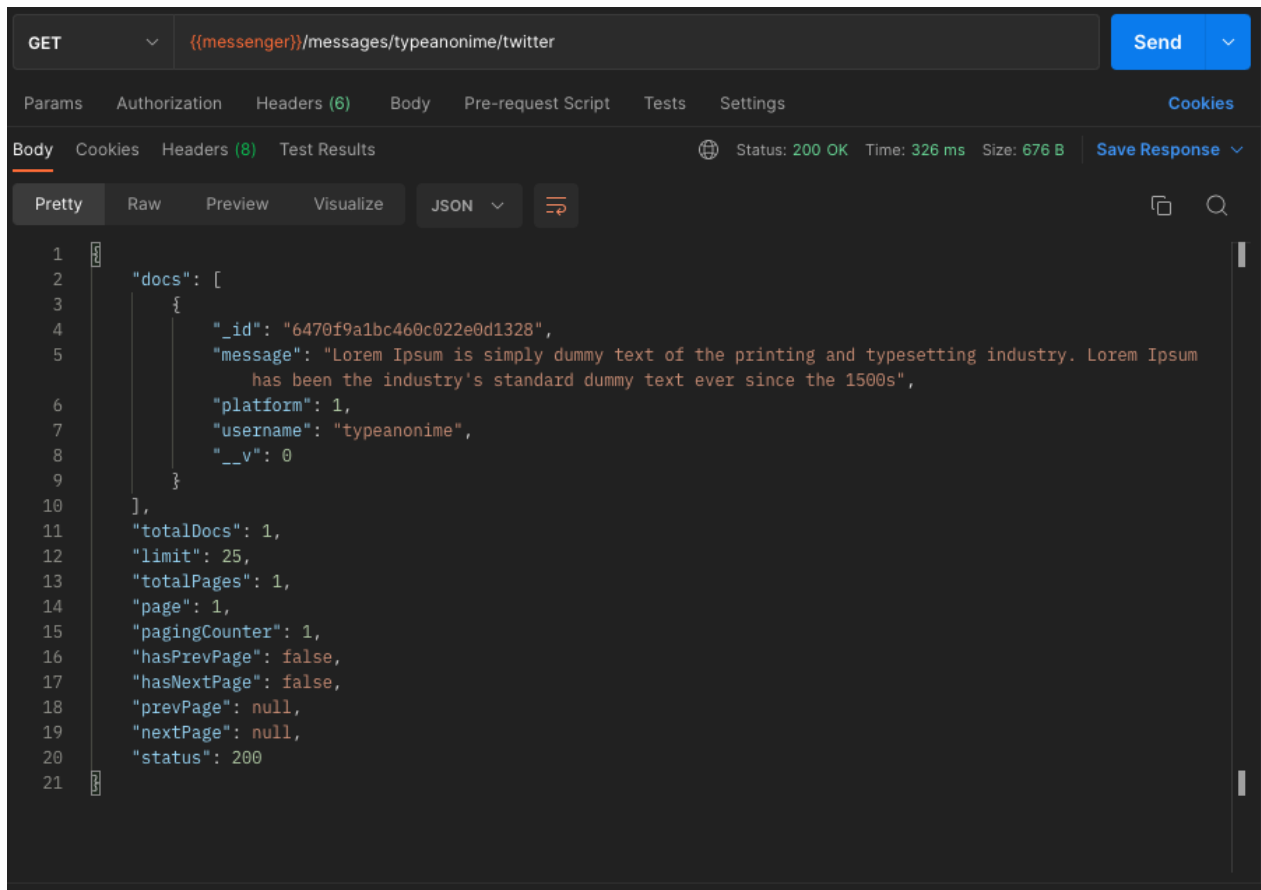
Figura 4.1 - Microserviço de mensagens – rota para adicionar mensagem



