SOAP SYNC: INTERAÇÕES COM API SOAP

Jefferson Lima dos Santos, Lucas Murari de Sousa, Fabricio Henrique Gustavo

[1]Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)

Ribeirão Preto, SP – Brasil

jefferson.santos106@fatec.sp.gov.br,
lucas.sousa42@fatec.sp.gov.br,
fabricio.henrique@fatec.sp.gov.br

Resumo. *Este artigo apresenta o SOAP Sync, uma aplicação educacional que replica as funcionalidades do GraphQL para uma API SOAP. Desenvolvida com Node.js, React e Figma, a aplicação SOAP Sync proporciona aos usuários uma experiência prática no uso de requisições SOAP, permitindo selecionar métodos (GET, PUT, POST, DELETE), definir URLs de destino, visualizar o histórico de requisições, editar o corpo da requisição, adicionar headers e ver as respostas. Os resultados da implementação demonstram que a aplicação atendeu aos requisitos estabelecidos, facilitando o aprendizado e a interação com APIs SOAP.*

Abstract. *This article presents SOAP Sync, an educational application that replicates the functionalities of GraphQL for a SOAP API. I was developed with Node.js, React, and Figma, the SOAP Sync application offers an intuitive web interface divided into three sections: method and object selection, method visualization, and request response display. Users can select methods (GET, PUT, POST, DELETE), define target URLs, view request history, edit the request body, add headers, and view responses. The implementation results demonstrate that the application met the established requirements, facilitating learning and interaction with SOAP APIs.*

1. Introdução

Na era digital, as APIs (Interfaces de Programação de Aplicativos) assumem um papel fundamental como pontes que conectam e integram informações entre diferentes sistemas e aplicações. Sua importância reside na capacidade de ampliar a funcionalidade de softwares, facilitar o acesso a dados e promover a conectividade entre diferentes sistemas.

Neste contexto, surge a necessidade de explorar e compreender as diferentes tecnologias de APIs, suas funcionalidades e aplicações práticas. Este projeto visa desenvolver uma aplicação que reproduza as funcionalidades do GraphQL para uma API SOAP. Com foco educacional, a aplicação permitirá que os usuários aprendam na prática como fazer requisições SOAP. Além de aprofundar o conhecimento sobre a tecnologia, o projeto visa proporcionar uma experiência prática e didática na sua utilização.

A proposta central deste trabalho é a implementação dessa aplicação em uma única tela web, segmentada em três seções distintas: uma dedicada à seleção de métodos e objetos, outra à visualização dos métodos, seguida por uma seção para a observação

da resposta da requisição. Detalhes mais aprofundados sobre essas seções serão apresentados ao longo da evolução do trabalho.

Assim, este artigo está organizado da seguinte maneira: na Seção 2, serão abordadas as bases teóricas que sustentam os princípios discutidos ao longo deste artigo, incluindo conceitos e funcionalidades das APIs, REST e SOAP. Na Seção 3, serão apresentados os materiais e métodos empregados no projeto, destacando as tecnologias chave utilizadas, como Node.js, React e Figma, além da inspiração do GraphQL. Na Seção 4, serão descritos os detalhes do desenvolvimento da aplicação, incluindo os documentos de requisitos, protótipo da aplicação, diagramas de casos de uso e classes. Finalmente, a Seção 5 trará as considerações finais, destacando os resultados alcançados e as conclusões do projeto.

2. Referencial teórico

Nesta seção serão exploradas as bases teóricas que irão sustentar os princípios abordados ao longo deste artigo. Serão abordadas as referências de maior relevância e as mais amplamente utilizadas para esse propósito.

2.1. Conceitos e funcionalidades das APIs

Uma API (Interface de Programação de aplicativos) nada mais é do que um componente de software que possui um conjunto de funções que fazem a intermediação do acesso as funcionalidades de um sistema.

É essencial, todavia entender e compreender mais profundamente os conceitos e fundamentos das APIs.

