Um Estudo Sobre Boas Práticas Para Documentação De Apis

A Study On Best Practices For Apis Documentation

Ana Karoliny Silva de Araujo
Estudante do curso de Bacharelado Interdisciplinar em Tecnologia da Informação pela Universidade Federal Rural do Semi-Árido
Instituição: Universidade Federal Rural do Semi-Árido.
Endereço: Departamento de Engenharias e Tecnologia. Universidade Federal Rural do Semi-Árido (UFERSA), Rodovia BR-226, KM 405, s/n - São Geraldo, Pau dos Ferros – RN, Brasil, 59900-000.
E-mail: anakarolisilvaaaa@gmail.com

Jose Ferdinandy Silva Chagas
Mestre em Sistema Computacionais pela Universidade Federal Rural do Semi-Árido (UFERSA)
Instituição: Universidade Federal Rural do Semi-Árido
Endereço: Departamento de Engenharias e Tecnologia. Universidade Federal Rural do Semi-Árido (UFERSA), Rodovia BR-226, KM 405, s/n - São Geraldo, Pau dos Ferros – RN, Brasil, 59900-000.
E-mail: ferdinandy@ufersa.edu.br

Resumo. Documentação de API é um conjunto de informações que descreve como interagir com um determinado software. Essa documentação pode incluir informações como os *endpoints* disponíveis, os parâmetros aceitos e os tipos de dados que a API espera ou retorna. Embora a documentação de API possa ser uma tarefa desafiadora e frequentemente subestimada pelos desenvolvedores, a implementação de boas práticas pode ter um impacto significativo no produto final. Por esse motivo, é altamente recomendável que os desenvolvedores adotem essas práticas de documentação de APIs em seus projetos para maximizar a eficácia e a utilidade de suas aplicações. Neste caso, este trabalho tem como objetivo apresentar essas boas práticas, que podem ajudar desenvolvedores e usuários a entender melhor como utilizar esse documento. Além disso, discutimos também ferramentas que podem ajudar a criar e manter a documentação de API sempre atualizada, bem como os desafios comuns enfrentados pelos desenvolvedores em documentação e desenvolvendo um estudo de caso da API de pet. Por fim, destacamos a importância de uma boa documentação de API como um elemento essencial para o sucesso da mesma.

Palavras-chave: API (*Application Programming Interface*); Documentação; Desenvolvimento de software; Ferramentas de documentação.

Abstract*:* API documentation is a set of information describing how to interact with certain software. These documents can include information such as available endpoints, accepted parameters, and types of data the API expects or returns. While API documents can be a challenging and often overlooked task for developers, implementing best practices

can have a significant impact on the final product. For this reason, it is highly recommended that developers adopt these API document practices in their projects to maximize the effectiveness and usefulness of their applications. In this case, this work aims to present these good practices, which can help developers and users to better understand how to use this document. In addition, we also discuss tools that can help create and keep API documents always up to date, as well as common challenges faced by developers in documents and developing a Pet API case study. Finally, we highlight the importance of good API documentation as an essential element for its success.

Keywords: API (*Application Programming Interface*); Documentation; Software development; Document tools.

Disponibilidade:
https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/59332

1. Introdução

A crescente importância das APIs como ferramenta de integração e comunicação entre sistemas tem levado a uma maior necessidade de documentação clara e completa dessas aplicações. No entanto, muitas vezes a documentação desses sistemas são negligenciada ou realizada de forma inadequada, o que pode levar a problemas de integração, dificuldades de manutenção e aumento dos custos de desenvolvimento. Diante disso, ao surgir a ideia de desenvolver um *software* é importante pensar nos processos que serão executados, a fim de garantir qualidade na entrega final do produto. O *software* evolui através de mudanças que levam ou brotam de novas ideias que iniciam o ciclo novamente (GRUBB, et. al 2003, pg.61). Desse modo, pensando na eficiência do sistema deve-se levar em consideração o ciclo de vida do *software*, já que ele define a ordem em que cada etapa deve ser executada. Para prolongar sua utilidade, o *software* passa por uma série de mudanças, sejam por alterações nas regras de negócio, expectativas do usuários, ou fatores externos, nesse caso a necessidade de mudanças geram novos requisitos.

Para que o *software* seja de alta qualidade, vale a pena empregar recursos no projeto durante sua implementação para que sejam diminuídos os custos em mudanças futuras. Principalmente investir em uma documentação detalhada e de fácil acesso aos desenvolvedores para que eles tenham uma boa compreensão do sistema e saibam analisar o impacto de novas modificações.

> As pesquisas em geral concordam que a manutenção de software ocupa uma proporção maior dos orçamentos de TI que o desenvolvimento. Elas também

> concordam que se gasta mais do orçamento de manutenção na implementação de novos requisitos do que na correção de bugs. (SOMMERVILLE, 2011, pg.171)

Para enfatizar a importância de recursos investidos em documentação, muitos sistemas de *software* são construídos a partir de componentes reutilizáveis (TRIPATHY, et. al., 2014). Porém, como consequência disso, muitos integradores não possuem uma visão detalhada do funcionamento interno dos componentes que pretende utilizar. Um tipo conhecido de componente reutilizável são as APIs, formadas por uma série de recursos que permitem a integração entre sistemas diferentes que automatiza o processo de troca de dados. Dessa forma, as APIs buscam garantir eficiência no desenvolvimento de novas e antigas aplicações, diminuindo tanto a jornada de trabalho dos desenvolvedores, quanto os custos da organização.

Buscando facilitar o gerenciamento, manutenção e evolução do software. Este trabalho busca propor recomendações de boas práticas e aplicá-la no desenvolvimento de uma aplicação, a fim de facilitar a compreensão do sistema, contribuindo para o processo de documentação de APIs no contexto de desenvolvimento web.

2. **Problema**

No âmbito de desenvolvimento de *software* é necessário que seja construída uma documentação clara e detalhada de API a partir do momento em que a mesma começa a ser desenvolvida, visto que é uma das principais fontes de informação que os desenvolvedores precisam para incorporar funcionalidades, agilizar rotinas e desenvolver recursos mais inteligentes do sistema.

Entretanto, é possível notar que há uma baixa preocupação por parte dos desenvolvedores em relação à criação de documentação de API. Segundo (SOMMERVILLE, 2011, pg.105) algumas pessoas pensam que a documentação detalhada não é útil e não vale o custo de seu desenvolvimento. No entanto, essa perspectiva pode acarretar sérios desafios e problemas, especialmente quando se trata da participação de novos colaboradores na equipe. Se a equipe de desenvolvimento sofre alterações, torna-se difícil para os novos membros da equipe construir um entendimento completo do sistema e seus componentes.

Nesse caso a medida em que o *software* evolui, torna-se cada vez mais complexos e volumosos, o que representa um grande desafio para a empresa de *software* em relação a sua manutenção. Este caso é um outro ponto que requer atenção, tendo em vista que a melhoria ou correção em determinado ponto deve ocorre analisando o passado e evitando possíveis impactos futuros.

Contudo, a equipe de desenvolvimento acaba pecando na documentação de API, devido ao gerenciamento inadequado das atividades. O que faz com que a empresa não tenha uma visão geral de todos os projetos que estão sendo desenvolvidos, nem dos projetos que estão por vir, e muito menos da prioridade de cada um deles (PRIKLADNICKI, 2004). Por esse motivo os desenvolvedores se sentem pressionados e acabam desenvolvendo o sistema de forma inadequada e da maneira que é entendido apenas pelo próprio autor.

Atribuído a isso os desenvolvedores se sentem desafiados também para adicionar novas tecnologias ao *software*, já que engenheiros de *softwares* muitas vezes têm que confiar em suas próprias investigações para descobrir dependências entre módulos de *software* ou para descobrir potencial efeitos *ripple* porque o sistema original e as alterações subsequentes não são documentadas (GRUBB, et. al., 2003, pg.74). Assim, apresenta-se a questão de pesquisa: "Como auxiliar desenvolvedores na documentação de APIs seguindo recomendações de boas práticas com apoio de ferramentas para documentação?"

3. Justificativa

O suporte para APIs é de responsabilidade de uma organização ou comunidade de *software*, porém quando uma empresa não preza em uma documentação estruturada do sistema que é desenvolvido ela acaba encontrando dificuldade para auxiliar os desenvolvedores a manter e desenvolver o *software*. Segundo (GRUBB, et. al., 2003) os sistemas de software são modificados e aprimorados por indivíduos que não estavam envolvidos com o desenvolvimento inicial.

É muito comum adquirir *software* open source, pelo fato de ser gratuito e possuir uma alta flexibilidade. No entanto, se você precisar de documentação e suporte, você pode ter de pagar por isso, embora os custos sejam usualmente bastante baixos. (SOMMERVILLE, 2011).

Nesse caso, se faz necessário buscar ferramentas que auxilie na documentação de APIs para que a manutenção e evolução do software seja de forma rápida e segura. Já que existem ferramentas de suporte para ajudar com a classificação e atualização da documentação. E visto que, a tarefa de manutenção de software é tão vital e complexa
que não pode mais ser feito de forma eficaz sem suporte automatizado (GRUBB, et. al., 2003, pg.314).

4. Objetivos

4.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver um conjunto de boas práticas de programação com ferramentas automatizadas para documentação de APIs buscando facilitar a manutenção e evolução da mesma durante o seu ciclo de vida.

4.2 Objetivos específicos

Para alcançar o objetivo geral foram definidos os seguintes objetivos específicos:

1. Identificar as principais ferramentas utilizadas para a documentação de APIs.
2. Fazer um levantamento sobre as boas práticas de documentação de APIs.
3. Aplicar as recomendações na implementação de uma API para exemplo.
4. Realizar uma pesquisa para avaliar a compreensibilidade da documentação da API para exemplo.

