Versionamento de API

No desenvolvimento de software, o versionamento de APIs é uma prática essencial para a evolução e manutenção de aplicações. Ele permite que diferentes versões de uma API coexistam, garantindo a continuidade e a compatibilidade com versões anteriores. Ao versionar uma API, indicamos que novas funcionalidades, modificações ou remoções foram implementadas, utilizando técnicas específicas como a inclusão de números de versão na URL, parâmetros de controle ou cabeçalhos personalizados. A importância do versionamento reside na capacidade de manter a estabilidade das aplicações, assegurando que diferentes clientes possam utilizar a versão da API que melhor atende às suas necessidades. Nesta aula, vamos explorar o que é versionamento de API, como identificá-lo e compreender sua importância no desenvolvimento de software.

Definição de Versionamento de API

O versionamento de API é um processo crucial no desenvolvimento de software, especialmente quando estamos lidando com a evolução e manutenção de aplicações. Trata-se da prática de implementar e gerenciar mudanças nas APIs, garantindo que diferentes versões possam coexistir e funcionar corretamente sem quebrar a compatibilidade com versões anteriores.

Quando falamos de versionamento de API, nos referimos à necessidade de indicar que uma nova versão de uma API está disponível. Isso é feito através de técnicas específicas que ajudam a identificar qual versão da API está sendo usada. Exemplos comuns incluem a adição de um número de versão

na URL (como “/v2/”), o uso de parâmetros de controle ou o uso de cabeçalhos personalizados.

A importância do versionamento de API reside na capacidade de manter a estabilidade e a funcionalidade das aplicações ao longo do tempo. Quando implementamos novas funcionalidades, modificamos ou removemos partes da API, é fundamental separar essas mudanças em versões distintas. Isso permite que diferentes clientes e aplicações utilizem a versão da API que melhor atende às suas necessidades específicas.

Para ilustrar, imagine que você tem uma aplicação de banco que usa uma API para transações. Se uma nova funcionalidade for adicionada, como suporte a um novo tipo de transação, essa mudança pode ser introduzida na versão “v2” da API. Clientes que ainda utilizam a versão “v1” não serão afetados por essa mudança, garantindo continuidade e compatibilidade.

Documentação de APIs

A documentação de APIs é essencial para a implementação, teste e verificação de contratos e dependências de uma API. Uma boa documentação define claramente o contrato da API, especificando as entradas e saídas esperadas, o que facilita o trabalho dos desenvolvedores ao integrar e utilizar a API.

Existem várias formas de documentar uma API. Algumas das mais comuns incluem o JavaDoc, o Swagger (OpenAPI) e o Spring Rest Docs.

JavaDoc: Ferramenta utilizada para gerar documentação a partir do código fonte Java. Ele adiciona comentários no código que explicam a funcionalidade de métodos, classes e pacotes, e pode ser exportado para um documento HTML.

Swagger (OpenAPI): Uma especificação que documenta APIs RESTful de forma interativa. Ele permite a descrição detalhada de endpoints,

parâmetros, respostas e exemplos, facilitando a integração e testes de APIs.

Spring Rest Docs: Integrado ao ecossistema Spring, esta ferramenta gera documentação legível e interativa a partir de testes escritos, exportando a documentação para HTML ou outros formatos.

Uma boa documentação diminui o risco de erros e facilita a evolução e manutenção do código, além de ser fundamental para que o time de desenvolvimento entenda como interagir corretamente com a API.

Versionamento por URI

O versionamento por URI é uma prática de design onde a versão da API é incorporada na URL. Isso facilita a manutenção e garante a compatibilidade, permitindo que diferentes versões da API coexistam.

Para implementar essa prática, é comum adicionar um identificador de versão na URL, como "/v1/", "/v2/", etc. Por exemplo, se a versão inicial da sua API está em "v1", você pode criar uma nova versão "v2" para implementar novas funcionalidades ou modificações. Assim, usuários que ainda dependem da versão "v1" podem continuar a usá-la sem interrupções.