Segundo Amazon (2022a),

> As APIs são mecanismos que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos. Por exemplo, o sistema de software do instituto meteorológico contém dados meteorológicos diários. A aplicação para a previsão do tempo em seu telefone “fala” com esse sistema por meio de APIs e mostra atualizações meteorológicas diárias no telefone.

APIs são estruturas de *back-end* criadas para lidar exclusivamente com dados de maneira centralizada, facilitando o desenvolvimento independente de aplicações clientes com interfaces direcionadas ao usuário final. Esses aplicativos clientes frequentemente abrangem mobile apps, aplicações desktop ou web apps (RIBEIRO, 2016).

Um caso real que exemplifica de maneira efetiva a utilidade das APIs é o caso de previsões do tempo em aplicativos mobile ou web. O sistema de software do instituto meteorológico contém dados detalhados sobre as condições climáticas, como temperatura, umidade e previsões para os próximos dias.

A aplicação de previsão do tempo em um celular utiliza uma API fornecida pelo instituto meteorológico para obter acesso a esses dados. Por meio dessa API, a aplicação pode enviar solicitações específicas para obter informações sobre a temperatura atual, previsões para as próximas horas ou dias, e outros dados meteorológicos relevantes (AMAZON, 2022a).

2.2. Rest

REST ou representação de estado transferência é um estilo de arquitetura de software que os desenvolvedores aplicam às APIs da web e, de acordo com o site (CYCLR, 2023)[1]

> uma API REST funciona de forma semelhante à quando você solicita ou pesquisa algo no Google. Em resposta, você obtém uma lista de resultados do serviço que você está solicitando. AS APIs são a espinha dorsal da web e dos aplicativos móveis.

O uso de APIs REST é recomendado para projetos que exigem escalabilidade, compatibilidade e desempenho, pois têm melhor velocidade de execução e consomem menos recursos de memória em comparação com o SOAP (TIHOMIROVS, GRABIS 2016)[2].

As APIs REST são projetadas para utilizar os métodos HTTP padrão, como GET, POST, PUT e DELETE, para operações de dados.

Um exemplo de aplicação de uma API REST poderia ser um *web service* que fornece uma interface para gerenciar produtos em uma loja online. Ele poderia ser acessado utilizando operações HTTP padrão, como os já citados métodos GET, POST, PUT e DELETE. As mensagens REST destinadas a este serviço web seriam formatadas em JSON, e cada recurso teria um URL específico.

Enquanto o SOAP oferece vantagens como independência de linguagem e padronização, o REST é preferido por sua facilidade de uso, eficiência e flexibilidade, especialmente devido ao uso de formatos de mensagem menores como JSON (SOAPUI, 2023)[3].

2.3. Soap

SOAP ou *Simple Object Access Protocol*, que significa Protocolo Simples de Acesso a Objetos, é definido de acordo com o site (CYCLR, 2023)[4] como

> uma especificação de mensagem para a troca de informações entre sistemas e aplicações. Ele permite que processos usando diferentes sistemas operacionais se comuniquem via HTTP. APIs SOAP são projetadas para criar, recuperar, atualizar e excluir registros.

O SOAP e o REST são os dois modelos de design mais proeminentes para serviços da web, com o SOAP sendo conhecido por sua forte tipagem de mensagens baseada em XML e pelo uso do WSDL como contrato entre provedor e consumidor de serviço (SOAPUI, 2023)[5].

Uma analogia comum compara SOAP a um envelope e REST a um cartão postal. Enquanto um cartão postal é mais rápido e mais barato, acessar um envelope requer alguns passos extras. Ao projetar APIs, o uso de SOAP se concentra na mensagem. (CYCLR, 2023)[6].

---

[1] Tradução livre dos autores.
[2] Tradução livre dos autores.
[3] Tradução livre dos autores.
[4] Tradução livre dos autores.
[5] Tradução livre dos autores.
[6] Tradução livre dos autores.

A descrição de uma arquitetura de web *service* baseada em SOAP, destaca três partes fundamentais: o provedor de serviços, encarregado de publicar o web *service*; o registro do servidor, responsável por documentar todas as operações disponíveis; e o consumidor, que se conecta ao serviço e o utiliza para fazer requisições (LOPES, SILVA, PONCIANO, 2023).