5. Metodologia da pesquisa

Para a realização deste trabalho foi desenvolvido um estudo exploratório focado na aplicação de boas práticas de documentação de APIs no contexto de sistemas web. Inicialmente foi necessário realizar um estudo na literatura para obter informações necessárias sobre as boas práticas de documentação de APIs. Em uma segunda etapa foi feito um levantamento sobre as ferramentas disponíveis para automação da documentação de APIs. Após a identificação de boas práticas de documentação, e caracterização das ferramentas disponíveis obtidas durante as fases iniciais foi construída uma documentação de uma API como estudo de caso. Essa documentação tem como objetivo facilitar tanto o entendimento de novos colaboradores quanto da equipe de desenvolvimento de uma ferramenta web. Por fim, a pesquisa realizou a aplicação de um questionário para avaliar a percepção de compreensibilidade da API desenvolvida a partir da visão de outros programadores que não eram integrantes da equipe do projeto.

6. Boas Práticas de documentação[1]

Documentar APIs corretamente é essencial para garantir que outras pessoas possam usar e entender facilmente suas funcionalidades, o que pode levar a uma maior adoção e sucesso do projeto, economizando tempo e reduzindo erros.

[1] Disponível em: https://appmaster.io/blog/tips-for-restful-api-documentation

Nesse contexto são listadas algumas das boas práticas de documentação de APIs que ajudam a tornar a documentação clara e fácil de usar:

- Uma descrição detalhada do que a técnica ou item faz
- Chamadas que transmitem detalhes cruciais para os desenvolvedores, como problemas e cuidados
- Uma chamada de exemplo com o conteúdo do tipo de mídia correspondente
- Uma lista de verificação das variáveis utilizadas por esta técnica, juntamente com informações sobre seus tipos, requisitos específicos de estruturação e se são necessários.
- Um exemplo de resposta com corpo de tipo de mídia
- Amostras de scripts em várias linguagens que contêm todo o código necessário (por exemplo, Java, .Net, Ruby, etc.)
- Uma introdução à API de REST
- Como obter as credenciais dos utilizadores
- Recursos necessários para a utilização do API
- Mensagens de erro ao comunicar com o API
- Termos de utilização
- Instâncias do SDK
- Eles demonstram como usar o SDK para seu dialeto para alcançar o serviço ou procedimento.
- Atividades valiosas para testar e experimentar solicitações de API (API Console, API Notebook)
- Consultas e situações com códigos são comumente questionadas.

Contudo devemos garantir que uma documentação de API use esses princípios para que no futuro não haja dificuldades de uso, pois os desenvolvedores terão que adivinhar como a API funciona ou tentar obter informações por conta própria. Isso pode levar a erros, retrabalho e uma experiência geral ruim e consequentemente limita a utilidade da API tornando menos valiosa para os usuários finais. Sem contar na dificuldade em manter e atualizar a API, pois sem uma documentação clara, pode ser difícil manter e atualizar a API com novos recursos ou correções de erros. Isso pode levar a uma API desatualizada e menos eficaz ao longo do tempo.

7. Ferramentas de api

As ferramentas de documentação de APIs têm se tornado cada vez mais importantes para a indústria de software, facilitando a documentação e a utilização de APIs por desenvolvedores permitindo a criação de documentações claras, objetivas e padronizadas.

Neste contexto, as ferramentas de documentação de APIs são uma excelente opção para os desenvolvedores economizarem tempo e manterem a documentação sempre atualizada. Além disso, essas ferramentas também oferecem recursos adicionais, como a capacidade de gerar código de cliente e interfaces de usuário personalizáveis. Há uma ampla variedade de ferramentas disponíveis no mercado, cada uma com suas próprias características e funcionalidades.

7.1 Swagger[2]

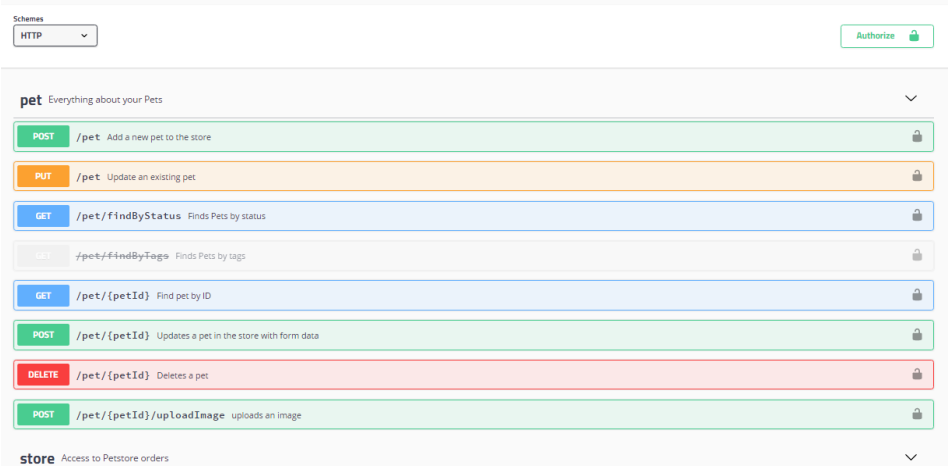
O swagger, agora conhecido como OpenAPI, é uma das ferramentas de documentação de APIs mais populares e amplamente utilizadas, além disso é uma ferramenta de código aberto que permite documentar, projetar e testar APIs RESTful a mesma é usada por desenvolvedores em todo o mundo para criar e manter a documentação de suas APIs de forma eficiente e precisa, ou seja, a documentação gerada é interativa e inclui detalhes sobre cada *endpoint* da API, como parâmetros e tipos de dados aceitos, bem como exemplos de solicitações e respostas, o que torna mais fácil para outros desenvolvedores entenderem e usarem a API, além disso o swagger é compatível com várias linguagens de programação, tornando-o uma opção viável para equipes de desenvolvimento que usam diferentes linguagens. Outro ponto que chama atenção no swagger é o fato de gerar a documentação das APIs automaticamente, o qual economiza tempo e esforço para os desenvolvedores. No entanto, a configuração inicial é uma tarefa complicada e demorada, especialmente se o desenvolvedor não tiver experiência anterior com a ferramenta.

Portanto, é possível concluir que o swagger é uma ferramenta essencial para o desenvolvimento de APIs RESTful, contribuindo para a eficiência e qualidade do trabalho dos desenvolvedores. Através da documentação automatizada e dos recursos de teste, é possível reduzir o tempo e esforço dedicados à criação e manutenção da documentação, além de garantir uma maior qualidade do software desenvolvido.

---

[2] Link para a ferramenta: https://swagger.io/tools/swagger-ui/

Figura 1 - Estrutura do swagger



Fonte: https://swagger.io/tools/swagger-ui/

7.2 Postman[3]

Uma das principais características do *postman* é a capacidade de criar coleções de API, que são grupos de solicitações relacionadas e que podem ser facilmente compartilhadas com outros membros da equipe, principalmente quando se trabalha em projetos de desenvolvimento de software que envolvem muitas APIs diferentes, ou seja, a ferramenta permite que os desenvolvedores gerenciam e documentam essas APIs de maneira eficiente e fácil, o que torna o processo de documentação e teste de APIs mais eficiente e colaborativo. Além disso, é possível configurar e enviar solicitações usando diferentes métodos HTTP, como *GET*, *POST*, *PUT*, *DELETE*, entre outros.

Outro recurso importante do *postman* é a sua capacidade de gerar documentação de API automaticamente. Com apenas alguns cliques, os desenvolvedores podem produzir documentação clara e completa para suas APIs, incluindo exemplos de solicitações, respostas parâmetros e outros detalhes importantes, tornando mais fácil para outros desenvolvedores entenderem como usar a API e acelerar o processo de integração com outros sistemas

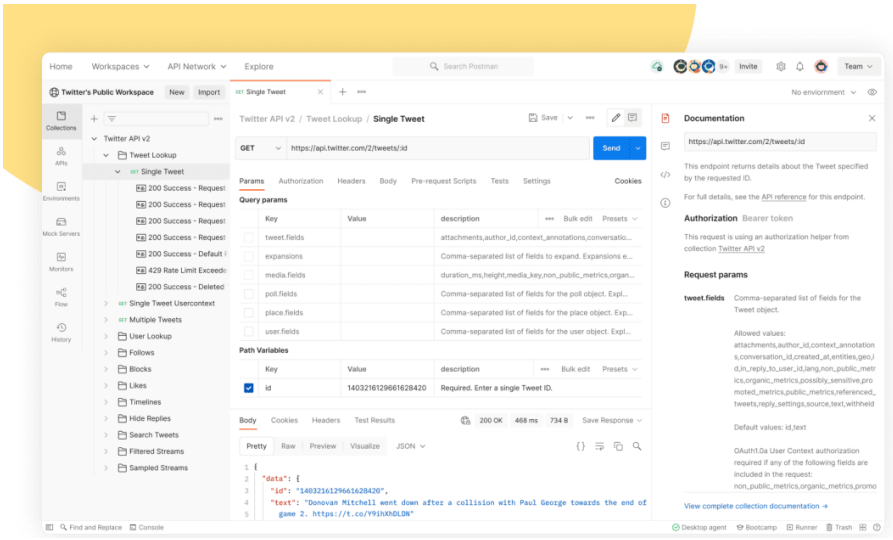
Além disso o *postman* possui uma interface fácil de usar, o que torna a criação e gerenciamento de solicitações e coleções de API uma tarefa fácil e intuitiva. Os usuários podem facilmente criar solicitações HTTP, definir cabeçalhos e parâmetros e definir variáveis tudo a partir da interface do *postman*.