A estrutura do código geralmente envolve a criação de pacotes separados para cada versão, contendo as classes e serviços necessários. Isso evita a duplicação desnecessária de código e permite uma organização mais clara das diferentes versões da API.

Versionamento por Controle de Parâmetro

O versionamento por controle de parâmetro envolve a adição de um parâmetro na solicitação da API para especificar a versão desejada. Esse método é útil para manter a mesma URL base, adicionando apenas um parâmetro para indicar a versão.

Por exemplo, em um endpoint "/users", você pode adicionar um parâmetro "version" para distinguir entre diferentes versões da API: "/users?version=v2". O controlador verifica o valor do parâmetro e direciona a solicitação para a versão apropriada do serviço.

Essa prática mantém a URL mais limpa e pode ser mais segura, pois não expõe múltiplas URLs diferentes para a mesma funcionalidade. No entanto, exige atenção na implementação para garantir que a lógica de versionamento seja corretamente aplicada e testada.

Versionamento de Cabeçalho

O versionamento por cabeçalho utiliza um campo específico no cabeçalho da requisição HTTP para indicar a versão da API. Essa prática mantém a URL base limpa e centraliza a lógica de versionamento em um único lugar.

Um cabeçalho comum para versionamento é "Accept-Version", mas outros formatos como "X-API-Version" ou "API-Version" também são usados. Ao receber uma solicitação, a API verifica o valor do cabeçalho e direciona a solicitação para a versão apropriada do serviço.

Esse método facilita a atualização, pois os clientes não precisam modificar suas URLs, apenas o cabeçalho da requisição. No entanto, pode ser menos visível e mais difícil de identificar para desenvolvedores menos experientes.

Em resumo, o versionamento de API é fundamental para a evolução e manutenção de aplicações, garantindo compatibilidade e estabilidade ao longo do tempo. A escolha do método de versionamento depende das necessidades específicas do projeto e das preferências da equipe de desenvolvimento.

GitHub da Disciplina

Você pode acessar o repositório da disciplina no GitHub a partir do seguinte link:

https://github.com/FaculdadeDescomplica/Framework. Esse espaço é o seu portal para mergulhar fundo no universo da aprendizagem interativa. Nele, você encontrará todos os códigos, além dos links para os arquivos e dados.

Conteúdo Bônus

Um excelente conteúdo bônus gratuito para alunos de graduação que estão aprendendo sobre Frameworks e Versionamento de API é a documentação oficial do Spring Framework, que aborda boas práticas e métodos para versionamento de APIs RESTful.

Título: Building REST services with Spring

Plataforma: Spring.io

Este recurso oferece uma visão abrangente sobre como implementar e gerenciar o versionamento de APIs usando o Spring Framework. Os principais tópicos incluem:

Conceitos básicos de versionamento de API.

Métodos de versionamento de API, como versionamento de URI, versionamento de parâmetros de consulta, versionamento de cabeçalho e versionamento de mídia.

Exemplos práticos de implementação de cada método de versionamento no Spring Boot.

Boas práticas para manter a compatibilidade e facilitar a evolução das APIs.

A documentação é bem estruturada e fornece exemplos de código claros, ajudando os alunos a entender e aplicar o versionamento de APIs de forma eficaz em seus projetos.

Referência Bibliográfica

CARDOSO, L. da C. **Design de aplicativos**. Intersaberes: 2022

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7.ed. Pearson: 2018.

JOÃO, B. do N. **Usabilidade e interface homem-máquina**. Pearson: 2017

LEAL, G. C. L. **Linguagem, programação e banco de dados**: guia prático de aprendizagem. Intersaberes: 2015.

MEDEIROS, L. F. de. **Banco de dados**: princípios e prática. Intersaberes: 2013.

PUGA, S.; FRANÇA, E.; GOYA, M. **Banco de dados**: implementação em SQL, PL/SQL e Oracle 11g. Pearson: 2013.

SETZER, V. W.; SILVA, F. S. C. **Bancos de dados**. Blucher: 2005.

VICCI, C. (Org.). **Banco de dados**. Pearson: 2014.

Ir para exercício