3. Materiais e Métodos

Nesta seção, serão delineados os materiais e métodos empregados neste projeto, com foco especial nas tecnologias chave utilizadas: Node.js, GraphQL, React e Figma.

3.1. Node.js

O Node.js, ambiente de execução JavaScript, foi uma das ferramentas utilizadas na construção do projeto. Ele possibilita a interpretação de códigos em JavaScript ou TypeScript e se destaca por sua alta escalabilidade, atendendo a demandas expressivas sem comprometer o desempenho. Além disso, sua arquitetura de baixo nível permite acesso direto aos recursos do sistema operacional (PEREIRA, 2016).

De acordo com Pereira (2014, apud Barsoti, Gibertoni, 2020) o Node.js é um ambiente de tempo de execução de JavaScript e TypeScript criado por Ryan Dahl em 2009. Ele foi desenvolvido com o objetivo de oferecer uma tecnologia inovadora com uma arquitetura *non-blocking thread*, ou seja, as operações não esperam umas pelas outras para serem concluídas. A principal motivação para a criação do Node.js foi a limitação dos sistemas web desenvolvidos em linguagens como Java, Python, PHP e .NET, que tendem a suspender todo o processamento enquanto aguardam operações de entrada/saída no servidor.

O Node.js é uma plataforma versátil e escalável que pode ser usada para uma grande variedade de propósitos, incluindo a construção de APIs.

3.2. GraphQL

Em sua essência, o GraphQL representa uma inovação na forma como as solicitações de dados são gerenciadas. Ao possibilitar que desenvolvedores requisitem dados de várias fontes em uma única chamada de API, o GraphQL se destaca por fornecer precisamente o que os clientes solicitam, sem excessos. Essa abordagem focada na necessidade do cliente representa um avanço significativo na eficiência e na personalização das interações de dados, redefinindo assim a maneira como as APIs são projetadas e utilizadas (CYCLR, 2023)[7].

De acordo com o site (CYCLR, 2023)[8].

> GraphQL é usado onde os dados estão aninhados e precisam ser recuperados em uma única chamada, como no exemplo do nosso diagrama abaixo, em um exemplo de uma plataforma de rede social, seria utilizado essa metodologia quando for necessário recuperar posts, assim como comentários, curtidas e compartilhamentos.

Observando o diagrama apresentado na figura 1, é possível visualizar de maneira clara o funcionamento do GraphQL.

[7] Tradução livre dos autores.
[8] Tradução livre dos autores.

O GraphQL possibilita a agregação de dados de múltiplas fontes em uma única solicitação, resultando na diminuição do volume de chamadas de rede e da demanda por largura de banda (AMAZON, 2023b).

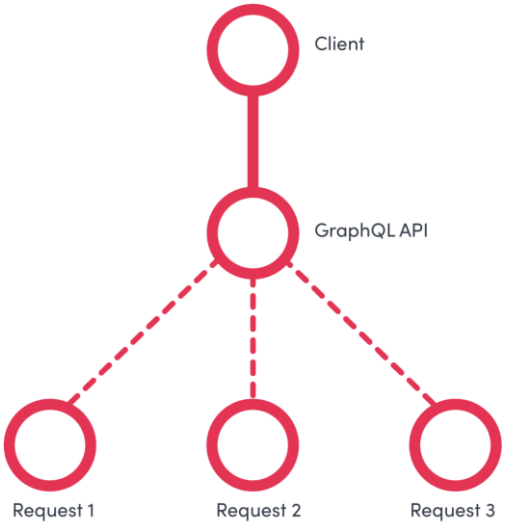

Figura 1. Funcionamento do GraphQL
Fonte: (CYCLR, 2023)

3.3. React

De acordo com as pesquisas de Kumar e Singh (2016 *apud* Falcão, 2022) o React é uma biblioteca de JavaScript, uma ferramenta de desenvolvimento de interface de usuário que foi criada pelo Facebook em 2013 com o intuito de simplificar a elaboração de componentes interativos, reutilizáveis e de estados dinâmicos, ou seja, à capacidade dos componentes de mudar e de se atualizar automaticamente de estado em resposta a ações do usuário ou eventos externos. (FALCÃO, 2022).