Embora o *postman* possua várias vantagens, o mesmo sai sendo desvantajoso em alguns pontos como: a versão paga pode ser cara para equipes maiores e a versão completa requer uma assinatura paga, documentação limitada, pois apesar do *postman* gerar documentação

[3] Link para a ferramenta: https://www.postman.com/

automaticamente, ela pode ser limitada em recursos comparando com outras ferramentas de documentação de API. Uma imagem ilustrativa dessa ferramenta pode ser vista na figura 2.

Figura 2 – Estrutura do postman



Fonte: https://www.postman.com/

7.3 apiDoc[4]

O apiDoc é uma ferramenta poderosa que oferece importantes capacidades para desenvolvedores, permitindo anexar todas as versões anteriores e mais recente da API na documentação. Com essa funcionalidade, os desenvolvedores podem gerenciar melhor as mudanças em sua API, garantindo a integridade e confiabilidade do serviço.

Além disso, a capacidade de anexar uma versão à API torna mais fácil para os usuários entenderem quais alterações foram feitas e quando elas ocorreram. Isso é particularmente útil em ambientes em que a API é usada por vários desenvolvedores e equipes, ajudando a garantir que todos estejam na mesma página em relação às mudanças feitas na API.

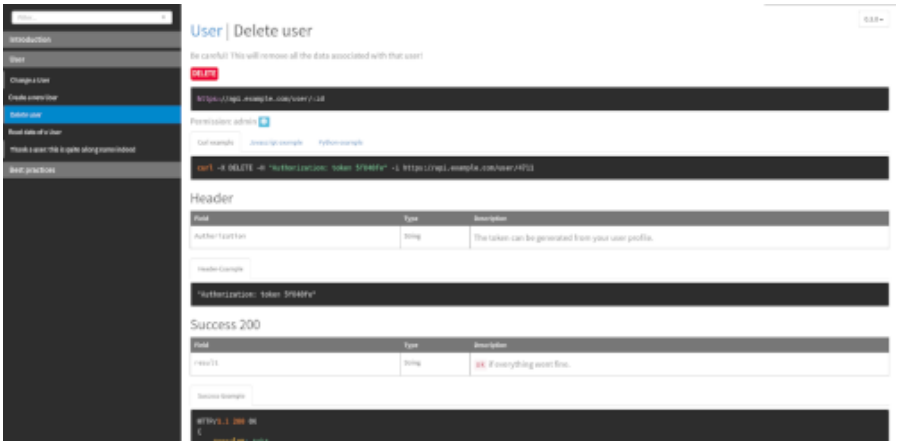
Outra vantagem importante do apiDoc é a possibilidade de usar herança, permitindo a criação de blocos de documentação genéricos, que podem ser compartilhados e adaptados em diferentes partes da aplicação, evitando a duplicação desnecessária de conteúdo. Além disso, o apiDoc oferece uma gama de anotações, permitindo definir métodos depreciados, descrições detalhadas sobre os métodos, mensagens de retorno de erro, nome do bloco de documentação, exemplo de documentação de parâmetro, entre outras funcionalidades.

No entanto, há algumas desvantagens a serem consideradas ao utilizar o apiDoc. A criação de um arquivo separado para criar a documentação pode ser um processo trabalhoso e

[4] Link para a ferramenta: https://apidocjs.com/

a herança só funciona com uma subclasse, pois pode tornar o código ilegível em casos complexos, mas que isso podia ser discutido para ser melhorado, além disso, as alterações na versão da API podem ser complexas, já que é necessário usar um arquivo separado e copiar toda a documentação antiga para o novo arquivo _apidoc.js.

Figura 3 - Estrutura do apiDoc



Fonte: https://apidocjs.com/

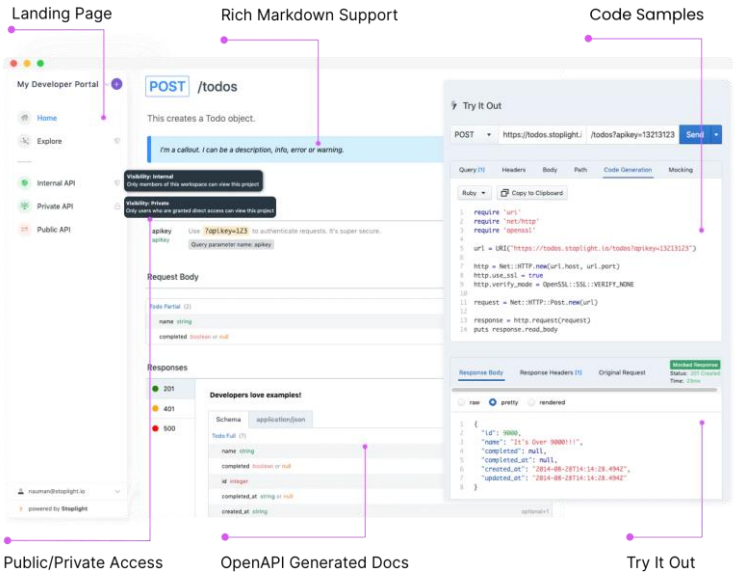
7.4 Stoplight[5]

O stoplight oferece uma interface intuitiva o qual se torna prático a localização de *endpoints*, documentação de referência e esquemas ao realizar uma pesquisa ampla do hub, essa ferramenta cria também boas experiências para os desenvolvedores, pois fornece uma documentação interativa, tutoriais e exemplos de código sempre atualizados nas linguagens populares como: python, ruby, java e muito mais, o qual permitem que os consumidores copiem e colem diretamente em seu próprio código. Um outro ponto que vale frisar dessa ferramenta é a visibilidade que ela possui, permitindo integração com o git a fim de manter toda a documentação atualizada conforme a atualização no código, dessa forma facilita o compartilhamento de APIs, o rastreamento de alterações, o gerenciamento de dependências e a criação de guias de estilo.

Apesar dessa ferramenta possui várias vantagens a mesma carrega alguns pontos negativos o qual pode acabar impedindo de algumas pessoas usarem, pois a mesma só é gratuita quando se trabalha individualmente, quando se deseja incluir mais de uma pessoa no time é necessário assinar uma licença para que possa usufruir dos seus recursos.

[5] Link para a ferramenta: https://stoplight.io/

Figura 4: Estrutura do stoplight



Fonte: https://stoplight.io/

7.5 Redocly[6]

Com o redocly é possível criar documentação automatizada de API, pois o workflows possui uma hospedagem rápida em um CDN global com invalidação instantânea de cache, além disso é possível realizar integração a ferramenta de versionamento em que é utilizado para o seu software. Um destaque importante é a possibilidade de visualizar os documentos a cada *pull request*, permitindo que a equipe revise, sugira melhorias e seja aplicada antes de enviar ao ambiente de produção garantindo maior qualidade e precisão, isso nos permite também ter sempre uma documentação atualizada e precisa.

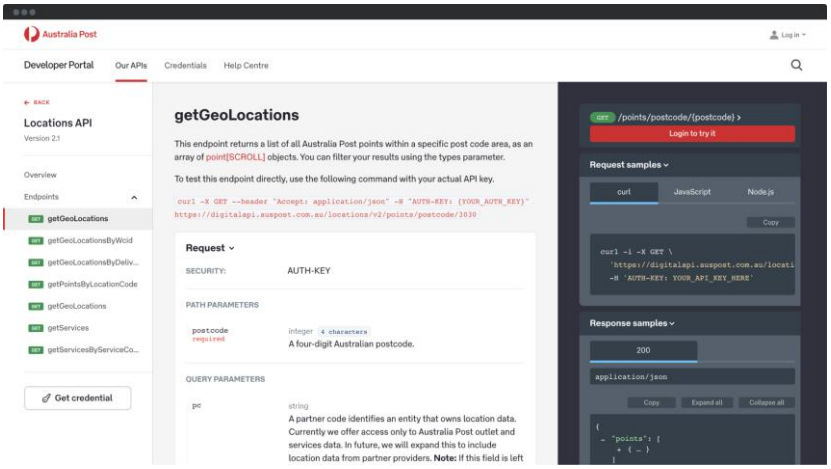
Além disso, o redocly oferece recursos de segurança para as documentações, como contas de visualizador, autenticação básica e de teste, e *login* via SSO (*Single Sign-On*). Esses mecanismos garantem um controle seguro do acesso aos documentos e um trabalho em equipe eficiente.

No entanto, é importante destacar que o redocly possui um plano gratuito apenas para usuários individuais. Para usufruir dos recursos disponíveis nos planos básico, profissional e de empreendedorismo, é necessário assiná-los.

---

[6] Link para a ferramenta: https://redocly.com/reference/

Figura 5: Estrutura do redocly



Fonte: https://redocly.com/reference/

8. Aplicação das boas prática de documentação

Após o levantamento das ferramentas de documentação utilizadas, foi desenvolvida uma documentação para a API de adoção de pets, levando em consideração os prós e contras das ferramentas avaliadas. Concluiu-se que a melhor opção seria utilizar o swagger para a documentação. Isso se deve ao fato de que o swagger se enquadra melhor nas boas práticas de documentação de API, além de permitir a geração automática da documentação, o que economizou tempo e esforço dos desenvolvedores envolvidos.

Apesar disso, a documentação gerada pelo swagger inclui informações sobre os *endpoints* disponíveis, os parâmetros de entrada e saída, os tipos de dados aceitos, além de apresentar-se os códigos HTTP tornando mais fácil para novos desenvolvedores que se integrem à equipe entenderem todos os *endpoints* mapeados, bem como para as pessoas que venham a consumir a API. Outro ponto que levou à escolha do swagger é sua compatibilidade com a linguagem de programação usada no desenvolvimento, como também, por ser uma ferramenta bastante conhecida e usada por várias empresas para documentar seus projetos. A aplicação dessa ferramenta pode ser vista em algumas imagens abaixo.

