UNIVERSIDADE LUTERANA DO BRASIL

Lucas Martins Tarachuck

BOAS PRÁTICAS NO DESENVOLVIMENTO DE APIs RESTful

Torres

2024

1. INTRODUÇÃO

Boas práticas no âmbito da programação, são práticas que deixam os códigos mais eficientes, fáceis de se manter e que facilitam na compreensão do código, por consequência deixando o código mais limpo e organizado. Logo as boas práticas do desenvolvimento de APIs RESTful segue o padrão de boas práticas citados anteriormente, porém também abordando o protocolo HTTP, com os métodos POST, GET, PUT, DELETE, entre outros métodos existentes.

2. Desenvolvimento.

2.1 MÉTODOS HTTP E O QUE É O REST

A API RESTful é como um Design Parttern, tendo a proposta de utilizar soluções reutilizáveis para que acelerar o processo de desenvolvimento, contando com um desenvolvimento bem testado, logo o padrão REST padroniza suas APIs, as tornando sua utilização mais homogênea. Os princípios do REST envolvem alguns ressources e cada tem que possuir sua própria URI, esses ressources são manipulados pelos métodos HTTP no qual cada um tem seu próprio significado e função, sendo eles: GET, POST, PUT, DELETE e PATCH. O método GET nunca deve criar, atualizar ou deletar um recurso, o método GET sempre tem o mesmo resultado para alguns conjuntos de informações, por exemplo da linha 15 a 19 de PedidoController onde é utilizado para que os objetos sejam listados posteriormente. O PUT pode atualizar recursos independente de quantas vezes for chamado, deve sempre ter como retorno 200 OK, tanto na primeira chamada após atualizar algo dentro do objeto que vai ser alterado, mas na segunda vez, caso esse objeto seja atualizado novamente para o mesmo nome atual, ele não será alterado, por exemplo: nome = “Lucas”, nome pode ser alterado para “Pedro”, logo nome = “Pedro”, nesse caso se quiserem alterar o nome Pedro para Pedro novamente, ele não será alterado. O DELETE é a mesma coisa que o PUT, porém ele deleta o objeto criado. O POST, toda vez que for chamado um recurso pode ser criado, e caso seja criado mais vezes, recursos similares vão ser criados todas essas vezes. O PATCH é parecido com o PUT, porém, o PATCH pode ser utilizado em chamadas de atualização podendo ter resultados diferentes

2.2 BOAS PRÁTICAS

Algumas das boas práticas mais utilizadas são: Usar Json para envio e recebimento de dados; usar substantivos ao invés verbos nos endpoints; nomear coleções com substantivos no plural; usar códigos de erro no tratamento de erros; usar aninhamento nos endpoints para mostrar suas relações; usar filtragem, ordenação e paginação para

obter dados solícitos; usar SSL para ter mais segurança; Ser claro com o controle de versão; Fornecer uma documentação de API adequada e precisa. Usar Json como formato para o envio e o recebimento de dados, ocorre porque Json é utilizado na maior parte das vezes para enviar dados em relação a APIs, antigamente era utilizado o formato XML, porém, era complicado de decodificar dados, cada linguagem de programação tem sua maneira de interpretar o Json, por exemplo em C# ao criar um projeto de webapi, dois arquivos em Json são criados, sendo eles: appsetting.Development.json e appsettings.json. Ao criar uma API se deve utilizar substantivos ao invés de verbos nos endpoints(endpoint é o local da API onde o sistema interage com uma API na web), acontece por conta que os métodos HTTP já usam uma forma verbal para serem feitas as operações do CRUD. Nomear coleções com substantivos no plural ajudam a não dar erro em alguns métodos como o DELETE por exemplo. Usar códigos de erro para corrigir outros erros ajuda os usuários a saberem o que está acontecendo e qual seu erro. Usar aninhamentos nos endpoints para mostrar as relações, endpoints diferentes podem estar ligados uns aos outros, por isso aninhá-los pode ajudar a compreender a ligação entre eles. Usar filtragem, ordenação e paginação para obter os dados solicitados, as vezes o banco de dados de uma API acaba ficando muito grande, deixando a obtenção de dados mais lento, então a filtragem, ordenação e paginação permitem que obter, ordenar e organizar os dados necessários para não sobrecarregar o banco de dados com solicitações. A utilização de SSL(secure socket layer) é fundamental para a segurança da API e garantirá que ela seja menos vulnerável a ataques, que é o que mostra em URLs e o HTTP se torna o HTTPS. Ser claro com o controle de versões, se dá pelo motivo de não forçar o usuário a trocar para versões mais novas, porque isso pode até quebrar a aplicação do mesmo, geralmente é utilizado no modelo semântico como 1.0, 1.1. Fornecer uma documentação de API adequada e precisa é para que facilite os usuários a compreenderem e utilizarem a API da maneira correta, sendo esse método da documentação extremamente eficaz.

3. CONCLUSÃO

As pesquisas feitas como base para o texto foram extremamente importantes para agregar num conhecimento sobre as boas práticas para o desenvolvimento de uma API RESTful, que deve possuir tudo do padrão REST para que seja considerada “RESTful”, algo também interessante, foi a explicação do porque o modelo JSON é mais utilizado para a compilação e leitura de dados, o uso do aninhamento de endpoints para que o banco de dados não seja sobrecarregado com várias requisições.

4. REFERÊNCIAS BIBLIOGRÁFICAS

https://reecodecamp.org/portuguese/news/melhores-praticas-para-apis-rest-exemplos-de-design-de-endpoints-rest/#:~:text=APIs%20REST%20devem%20ter%20versões,de%20controle%20de%20versão%20semântico.
https://www.brunobrito.net.br/api-restful-boas-praticas/
https://mailchimp.com/pt-br/resources/what-is-an-api-endpoint/#:~:text=O%20endpoint%20de%20API%20%C3%A9,de%20comunica%C3%A7%C3%A3o%20entre%20dois%20sistema