Uma das principais razões para utilizar o React é sua eficiência e desempenho. A tecnologia desenvolvida pelo Facebook opera com o *Virtual Document Object Model*, conhecido como DOM. Isso permite que o React crie uma cópia virtual da página web na memória do computador, resultando em uma manipulação mais responsiva dos elementos na tela (FALCÃO, 2022).

Segundo Falcão (2022, p. 21) “Outra particularidade marcante do React é a linguagem utilizada para gerar os componentes de interface, o JSX”.

JSX é uma extensão de sintaxe para JavaScript que permite a escrita de código JavaScript com a inclusão de elementos semelhantes ao HTML. O React não exige o uso obrigatório do JSX, mas oferece suporte a essa sintaxe. Para arquivos que combinam essa sintaxe com TypeScript, linguagem essa que será utilizada nesse projeto, utiliza-se a extensão .TSX (REACTJS, 2024).

3.4. Figma

A prototipação da aplicação foi feita através da plataforma Figma, uma ferramenta extremamente versátil e voltada para a criação de interfaces e protótipos que opera diretamente no navegador, permitindo a colaboração online. A plataforma facilitou a construção do protótipo, visto que, os usuários podem acessar seus projetos de qualquer lu-

gar, bastando fazer login na plataforma (HOSTINGER, 2023).

A plataforma é projetada com uma interface intuitiva, baseada em elementos visuais de fácil uso, além disso, oferece uma ampla gama de recursos avançados, que possibilitam a criação de praticamente qualquer tipo de design de projetos específicos (HOSTINGER, 2023).

4. Desenvolvimento

Nesta seção, abordaremos os aspectos relacionados ao desenvolvimento da aplicação. Serão apresentados os documentos de requisitos da aplicação, os diagramas de casos de uso e de classes. Além disso, incluirá um protótipo desenvolvido na plataforma Figma.

4.1. Requisitos do sistema

O projeto apresentado neste artigo foi desenvolvido com o propósito de criar uma ferramenta que simule as funcionalidades do GraphQL para uma API SOAP. Para atingir esse propósito, foram estabelecidos requisitos funcionais e não funcionais que moldam a eficácia e usabilidade da aplicação desenvolvida.

4.1.1. Requisitos funcionais e não funcionais

Os requisitos funcionais e não funcionais da aplicação desenvolvida foram organizados em uma tabela para facilitar a organização e a compreensão.

IDENTIFICADOR	DESCRIÇÃO
RF 01	A aplicação deve conter três seções diferentes exibidas paralelamente em uma única tela.
RF 02	A primeira seção da aplicação deve conter diferentes métodos SOAP, para serem selecionados pelo usuário (GET, PUT, POST, DELETE).
RF 03	A segunda seção deve conter o objeto da requisição, sendo eles o corpo e o cabeçalho.
RF 04	A terceira seção da aplicação deve exibir os resultados obtidos a partir da requisição e também manter um histórico das requisições feitas pelo usuário para visualização e consulta.
RF 05	Cada método listado deve proporcionar ao usuário a opção de visualizar informações detalhadas sobre seu funcionamento, por meio de um botão associado a cada método.
RF 06	A aplicação deve incluir um campo de texto onde o usuário pode selecionar ou especificar manualmente a URL de destino para a requisição.
RN 01	A interface da aplicação deve ser intuitiva e de fácil utilização, garantindo uma experiência acessível ao usuário.
RN 02	O sistema deve garantir que as consultas SOAP sejam respondidas em tempo hábil, sem atrasos significativos, assegurando uma experiência fluida e eficiente.
RN 03	Em termos de portabilidade, o sistema deve ser acessível via web nos principais navegadores utilizados (Google Chrome, Microsoft Edge, Firefox).
RN 04	A aplicação deve conter um ciclo de vida pequeno, mas com boa manutenibilidade.