Por sua vez, na figura 6, pode ser vista o *endpoint* de cadastro de adotador, o qual espera receber os dados conforme mostrado na imagem, mas caso o e-mail não seja bem formatado o consumidor irá receber a mensagem de erro como mostra na figura 7.

Figura 6 - *Endpoint POST* do adotador

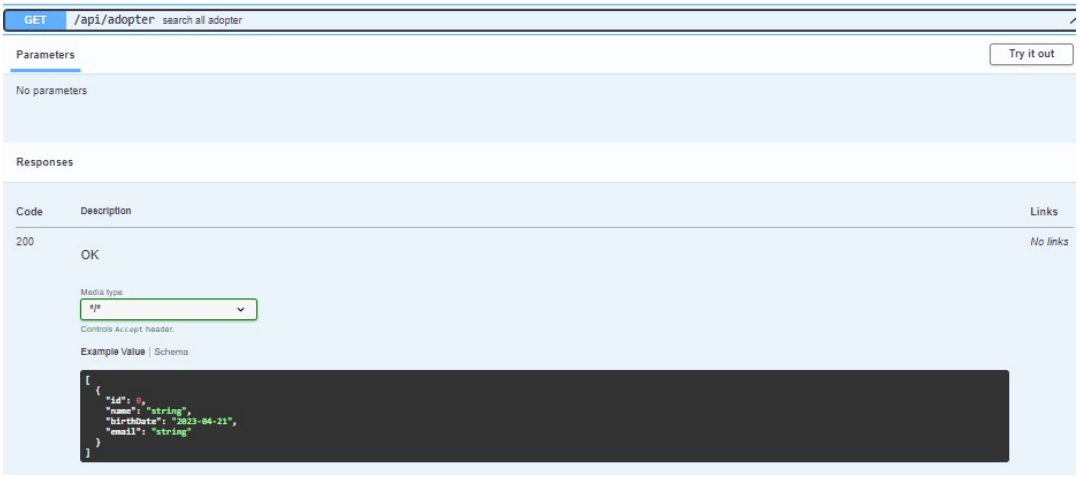
Fonte: Autoria própria(2023)

Figura 7- Erro no *endpoint* do *POST* de adotador

Fonte: Autoria própria(2023)

Na figura 8, é possível visualizar o *endpoint* responsável por listar todos os adoradores cadastrados até o momento, juntamente com um exemplo de saída que exibe os campos do doador, além das adoções realizadas. Como se trata de uma busca geral, não há necessidade de passar qualquer parâmetro.

Figura 8 - *Endpoint GET* de adotador



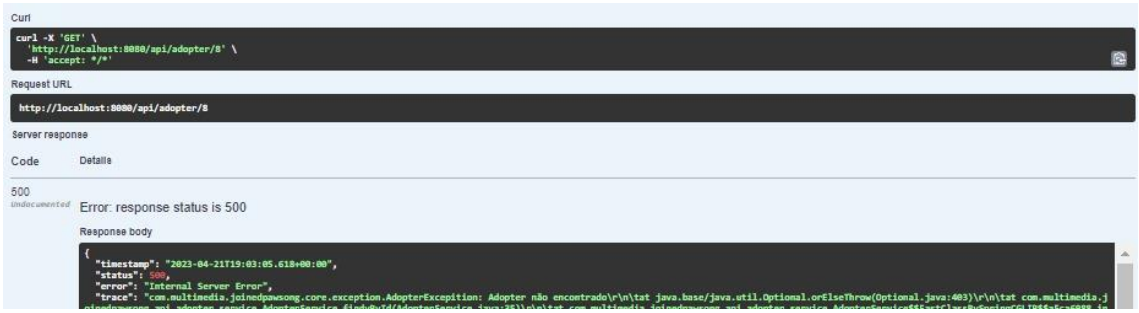
Fonte: Autoria própria(2023)

Na figura 9 é mostrado o *endpoint* que faz a busca por um adotador específico passando o seu id como parâmetro na url da requisição, como também o retorno da resposta. Mas, caso seja passado algum id inválido na busca, irá retornar um erro 500 para o usuário dizendo: "*Adopter* não encontrado", como mostra na figura 10.

Figura 9 - *Endpoint GET*/ api/*adopter*/{id} do adotador

Fonte: Autoria própria (2023)

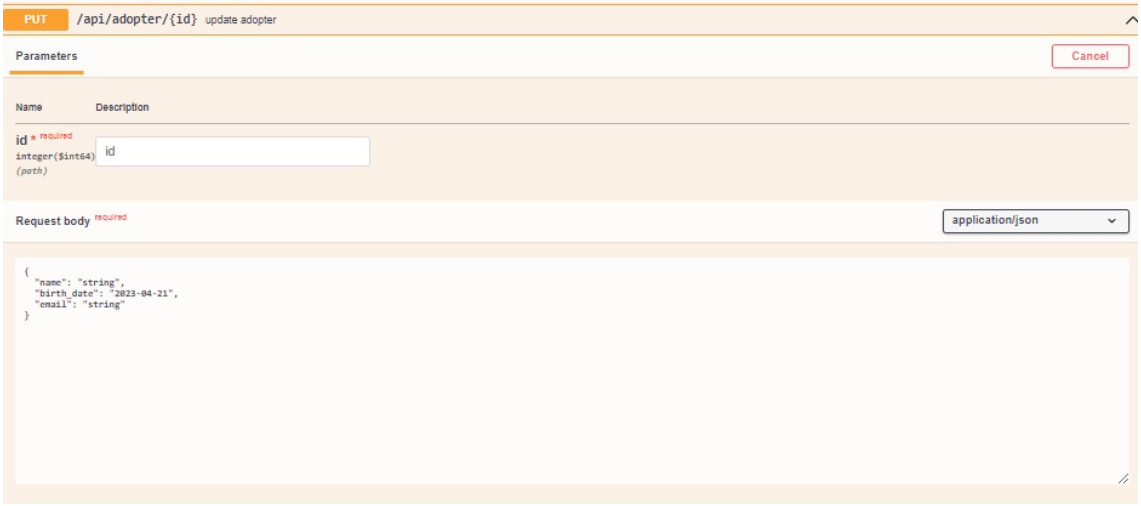
Figura 10 - Erro no *endpoint* GET /api/adopter/{id} do adotador



Fonte: Autoria própria (2023)

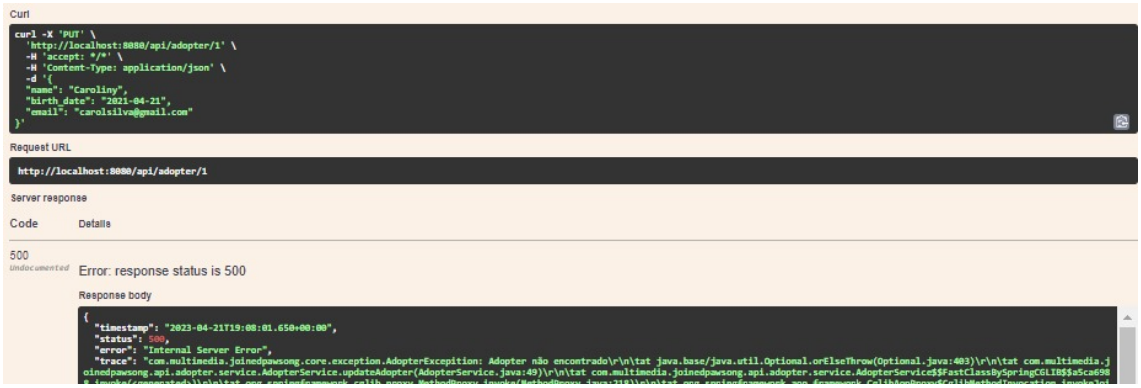
Na figura 11, apresenta-se o *endpoint* responsável pela atualização dos dados do adotador. Na imagem, é possível observar exemplos de JSON de entrada e saída. Além disso, o *endpoint* inclui uma validação do ID fornecido para verificar a existência do adotador correspondente. Se não houver correspondência, uma mensagem de erro é exibida, como ilustrado na Figura 12.

Figura 11- *Endpoint PUT* /api/*adopter*/ {id} do adotador



Fonte: Autoria própria(2023)

Figura 12 - Erro no *endpoint PUT*/ api/*adopter*/{id} do adotador



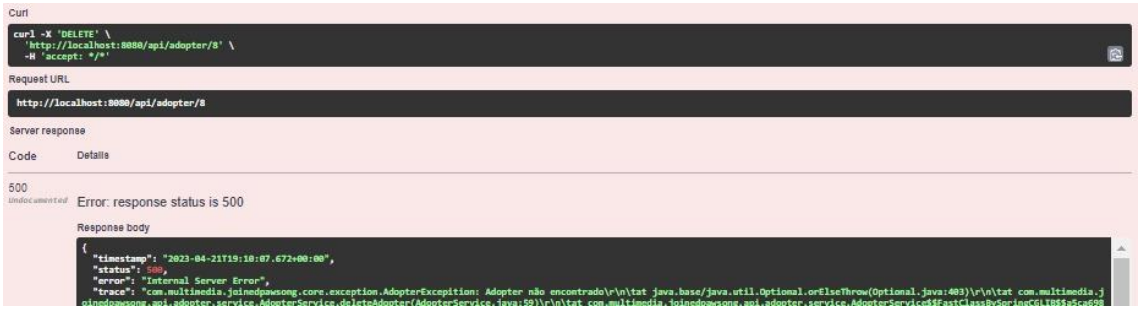
Fonte: Autoria própria (2023)

Conforme indicado na documentação da API, é possível utilizar uma requisição *DELETE* para excluir um adotador específico, como exemplificado na figura 13. Nesse caso, um valor é enviado como parâmetro, e uma verificação é realizada para exibir uma mensagem de erro caso não seja encontrado um adotador com o ID especificado na url conforme mostra a figura 14.