Fonte: autoria própria

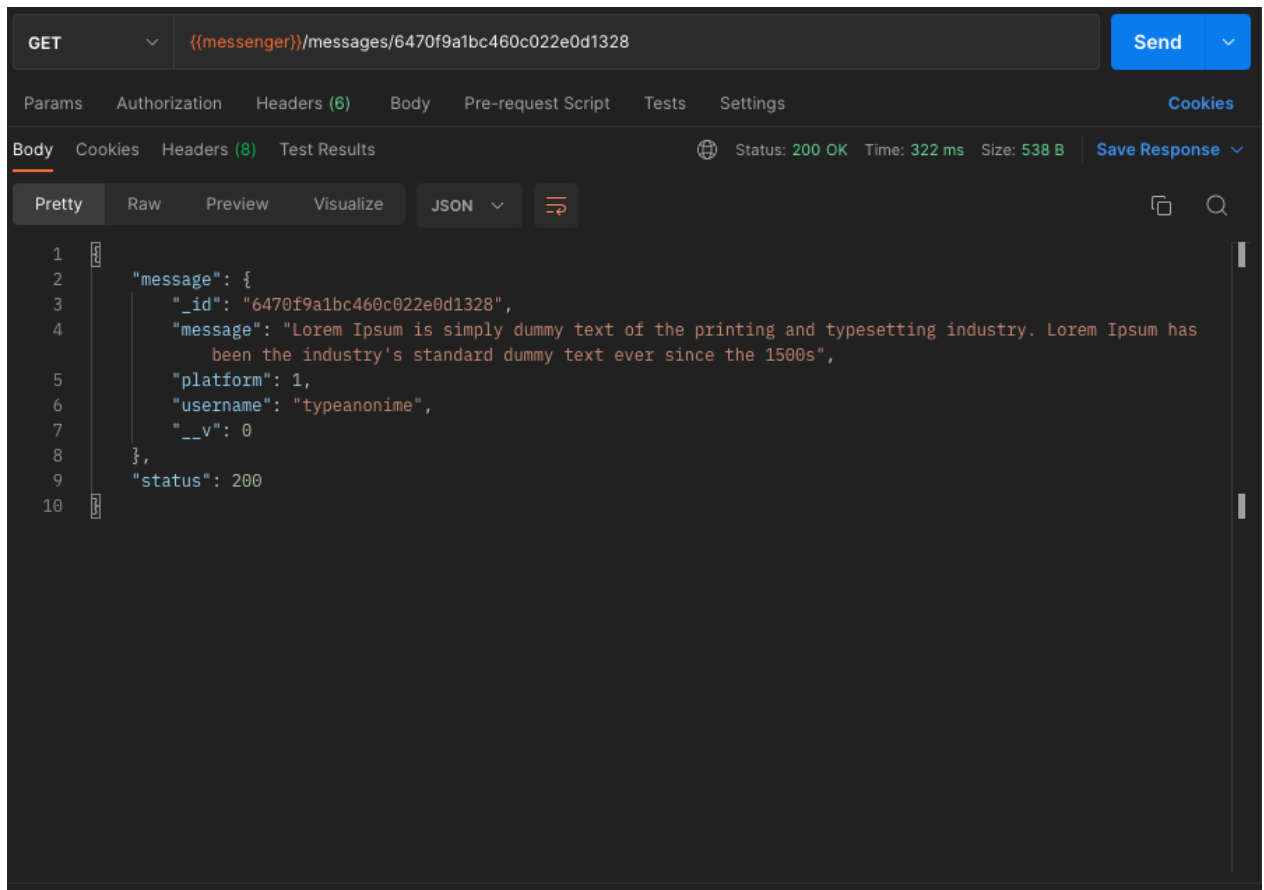


Figura 4.2 - Microsserviço de mensagens – rota para listar mensagens



Fonte: autoria própria

*Figura 4.3 - Microserviço de mensagens – rota obter mensagem única*

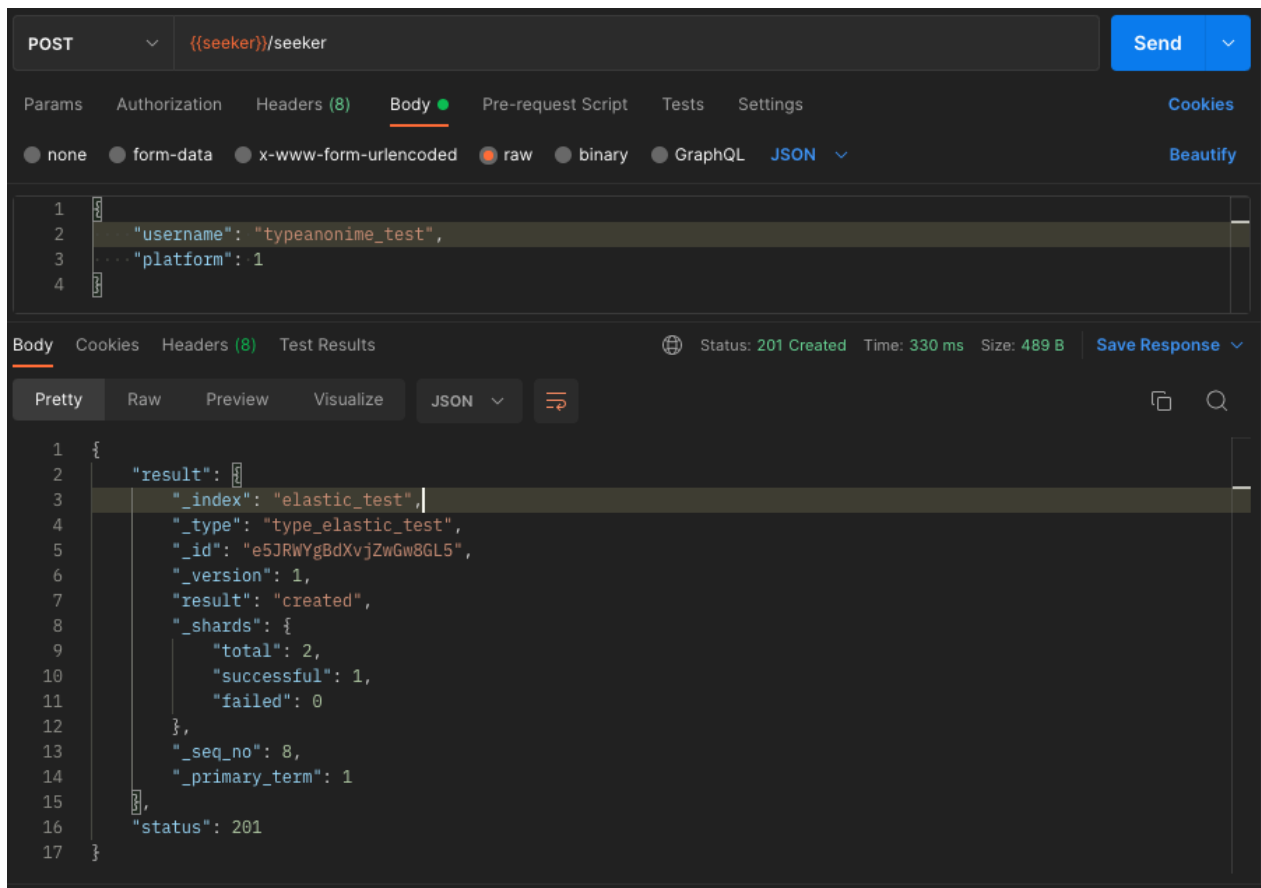


Fonte: autoria própria

#### 4.1.2 Microserviço de pesquisa

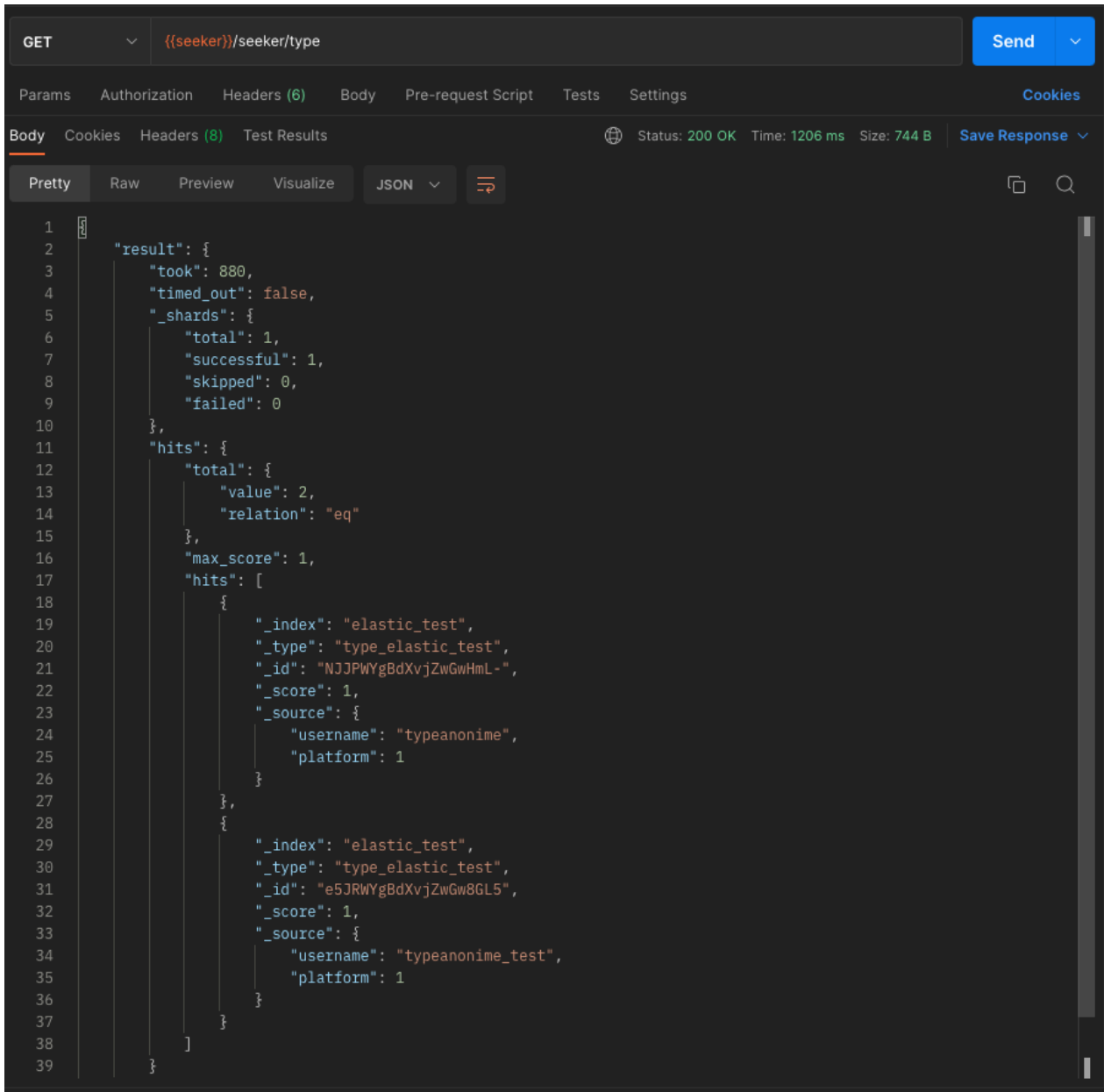
Esse módulo também foi desenvolvido com o *Express*, já a sua funcionalidade está empregada em criar, ler e pesquisar o nome do usuário mencionado na plataforma de maneira rápida se beneficiando do mecanismo do *ElasticSearch*. Abaixo, duas rotas para desse microserviço.