Figura 2. Requisitos da aplicação
Fonte: (Autoria própria, 2023)

4.1.2. Diagrama de caso de uso

O diagrama de caso de uso representa a interação entre o ator representado pelo usuário e as funcionalidades da aplicação desenvolvida.

Para o usuário as funcionalidades da aplicação consistem em:

- Selecionar um dos métodos listados: o usuário deve selecionar um dos quatro métodos que são utilizados pelo SOAP (GET, PUT, POST, DELETE).
- Selecionar a URL de destino já existente na aplicação ou digitar uma URL de destino de escolha.
- Visualizar o histórico completo de requisições: o usuário pode opcionalmente consultar o histórico de suas requisições anteriores, que são armazenadas através de cookies pelo navegador.
- Preencher o corpo da requisição: o usuário pode modificar as linhas de código no editor para personalizar a requisição. Esta ação não é obrigatória para que a requisição funcione.
- Adicionar um ou mais *headers*: o usuário pode adicionar um ou mais *headers* à requisição, preenchendo os campos de "*header key*" e "*header value*" para cada *header* adicionado, exemplo: o usuário pode opcionalmente definir a primeira página da requisição inserindo "*page*" no campo “chave do *header”* e "1" no campo “valor do *header”*.
- Visualizar a resposta da requisição: após selecionar um dos métodos listados, especificar a URL de destino e, opcionalmente, personalizar a requisição, o usuário receberá o resultado.

Os casos de uso listados representam as principais funcionalidades da aplicação SOAP Sync, conforme ilustrado no diagrama de caso de uso apresentado na figura 2.