Figura 13 - *Endpoint delete* /api/*adopter*/{id} do adotador

Fonte: Autoria própria (2023)

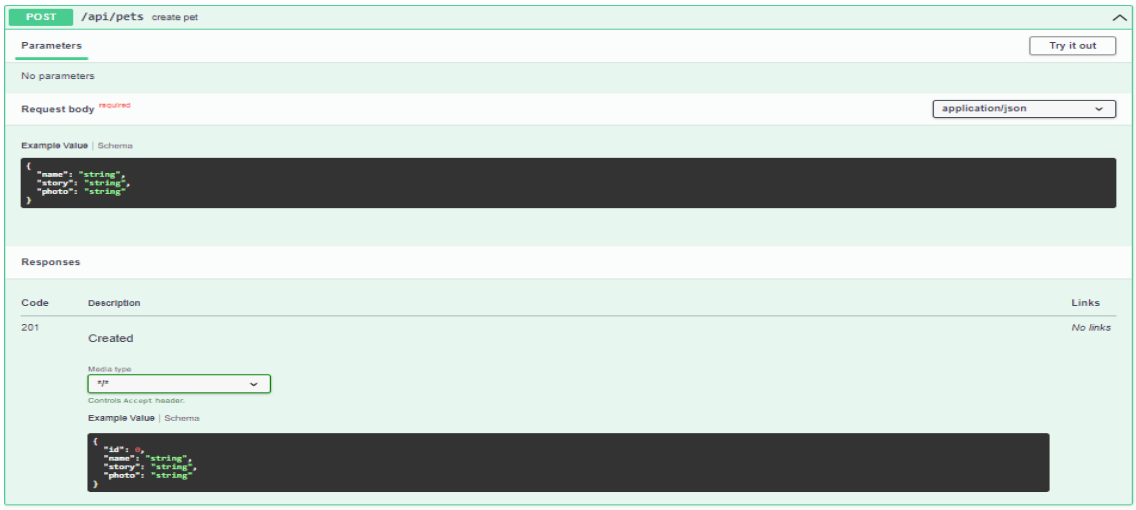
Figura 14 - Erro no *endpoint delete* /api/*adopter*/{id} do adotador



Fonte: Autoria própria (2023)

Na figura 15, é possível ver a requisição *POST* do *pet*, ou seja, *endpoint* responsável por realizar a criação dos pets, conforme mostra a figura esse *endpoint* recebe um JSON contendo três campos e o mesmo retorna um JSON com os valores informados anteriormente.

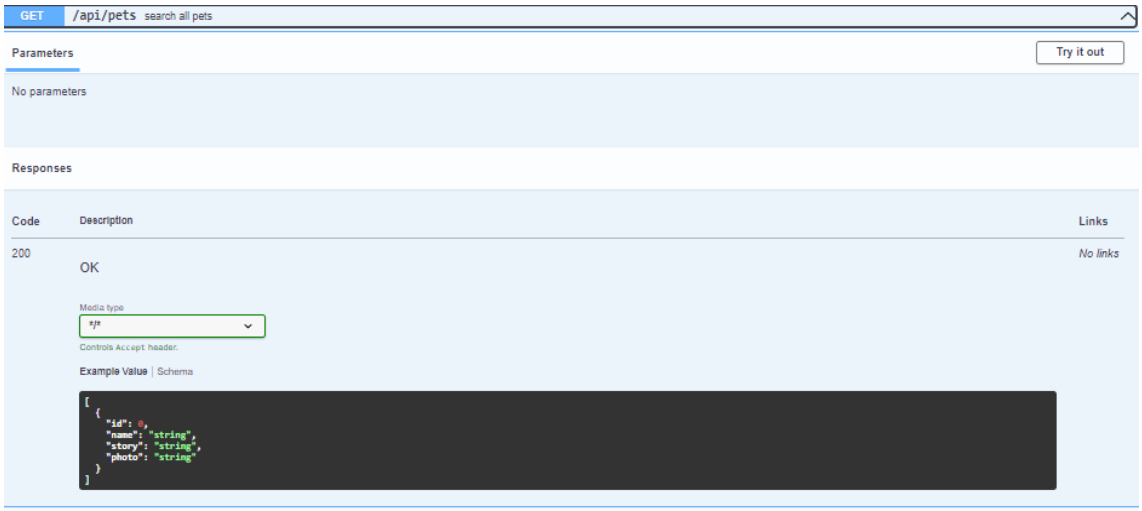
Figura 15 - *Endpoint POST* /api/pets de *pet*



Fonte: Autoria própria(2023)

Na figura 16, mostra o *endpoint* que realiza uma busca por todos os pets cadastrados até o momento, retornando um JSON com todas as informações que correspondem aos pets.

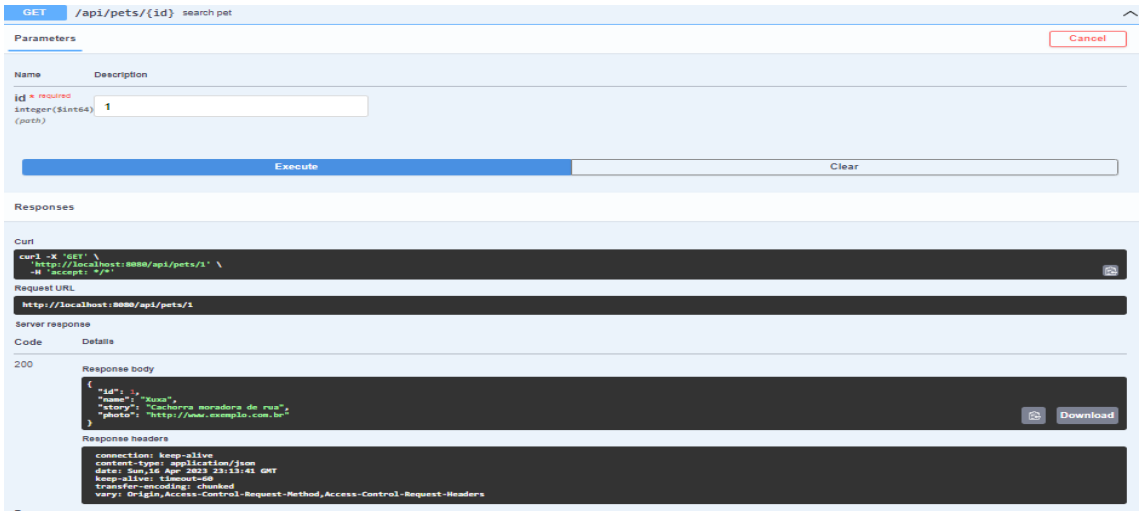
Figura 16- *Endpoint GET* /api/*pets* de *pet*



Fonte: Autoria própria (2023)

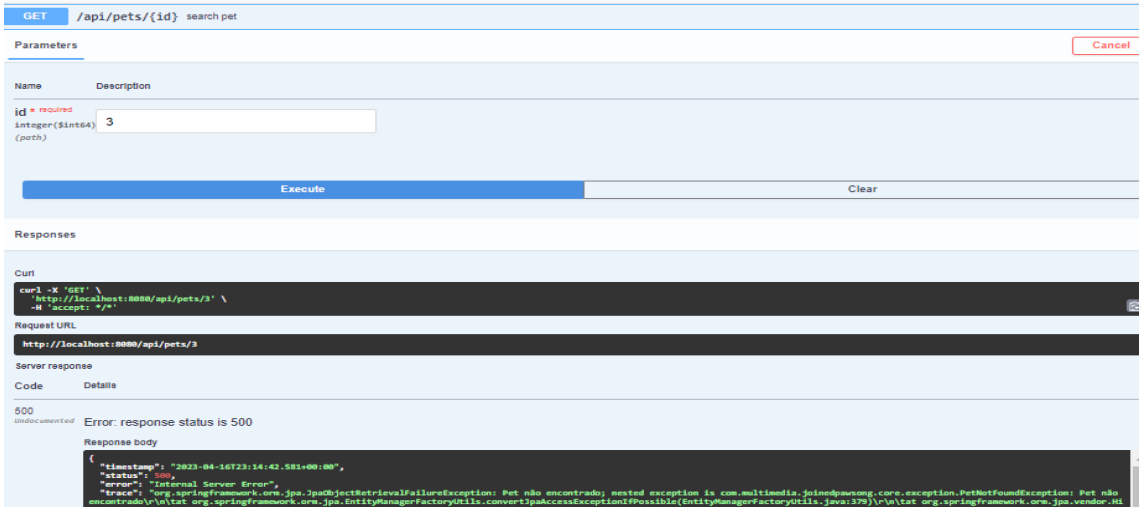
Na figura 17 é apresentada o *endpoint* que se responsabiliza pela busca de um determinado *pet* passando um id como parâmetro e como é realizado uma consulta no banco é feito uma verificação para saber se o id que está sendo passado realmente existe e caso não exista retorna uma mensagem de erro conforme pode ver na figura 18.