Figura 4.4 - Microserviço de pesquisa – salvar nome de usuário



Fonte: autoria própria

Figura 4.5 - Microserviço de pesquisa – rota de pesquisa por usuário



The screenshot shows a REST client interface with a GET request to the endpoint `{{seeker}}/seeker/type`. The response is a 200 OK status with a time of 1206 ms and a size of 744 B. The response body is displayed in JSON format, showing search results for a user.

```
1  {
2    "result": {
3      "took": 880,
4      "timed_out": false,
5      "_shards": {
6        "total": 1,
7        "successful": 1,
8        "skipped": 0,
9        "failed": 0
10     },
11     "hits": {
12       "total": {
13         "value": 2,
14         "relation": "eq"
15       },
16       "max_score": 1,
17       "hits": [
18         {
19           "_index": "elastic_test",
20           "_type": "type_elastic_test",
21           "_id": "NJJPWYgBdXvjZwGwHmL-",
22           "_score": 1,
23           "_source": {
24             "username": "typeanonime",
25             "platform": 1
26           }
27         },
28         {
29           "_index": "elastic_test",
30           "_type": "type_elastic_test",
31           "_id": "e5JRWYgBdXvjZwGw8GL5",
32           "_score": 1,
33           "_source": {
34             "username": "typeanonime_test",
35             "platform": 1
36           }
37         }
38       ]
39     }
40   }
41 }
```