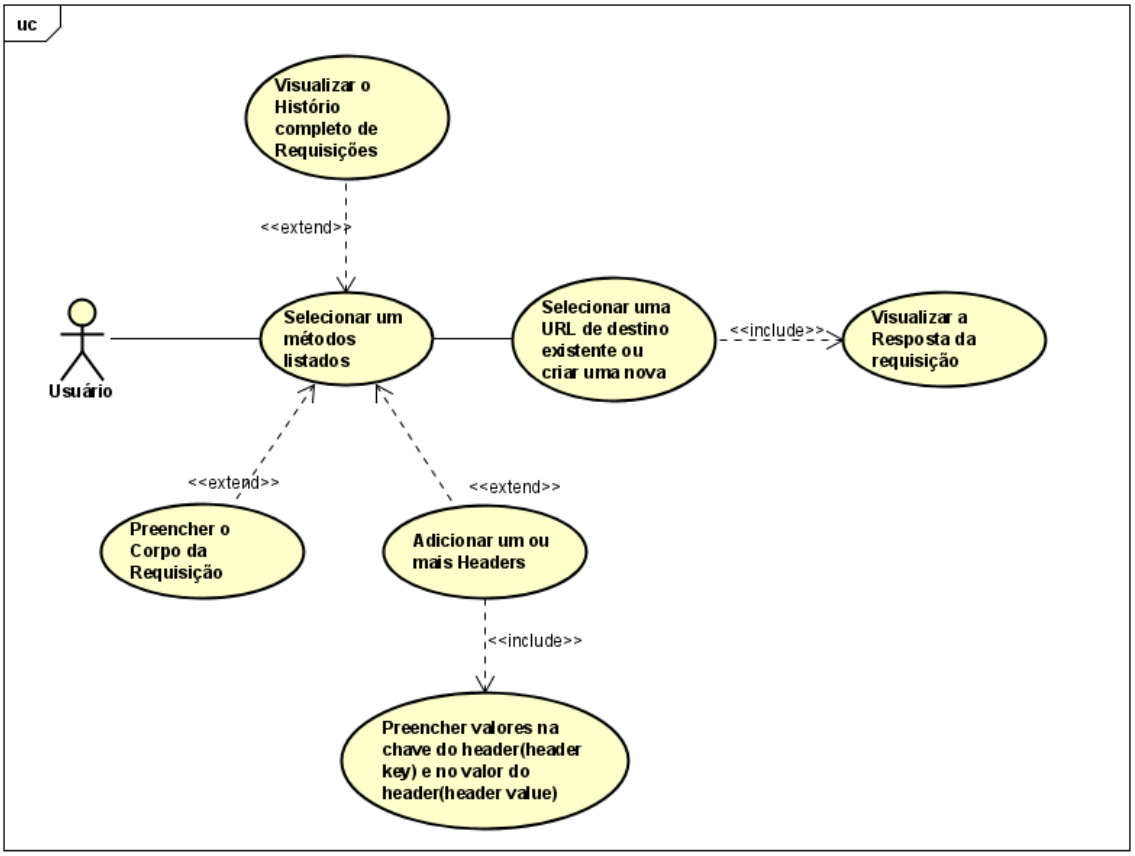

Figura 3. Diagrama de Casos de uso demonstrando as principais funcionalidades da aplicação soap sync
Fonte: (Autoria própria, 2023)

4.1.3. Diagrama de caso de classe

O diagrama de classe representa a estrutura estática da aplicação SOAP Sync, mostrando as classes que compõem a aplicação, seus atributos, métodos e os relacionamentos entre elas. O diagrama de classe segundo (PRESSMAN, p. 1695) "Fornece uma visão estática ou estrutural do sistema. Ele não mostra a natureza dinâmica das comunicações entre os objetos das classes no diagrama". O diagrama de classe desse projeto está demonstrado na Figura 4.

O repositório contendo o código da aplicação, está disponível para consulta no GitHub, por meio do link: https://github.com/

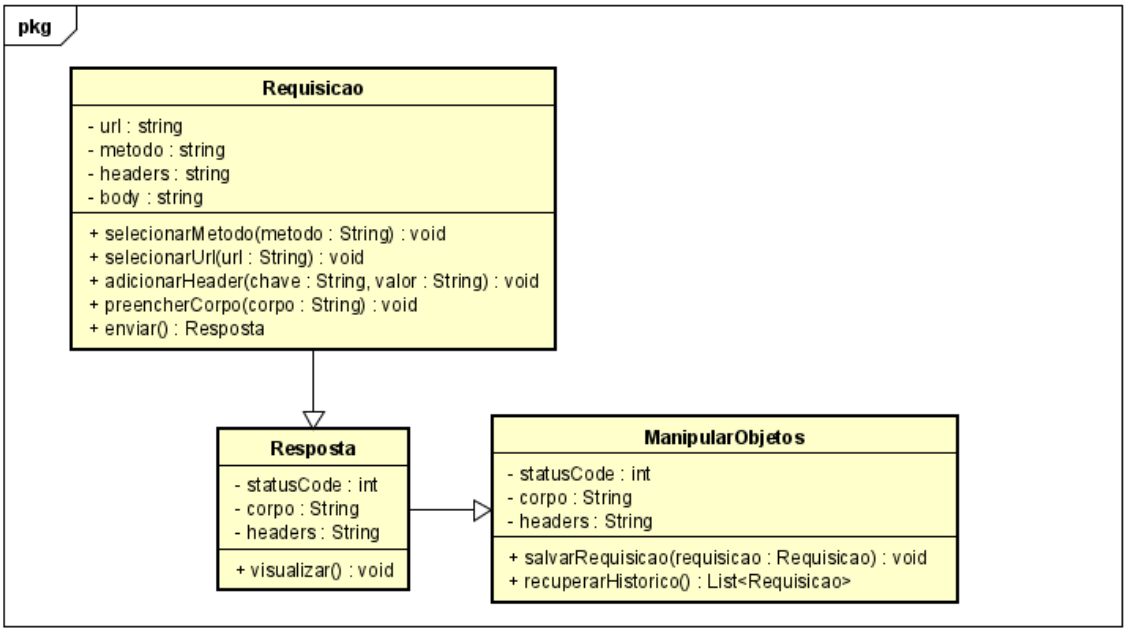

Figura 4. Diagrama de Classes do funcionamento da aplicação soap sync

Fonte: (Autoria Própria, 2023)

4.2. Protótipo

O protótipo da aplicação SOAP Sync, representado pela Figura 5, foi desenvolvido utilizando a plataforma Figma. Seu objetivo principal é fornecer uma visão clara e intuitiva do funcionamento da aplicação.