Figura 17- *Endpoint PUT* /api/*pets*/{id} de *pet*



Fonte: Autoria própria (2023)

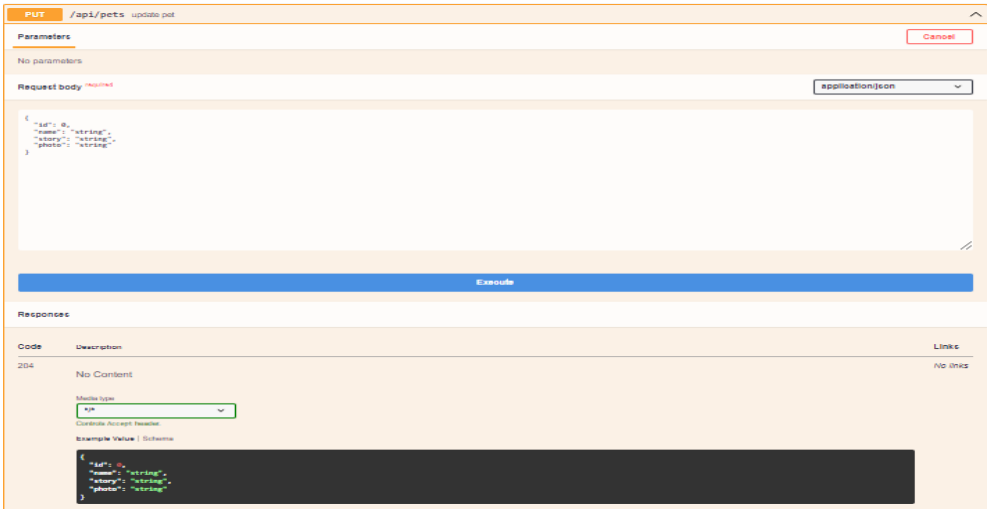
Figura 18- Erro no *endpoint PUT* /api/pets/{id} de *pet*



Fonte: Autoria própria (2023)

A API permite também fazer a atualização dos *pets* por meio de uma requisição *PUT*, conforme pode se observar na figura 19, e como é passado o id do *pet* também é feito uma verificação, Caso o ID não exista, uma mensagem de erro será exibida, conforme demonstrado na figura 20. Essa medida é importante para que o consumidor possa compreender o motivo pelo qual o objeto não foi atualizado.

Figura 19- *Endpoint PUT* /api/pets/{id} de *pet*



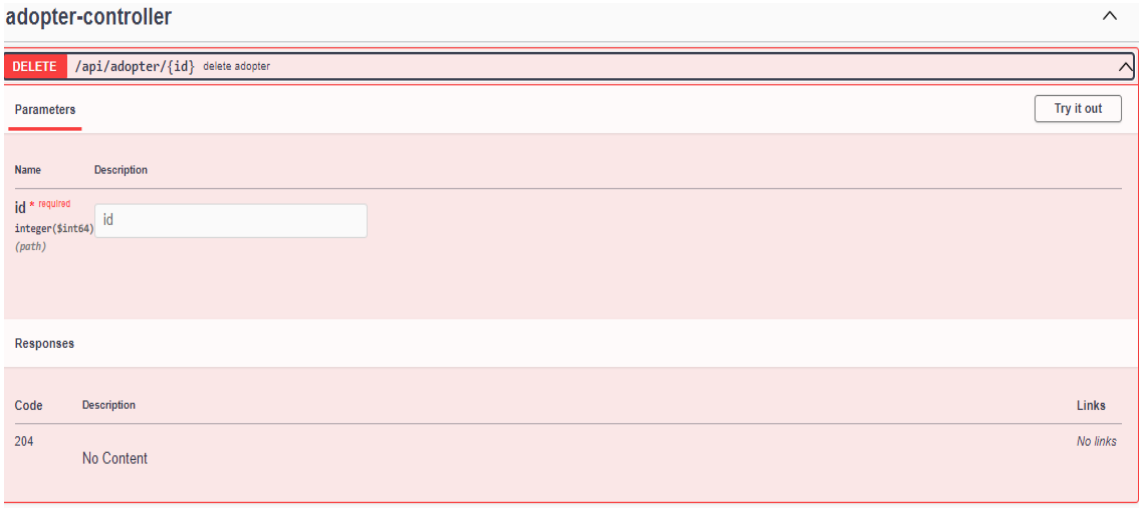
Fonte: Autoria própria (2023)

Figura 20 - Erro no *endpoint PUT* /api/pets/{id} de *pet*

Fonte: Autoria própria (2023)

Além disso é possível também realizar a exclusão do *pet*, por meio da requisição *delete*, conforme exemplificado na figura 21, e como é feita uma consulta no banco buscando pelo id informado no parâmetro, caso não seja encontrado nenhum registro correspondente, uma mensagem de erro é retornada, como apresentado na figura 22.

Figura 21- *Endpoint delete* /api/pets/{id} de *pet*



Fonte: Autoria própria (2023)

Figura 22 - Erro no *endpoint delete* /api/pets/{id} de *pet*

Fonte: Autoria própria(2023)

Como a API se trata de adoção de *pets*, a mesma possui também um *endpoint POST*, que se responsabiliza pela criação de adoção, onde espera receber no JSON o id do pet e do adotador e retorna um JSON contendo todos os dados do pet e adotador que foi passado no corpo da requisição como mostra na figura 23, e caso não exista os ids passado será retornado um erro conforme mostra a figura 24.

Figura 23 - *Endpoint POST* /api/*adoption* de adoção

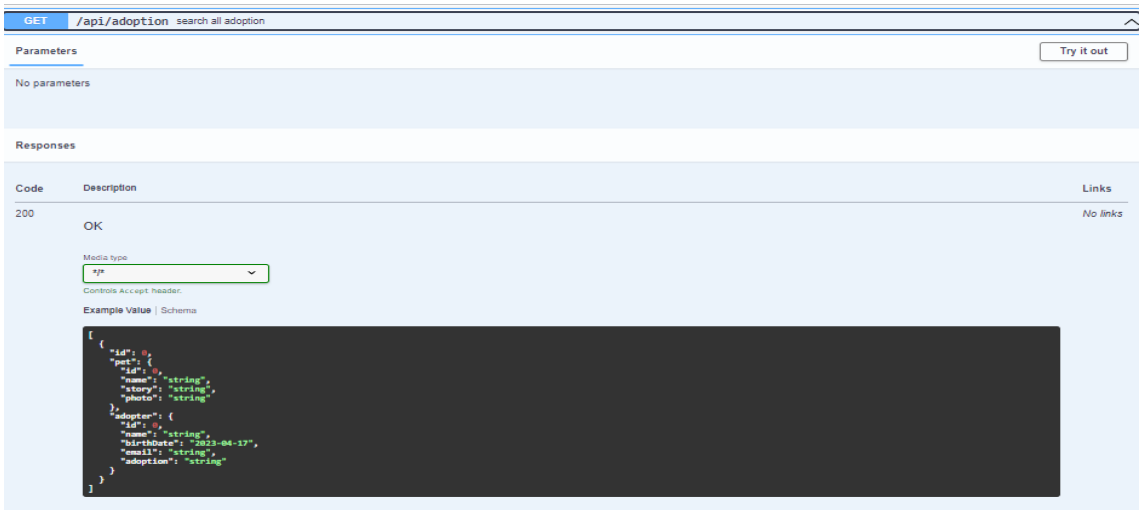
Fonte: Autoria própria(2023)

Figura 24 - Erro no *endpoint POST* /api/*adoption* de adoção

Fonte: Autoria própria(2023)

A aplicação permite também por meio de uma requisição *GET* listar todas as adoções que já foram cadastradas e isso pode ser visualizado na figura 25.

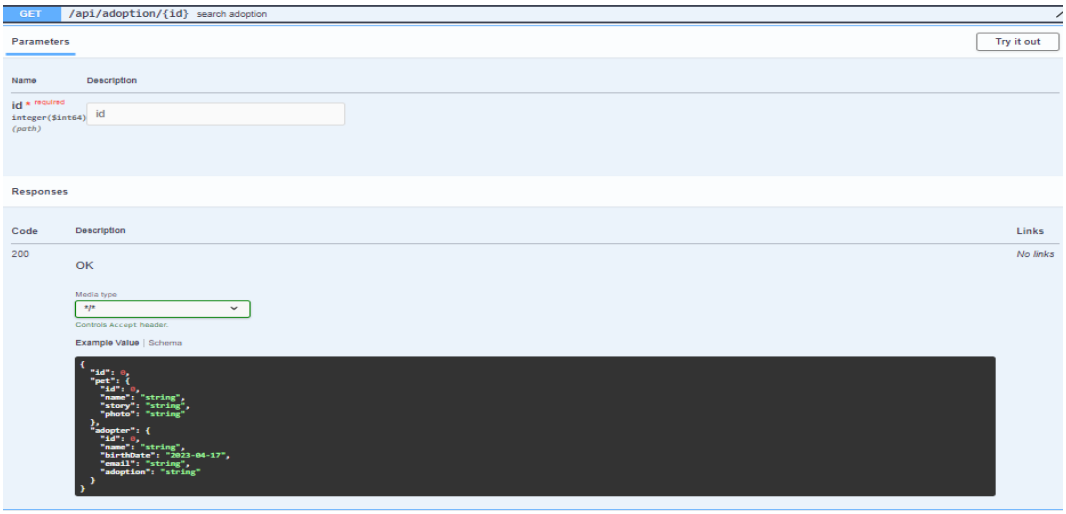
Figura 25 - *Endpoint GET* /api/*adoption*/{id} de adoção



Fonte: Autoria própria (2023)

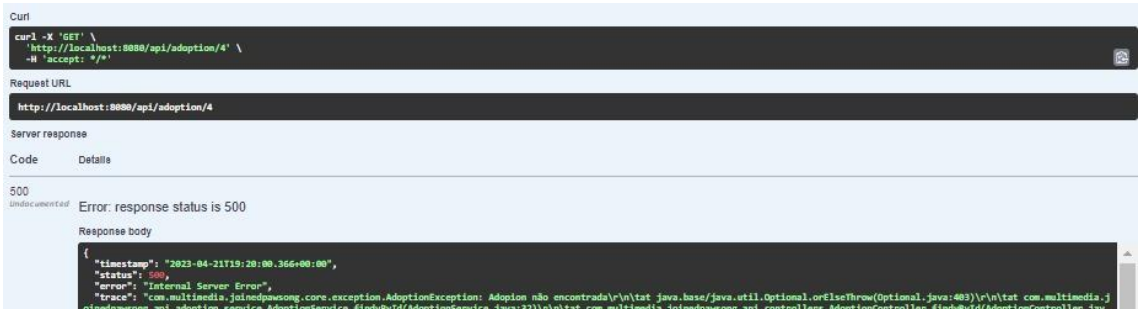
Além de permitir a listagem de todas as adoção é possível também realizar uma busca mais profunda, passando o id como parâmetro para buscar uma adoção específica e retornando um objeto de adoção e de *pet* como ilustra a figura 26 e caso não encontre a adoção com id passado o consumidor irá receber um erro com uma mensagem como mostra a figura 27.