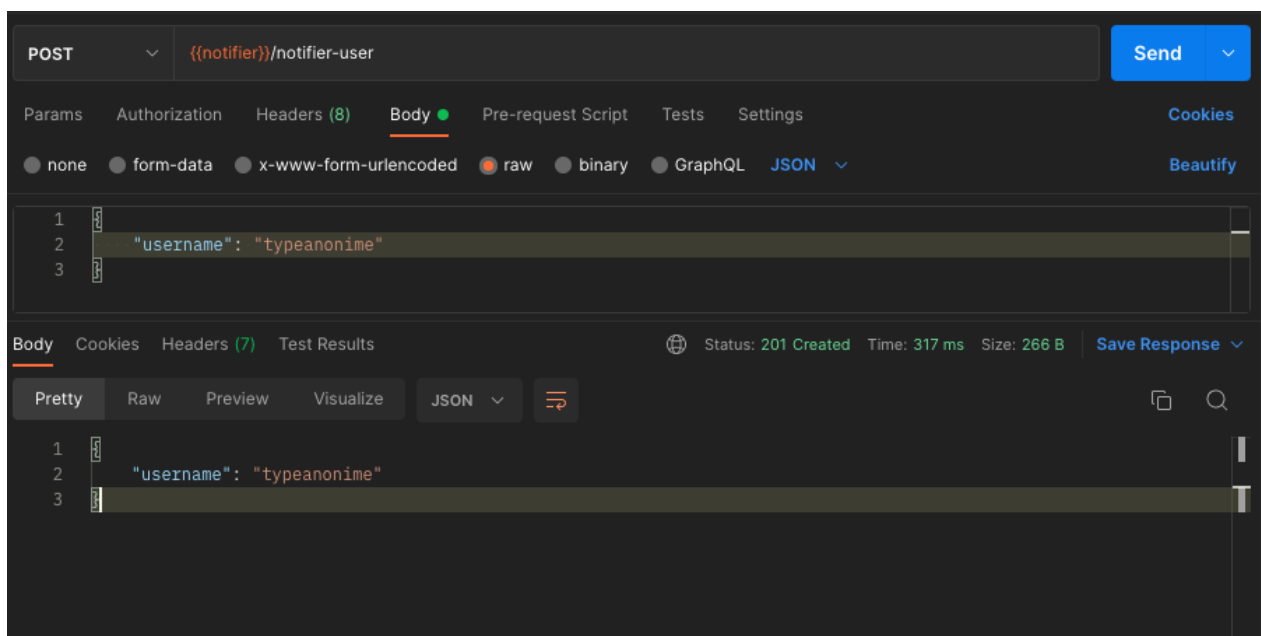
Fonte: autoria própria

### 4.1.3 Microserviço de notificação

Esse módulo foi desenvolvido com o *NestJs*, a sua funcionalidade está empregada em receber o nome do usuário e executar uma fila de processos. Esse processo é para o envio da notificação na plataforma do *Twitter*, como se trata de uma requisição para a *API* da rede social e é de formato assíncrona, foi implementado essa solução tarefas em segundo plano, adicionando em fila as requisições sem atrapalhar o fluxo da experiência do usuário na plataforma.

A motivação para esse microserviço ser realizado com outro *framework* é para o demonstrativo prático de que os microserviços não se limitam a tecnologia embarcada. Nesse microserviço foi projetado com dois bancos de dados diferentes para realização de diferentes tarefas: o *MongoDB*, como já citado anteriormente, salvará informações sobre o usuário de maneira persistente; O outro banco de dados é o *Redis*, um banco de dados em memória que persiste por tempo determinado para execução de tarefas em segundo plano, as filas dos processos. Abaixo a única rota desse serviço.

Figura 4.6 - Microserviço de notificações – rota para notificar usuário no Twitter



Fonte: autoria própria

## 4.2 Frontend – Microfrontend

De forma geral, os microfrontends foram construídos baseados na biblioteca *ReactJs* e compartilham algumas dependências entre si, um exemplo é *styled-components*, uma biblioteca para o desenvolvimento de componentes baseados no *CSS-in-JS*, que visa a facilidade da produção de telas com estilizações dentro do *JavaScript*.