A interface principal da aplicação está dividida em três seções distintas, cada uma desempenhando um papel crucial na experiência do usuário. A primeira seção, localizada no topo da interface, é dedicada à seleção de métodos e objetos. Aqui, os usuários podem escolher entre os quatro métodos SOAP (GET, PUT, POST, DELETE).

A segunda seção do protótipo é a área de visualização dos métodos. Nesta parte da interface, os usuários podem selecionar ou inserir a URL de destino e visualizar e editar o corpo da requisição. O editor de código embutido permite que os usuários modifiquem as linhas de código conforme necessário para personalizar suas requisições. Além disso, os usuários têm a opção de adicionar um ou mais headers à requisição, preenchendo os campos de *"key"* e *"value"* do *"header"*.

A terceira seção, situada na parte inferior da interface, é dedicada à exibição das respostas das requisições. Após configurar e enviar uma requisição, os usuários podem visualizar as respostas recebidas diretamente nesta área. Outro recurso importante do protótipo é a funcionalidade de histórico de requisições, os usuários podem acessar um registro completo de suas requisições anteriores.

A prototipação completa da aplicação, está disponível para consulta no Figma, por meio do link: figma.com/design

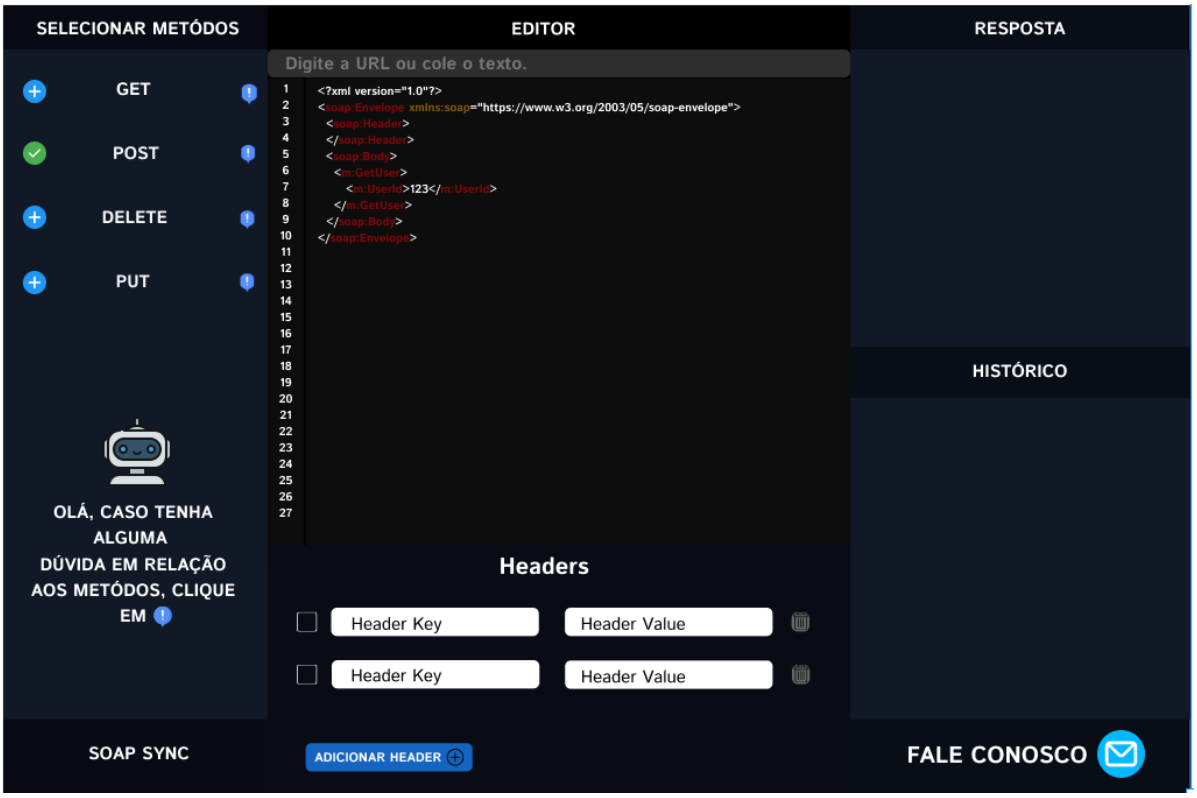

Figura 5. Protótipo da aplicação

Fonte: (Autoria Própria, 2023)