Figura 26 - *Endpoint GET* /api/*adoption*/{id} de adoção



Fonte: Autoria própria (2023)

Figura 27 - Erro no *endpoint GET*/api/*adoption*/{id} de adoção



Fonte: Autoria própria (2023)

9. Estudo de Caso

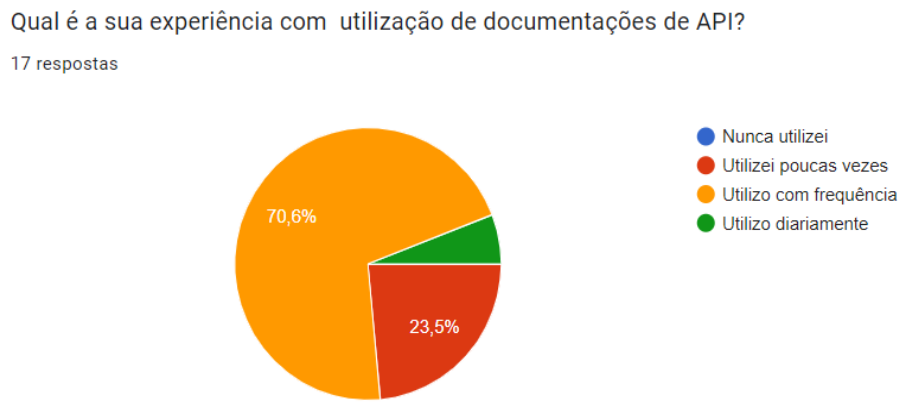
Foi disponibilizado um formulário de pesquisa do tipo questionário com alguns desenvolvedores com o intuito de identificar se os mesmos teriam alguma dificuldade em entender o objetivo da API como também efetuar manutenção mesmo tendo realizado toda documentação da aplicação. O questionário esteve disponível publicamente entre os dias 16/04/2023 e 20/04/2023.

9.1 Resultados e discussões

O formulário ao todo contou com uma quantidade 10 de perguntas, sendo 9 objetiva e 1 subjetiva. Essas perguntas foram de suma importância para validar a clareza e utilidade da documentação, identificar possíveis melhorias no código e até mesmo compartilhar conhecimento e boas práticas de documentação.

Sendo assim, a pesquisa obteve uma quantidade 17 respostas, e se tratando das experiências com documentação de APIs 70,6% diz que utiliza com frequência, 23,5% usa pouca vezes e 5,9% faz o uso todos os dias, conforme mostra o gráfico 1.

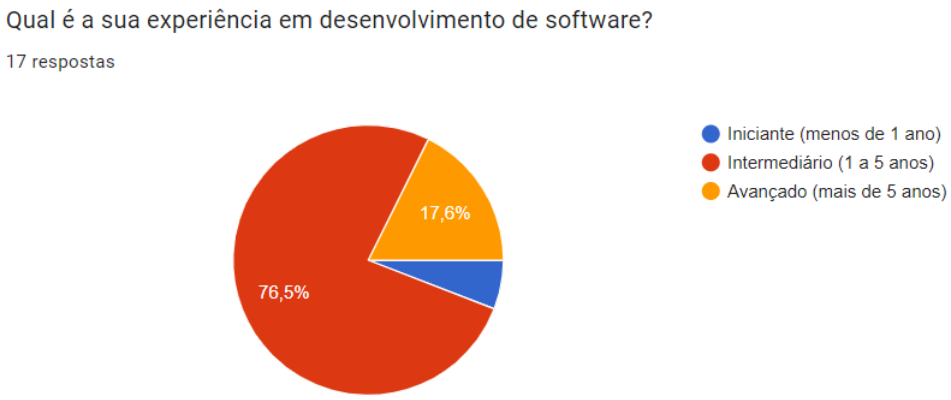
Gráfico 1 – Experiência com documentação de API



Fonte: Autoria própria (2023)

Outra pergunta que buscava identificar o tempo das pessoas no âmbito de desenvolvimento de *software*, cerca de 76,5% estão entre um e cinco anos de experiência, 17,6% com mais de cinco anos e 5,9% estão com menos de um ano. As respostas referente a esta pergunta podem ser visualizadas no gráfico 2.

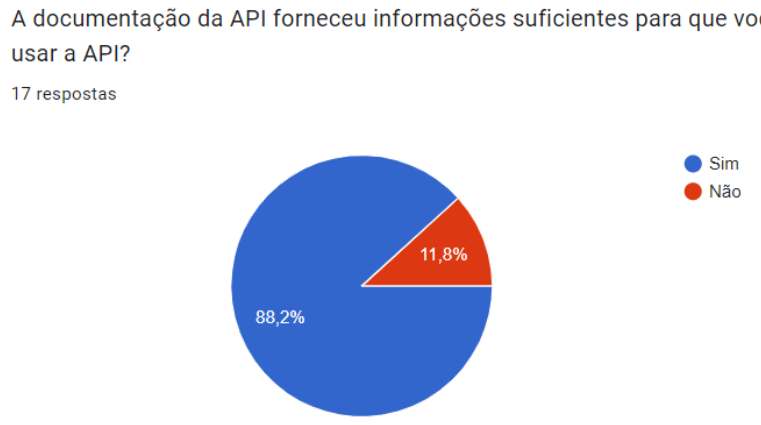
Gráfico 2 – Experiências com desenvolvimento de *software*



Fonte: Autoria própria (2023)

Com relação a documentação desenvolvida 88,2% responderam que a API forneceu informações suficientes para um bom uso e apenas 11,8% achou que não as informações não eram tão detalhadas, como pode ser visto no gráfico 3.

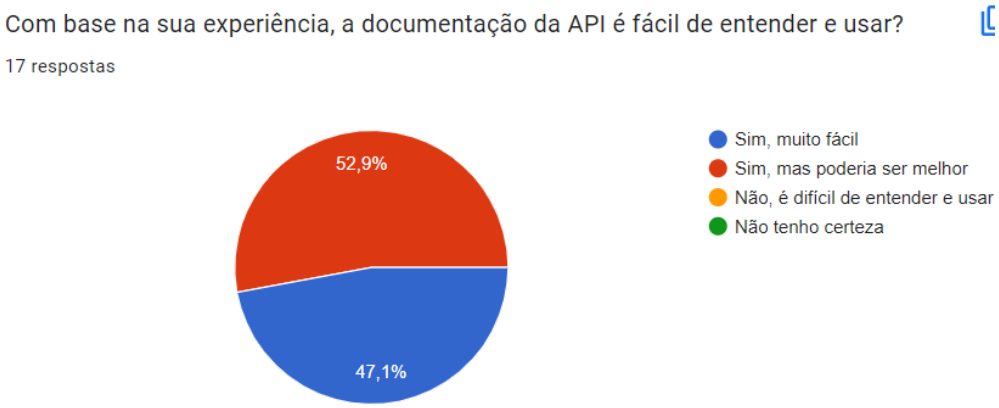
Gráfico 3 – Percentual sobre a utilização da API



Fonte: Autoria própria (2023)

Por outro lado, no gráfico 4, pode-se visualizar que 47,1% acharam que a documentação é muito fácil de se usar e 52,9% acharam que podia melhorar.

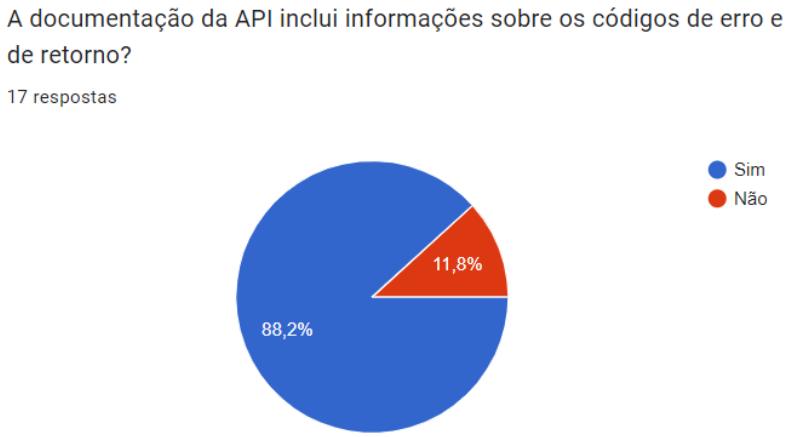
Gráfico 4 – Percentual sobre o entendimento da API



Fonte: Autoria própria (2023)

Além disso, a pesquisa buscou saber também se a documentação possuía informações sobre os códigos de erro e mensagens de retorno, 88,2% afirmaram que sim e 11,8% achou que não, esses resultados são ilustrados no gráfico 5.