### 4.2.1 Microfrontend – Home

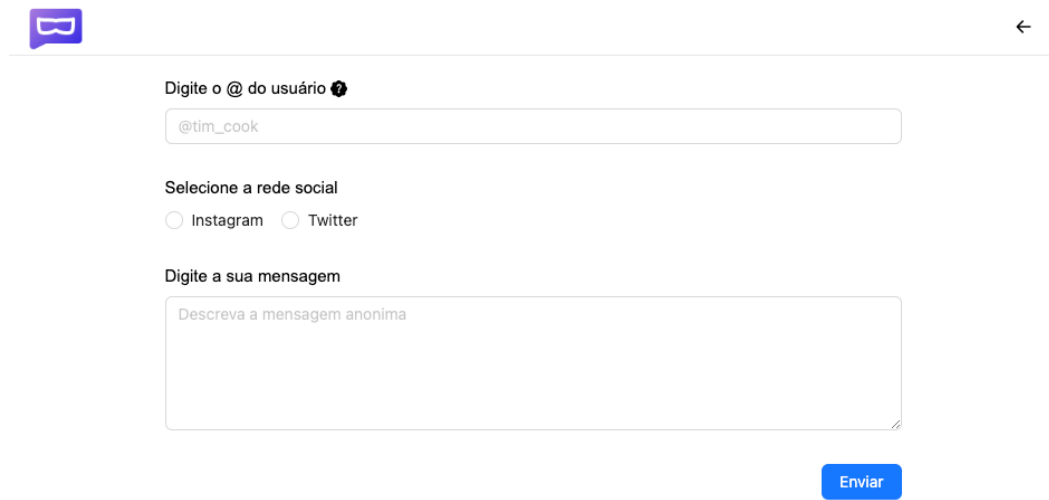
Esse módulo tem duas telas: a página inicial e a página do preenchimento da mensagem.

*Figura 4.7 - Microfrontend módulo Home – Página inicial*



Fonte: autoria própria

*Figura 4.8 - Microfrontend módulo Home – Página de formulário*

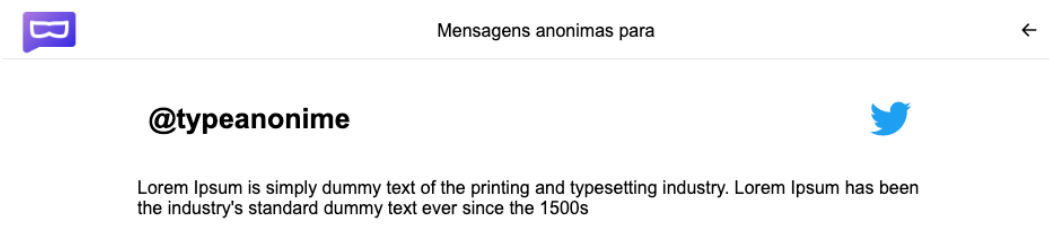


Fonte: autoria própria

#### 4.2.2 Microfrontend – Perfil do Usuário

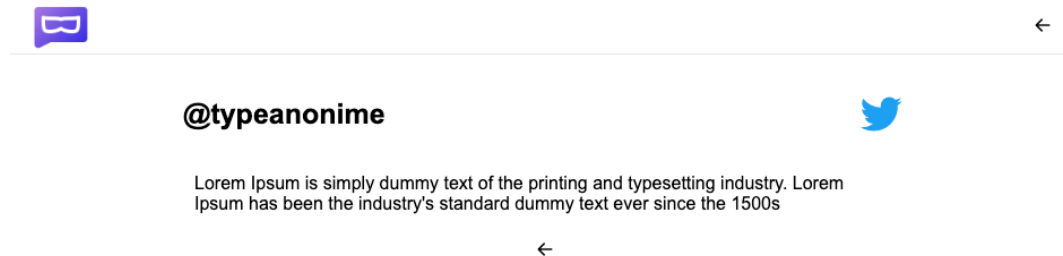
Para esse módulo também tem duas telas: página do perfil do usuário, onde lista todas as mensagens recebidas e a tela da mensagem em si, para o usuário poder compartilhar o endereço da página.