5. Considerações Finais

Neste trabalho, foi desenvolvida a aplicação SOAP Sync com o propósito de replicar as funcionalidades do GraphQL para uma API SOAP, proporcionando uma experiência educacional prática e intuitiva. Durante o desenvolvimento da aplicação, enfrentamos desafios ao compreender o funcionamento das requisições SOAP, especialmente na implementação correta dos métodos. Esses desafios evidenciaram a necessidade de métodos de aprendizado mais curtos e simples para desenvolvedores que trabalham com integrações SOAP.

Como próximo passo para a evolução deste projeto, planejamos expandir as capacidades da aplicação adicionando suporte para requisições REST e explorar outros tipos de requisições. Essa expansão tornará o SOAP Sync uma ferramenta ainda mais abrangente e útil, auxiliando os desenvolvedores a dominar uma variedade maior de tecnologias de integração de sistemas.

Referências

AMAZON. O que é uma API? **Amazon.** 2022a. Disponível em: https://aws.amazon.com/pt/what-is/api/. Acesso em: 07 mai. 2023.

AMAZON. What is GraphQL? **Amazon.** 2023b. Disponível em: https://aws.amazon.com/pt/graphql/. Acesso em: 07 mai. 2023.

BARSOTI, N.; GIBERTONI, D. **Impacto que o Sequelize traz para o desenvolvimento de uma API construída em Node.js com Express.js.** 2020. Trabalho

de Conclusão de Curso (Graduação) – Faculdade de Tecnologia (Fatec), Taquaritinga, 2020.

CYCLR. Know Your API Types – REST vs SOAP vs GraphQL **Cyclr.** 2023. Disponível em: https://cyclr.com/blog/rest-vs-soap-vs-graphql-api-types. Acesso em: 22 mar. 2023.

FALCÃO, F. D. **Desenvolvimento do aplicativo Turistando Beberibe utilizando React Native.** 2022. Trabalho de Conclusão de Curso (Graduação em Sistemas e Mídias Digitais) – Universidade Federal do Ceará, Instituto Universidade Virtual - UFC Virtual, Fortaleza, 2022.

HOSTINGER. Eleve Seus Designs: O Que é Figma e Como Usá-lo Corretamente! **Hostinger.** 2023. Disponível em: https://www.hostinger.com.br/tutoriais/figma-o-que-e. Acesso em: 15 nov. 2023.

LOPES, I. S. M.; SILVA, L. H. F.; PONCIANO, L. **Análise do custo-benefício de GraphQL em comparação a REST e SOAP em aplicações para dispositivos móveis.** 2023. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) – Instituto de Ciências Exatas e Informática, PUC Minas, Belo Horizonte, 2023.

PRESSMAN, Roger S; MAXIM, Bruce R. Engenharia de Software: Uma Abordagem Profissional**.** 9. ed. Porto Alegre: AMGH, 2021.

REACTJS. Introduzindo JSX **ReactJS.** 2024. Disponível em: https://pt-br.legacy.reactjs.org/docs/introducing-jsx.html. Acesso em: 14 mar. 2024.

RIBEIRO, C. Construindo APIs REST com Node.js. São Paulo: Casa do Código, 2016.

SOAPUI. SOAP vs REST 101: Understand The Differences **SoapUI.** 2023. Disponível em: https://www.soapui.org/learn/api/soap-vs-rest-api/. Acesso em: 22 mar. 2023.

TIHOMIROVS, J.; GRABIS, J. **Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics.** 2016. Trabalho de Conclusão de Curso (Graduação) – Riga Technical University, Riga, Letônia, 2016.