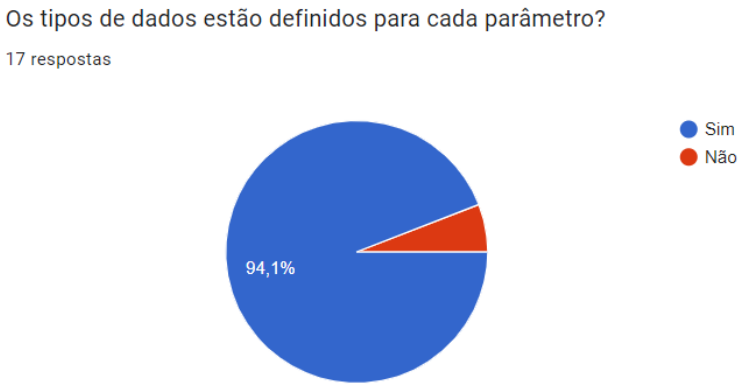
Gráfico 5 – Proporção sobre os códigos de erros e mensagens de retorno



Fonte: Autoria própria (2023)

Outra informação coletada, era se os tipos de dados estão definidos para cada parâmetro, e 94,1% confirmam que sim e apenas 5,9% discorda. Isso pode ser visto no gráfico 6.

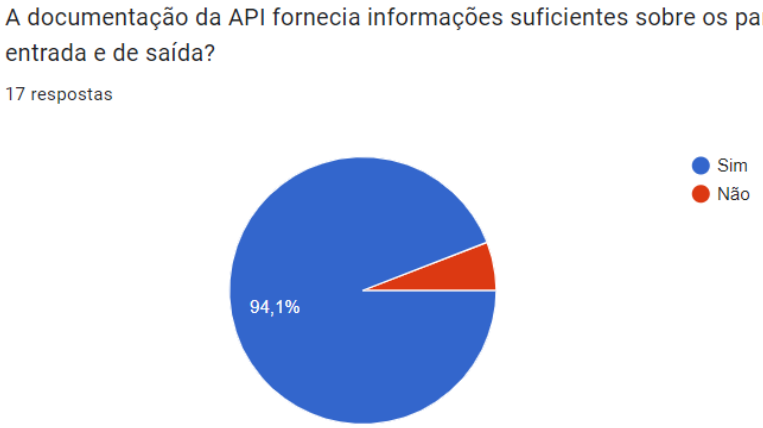
Gráfico 6 – Percentagem dos tipos de dados



Fonte: Autoria própria (2023)

Também foi feito um questionamento a respeito se a documentação da API fornece informações suficientes sobre os parâmetros de entrada e de saída. Para essa pergunta 94,1% afirmam que sim e 5,9% discordam, como pode ser visto no gráfico 7.

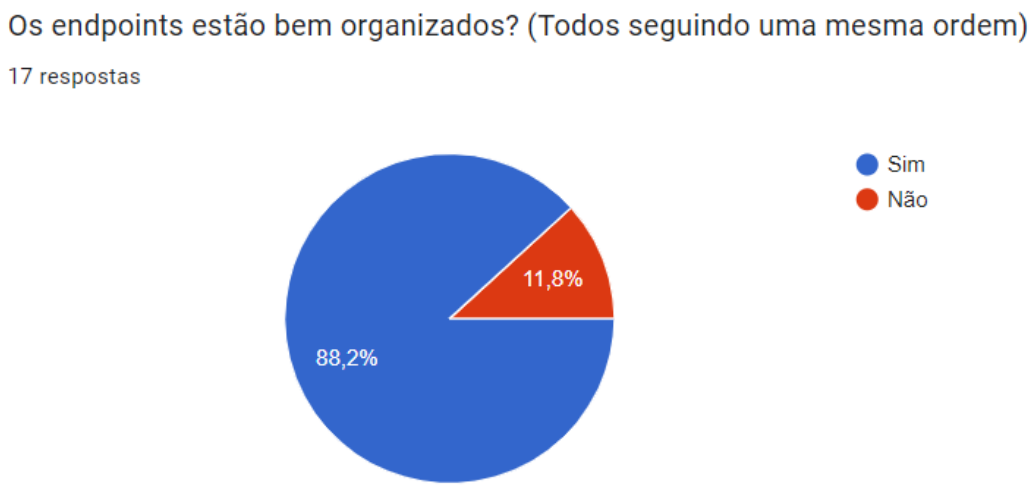
Gráfico 07 – Percentual sobre parâmetros de entrada e saída



Fonte: Autoria própria (2023)

O questionário buscou entender também se os *endpoints* estão bem organizados, 82,2% confirmaram que sim e 11,8% discordaram, prova disso pode ser visualizada no gráfico 8.

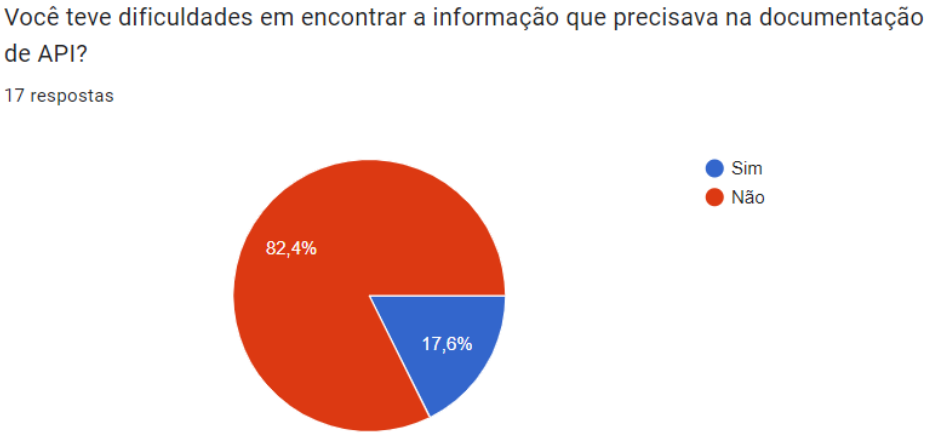
Gráfico 8 – *Endpoints* organizados



Fonte: Autoria própria (2023)

Um outro questionamento que fazia parte da pesquisa, era se tiveram dificuldades em encontrar a informação que precisavam na documentação de API, mas 80% afirmaram que não e 20% disseram que ainda tiveram dificuldades, isso pode-se visualizar no gráfico 9.

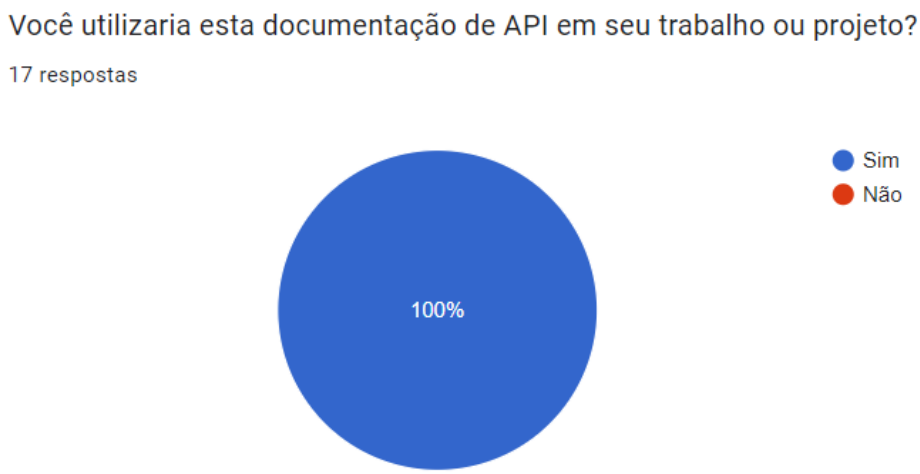
Gráfico 9 – Dificuldades em encontrar informações na documentação



Fonte: Autoria própria (2023)

E por fim a última pergunta, era se utilizariam a API para algum projeto, e como pode ser visto no gráfico 10, 100% responderam que sim.

Gráfico 10 – Percentual de utilização da documentação



Fonte: Autoria própria (2023)

Com isso, nota-se o quanto uma documentação atualizada, seguindo boas práticas é importante para o entendimento e interesse de desenvolvedores externos.

10. Considerações Finais

Em conclusão, este estudo destaca a importância de documentar APIs de forma clara e organizada, e apresenta uma série de boas práticas que podem ser utilizadas pelos desenvolvedores para melhorar a qualidade da documentação de suas aplicações. O uso dessas práticas pode aumentar a facilidade de uso das APIs, melhorar a experiência do desenvolvedor e reduzir a curva de aprendizado para o uso das APIs.

Além disso foram elencadas algumas ferramentas usadas para documentar API e por fim desenvolvida também uma aplicação simples e documentada usando as boas práticas de documentação em uma ferramenta automatizada.

Referências

SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

GRUBB, Penny; TAKANG, Armstrong A. Software maintenance: concepts and practice. World Scientific, 2003.

PRIKLADNICKI, Rafael. Problemas, Desafios e Abordagens do Processo de Desenvolvimento de Software. 2004. 69 f. Tese (Doutorado) - Curso de Ciência da Computação, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2004

TRIPATHY, Priyadarshi; NAIK, Kshirasagar. Software evolution and maintenance: a practitioner's approach. John Wiley & Sons, 2014.