*Figura 4.9 - Microfrontend módulo Perfil – Página de listagem de mensagens*



Fonte: autoria própria

*Figura 4.10 - Microfrontend módulo Perfil – Página de mensagem*



Fonte: autoria própria

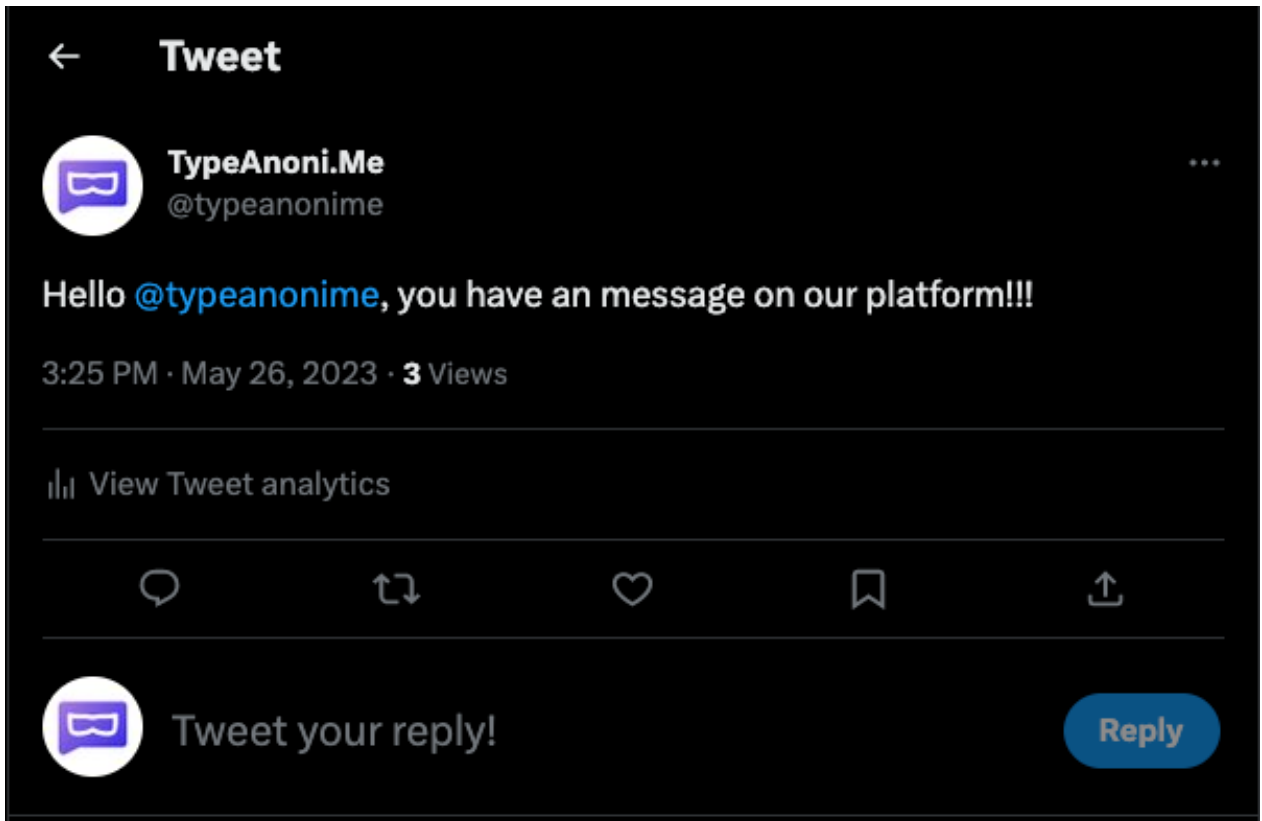
A diferença entre os módulos são as dependências instaladas, como por exemplo, no módulo *Home* está instalado um gerenciador de estados de formulário (o *React-Hook-Form*), já no outro módulo tem a dependência do *Zustand*, uma biblioteca de gerenciamento de estados globais da aplicação e por conta dessa modularização a aplicação fica mais leve pelo fato de ter dependências a menos quando não utilizadas.



### 4.3 Notificação no Twitter

Ao final da criação da mensagem, caso o usuário não tenha sido notificado nas últimas 24 horas, ele será mencionado pelo robô do Twitter, como no exemplo abaixo.

*Figura 4.11 - Notificação enviada ao Twitter*



Fonte: autoria própria

## 5 CONCLUSÃO

Esse novo mundo de microsserviços e microfrontend estão cada vez mais presentes nas aplicações de grandes empresas. Através deste trabalho, foi apresentado e compreendido o estudo dos benefícios sobre a arquitetura de microsserviços e microfrontend: a importância da modularização; escalabilidade; resiliência e diminuição de custos das aplicações, também os desafios de se trabalhar com essa arquitetura e como elas funcionam na prática com a aplicação de mensagens anônimas o *TypeAnoni.Me*.

Foi compreendido os conceitos sobre as tecnologias que abordam essa arquitetura, como os sistemas de mensageria (*Apache Kafka e RabbitMQ*) e chamadas *HTTP* no padrão *REST (API RESTful)*. Para o *frontend* entendemos conceitos com o *Module Federation*, o responsável pela modularização das interfaces de usuário da aplicação e como está o mercado atual com aplicações da mesma natureza que o *TypeAnoni.Me*.

Foi abordado também toda a sua modelagem de dados e o fluxo das demandas entre os módulos do *backend*, quais e como as tecnologias foram usadas, como a *API RESTful*. Concluindo o estudo sobre a arquitetura de microsserviços e microfrontend foi apresentado e desenvolvido com uma aplicação prática de todos os processos básicos dos conceitos aprendidos.

Como este trabalho é apenas acadêmico, para o futuro dessa aplicação a ser levado para produção e chegar na mão dos usuários deverá passar por testes maiores de qualidade, visando o aumento da sua experiência de quem usufrui, atendendo todas as normas *LGPD* para a proteção das informações dos usuários.

## REFERÊNCIAS

ALEXANDRE, M. **Introdução à arquitetura de microservices com *Spring Boot***. Disponível em: <<https://www.devmedia.com.br/introducao-a-arquitetura-de-microservices-com-spring-boot/33703>>. Acesso em: 13 maio. 2023.

AMAZON AWS. **O que é SOA? – Explicação sobre arquitetura orientada a serviços**. Disponível em: <<https://aws.amazon.com/pt/what-is/service-oriented-architecture/>>. Acesso em: 21 maio. 2023.

AMAZON AWS. **O que são microsserviços?** Disponível em: <<https://aws.amazon.com/pt/microservices/>>. Acesso em: 5 maio. 2023.

CAPITAL ONE. ***Anonymous Messaging Apps: Is Honesty Really The Best Policy?*** Disponível em: <<https://www.forbes.com/sites/capitalone/2017/09/28/anonymous-messaging-apps-is-honesty-really-the-best-policy/?sh=f92fe33c75f2>>. Acesso em: 26 abr. 2023.

DE CARVALHO, D. **Microserviços é coisa do passado. Agora a moda é Micro Front End!** Disponível em: <<https://medium.com/@dudousxd/microservi%C3%A7os-%C3%A9-coisa-do-passado-agora-a-moda-%C3%A9-micro-front-end-303ace0aa6de>>. Acesso em: 20 maio. 2023.

GOMES, D. **Introdução a Microsserviços**. Disponível em: <<https://medium.com/introducao-a-arquitetura-de-microservicos/introdu%C3%A7%C3%A3o-a-microsservi%C3%A7os-25378269e6f9>>. Acesso em: 13 maio. 2023.

KINDRAZKI, M. **Introdução ao Module Federation**. Disponível em: <<https://blog.arcotech.io/introdu%C3%A7%C3%A3o-ao-module-federation-f53b55a17c1b>>. Acesso em: 20 maio. 2023.

LIRA, D. **Introdução a microsserviços: um estudo de caso.** Disponível em: <<https://danilo-lira01.medium.com/introdu%C3%A7%C3%A3o-a-microsservi%C3%A7os-um-estudo-de-caso-6528a7ff059e>>. Acesso em: 13 maio. 2023.

NGL SAFETY CENTER. **Staying Safe on NGL.** Disponível em: <<https://ngl.link/safety>>. Acesso em: 26 abr. 2023.

REDHAT. **O que é API REST?** Disponível em: <<https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>>. Acesso em: 20 maio. 2023.

REDHAT. **O que são microsserviços?** Disponível em: <<https://www.redhat.com/pt-br/topics/microservices/what-are-microservices>>. Acesso em: 18 maio. 2023.

RESENDE, J. P. **O que é micro front end?** Disponível em: <<https://dev.to/jpbrab0/o-que-e-micro-front-end-4kci>>. Acesso em: 20 maio. 2023.

ROCHA, L. **Mensageria. Cada vez mais utilizado e exigido no TI.** Disponível em: <<https://www.linkedin.com/pulse/mensageria-cada-vez-mais-utilizado-e-exigido-ti-luciano-rocha/?originalSubdomain=pt>>. Acesso em: 16 maio. 2023.

TELLONYM. **How we safeguard our users.** Disponível em: <<https://info.tellonym.me/users/safety>>. Acesso em: 26 abr. 2023.

MONGODB. **What Are Microservices?** Disponível em: <<https://www.mongodb.com/databases/what-are-microservices>>. Acesso em: 20 maio. 2023.