Aula 4

Created	@February 6, 2024 8:56 PM
Data	@February 7, 2024
Status	Não iniciado
Tipo	Aula

Escrevendo sua primeira documentação de API (swagger)

A documentação de API desempenha um papel crucial no desenvolvimento de software, facilitando a compreensão e a utilização das interfaces de programação de aplicativos (APIs). O Swagger, agora conhecido como OpenAPI Specification, é uma ferramenta popular para criar documentação de API. Aqui estão algumas razões pelas quais a documentação de API com Swagger é importante:

1. **Compreensão Rápida:**
    - A documentação fornece uma visão rápida e clara sobre como usar a API, seus endpoints, parâmetros, e tipos de dados. Isso acelera a integração para desenvolvedores novos ou inexperientes com a API.
2. **Padronização:**
    - O Swagger/OAS define um padrão comum para a documentação de API, o que torna mais fácil para desenvolvedores, testadores e outros membros da equipe entenderem a API de maneira consistente.
3. **Teste e Desenvolvimento Eficientes:**
    - A documentação detalhada facilita a criação de testes automatizados e acelera o desenvolvimento, pois os desenvolvedores podem se referir à

documentação para entender como os endpoints devem ser chamados e como os dados devem ser estruturados.

4. **Colaboração:**
    - Facilita a colaboração entre equipes de desenvolvimento, pois todos podem se referir a uma fonte única e atualizada de informações sobre a API. Isso é especialmente importante em projetos grandes ou com equipes distribuídas.
5. **Automação da Geração de Código:**
    - Muitas ferramentas podem gerar automaticamente código cliente ou servidor com base em uma especificação Swagger/OAS. Isso reduz os erros e a carga de trabalho manual associada à implementação da API.
6. **Validação e Consistência:**
    - A documentação pode incluir exemplos de solicitações e respostas, ajudando os desenvolvedores a validar suas implementações e garantir consistência nos dados trocados pela API.
7. **Facilita a Adoção:**
    - Para desenvolvedores externos ou clientes que desejam integrar-se à sua API, a documentação Swagger torna o processo de adoção mais suave, pois fornece todas as informações necessárias.
8. **Manutenção Simplificada:**
    - Quando há atualizações na API, a documentação Swagger pode ser facilmente atualizada para refletir as mudanças. Isso garante que todos os usuários estejam cientes das alterações e possam ajustar suas implementações conforme necessário.

Em resumo, a documentação de API com Swagger/OAS é uma ferramenta essencial para garantir uma integração suave, colaboração eficiente e manutenção simplificada ao desenvolver e utilizar APIs.

O que não pode faltar em uma documentação swagger

Uma boa documentação Swagger deve ser abrangente e fácil de entender para os desenvolvedores que desejam interagir com a API. Aqui estão alguns elementos essenciais que não devem faltar em uma documentação Swagger:

1. **Informações Gerais:**
    - **Título e Descrição da API:** Descreva o propósito e a funcionalidade geral da API.
    - **Versão da API:** Indique a versão atual da API para garantir rastreabilidade.
2. **Endpoints:**
    - **Lista de Endpoints:** Enumere todos os endpoints disponíveis na API.
    - **Métodos HTTP:** Para cada endpoint, especifique os métodos HTTP suportados (GET, POST, PUT, DELETE, etc.).
3. **Parâmetros e Tipos de Dados:**
    - **Parâmetros de Requisição:** Liste e explique os parâmetros necessários para cada endpoint.
    - **Tipos de Dados:** Descreva os tipos de dados esperados nos parâmetros e nas respostas.
4. **Exemplos de Solicitações e Respostas:**
    - **Exemplos Claros:** Forneça exemplos de solicitações e respostas para cada endpoint para ilustrar como a API deve ser usada.
5. **Autenticação e Autorização:**
    - **Métodos de Autenticação:** Especifique como autenticar solicitações à API (por exemplo, OAuth, chave de API).
    - **Autorizações:** Explique os níveis de autorização e quais recursos estão disponíveis para diferentes usuários.
6. **Manipulação de Erros:**
    - **Códigos de Resposta:** Liste todos os códigos de resposta possíveis e seus significados.

- **Mensagens de Erro:** Forneça mensagens de erro claras para facilitar a resolução de problemas.

7. **Paginação e Ordenação:**
    - Se aplicável, explique como a paginação e a ordenação são tratadas nos resultados da API.
8. **Webhooks (se aplicável):**
    - Se a API suportar webhooks, forneça informações sobre como configurá-los e os eventos disponíveis.
9. **Segurança:**
    - Detalhe as práticas de segurança recomendadas, especialmente se a API envolver informações sensíveis.
10. **Licença e Termos de Uso:**
    - Inclua informações sobre a licença da API e quaisquer termos de uso ou restrições associadas.
11. **Referências e Links Úteis:**
    - Forneça links para qualquer documentação adicional, recursos relacionados ou ferramentas úteis para os desenvolvedores.
12. **Teste Interativo (Opcional):**
    - Se possível, inclua uma ferramenta de teste interativo na documentação, como o Swagger UI, para que os desenvolvedores possam experimentar chamadas de API diretamente na documentação.

Ao seguir essas diretrizes, você pode criar uma documentação Swagger abrangente que facilite a adoção da sua API e melhore a experiência do desenvolvedor.

Como implementar o swagger em uma aplicação Express.js

Para implementar o Swagger UI em uma aplicação Express.js, você pode seguir os passos abaixo. Certifique-se de ter o Node.js instalado e ter inicializado um projeto Express.js. Vamos usar o pacote `swagger-ui-express` para integrar o Swagger UI. Você pode instalá-lo usando o npm.

1. Instale os pacotes necessários:

```
bashCopy code
npm install express swagger-ui-express
```

1. Crie um arquivo para a definição Swagger (por exemplo, `swagger.json`) que descreve a sua API. Aqui está um exemplo básico:

```
jsonCopy code
{
  "swagger": "2.0",
  "info": {
    "title": "Minha API",
    "version": "1.0.0",
    "description": "Descrição da minha API"
  },
  "paths": {
    "/endpoint": {
      "get": {
        "summary": "Obter dados",
        "description": "Obtém dados do endpoint",
        "responses": {
          "200": {
            "description": "Sucesso",
            "schema": {
              "type": "object"
            }
          }
        }
      }
```

```
    }
  }
}
```

Substitua as informações acima pelos detalhes específicos da sua API.

1. No seu arquivo principal do aplicativo Express.js, configure o Swagger e o Swagger UI:

```
javascriptCopy code
const express = require('express');
const swaggerUi = require('swagger-ui-express');
const swaggerDocument = require('./swagger.json'); // Substit
ua pelo caminho do seu arquivo swagger.json

const app = express();
const PORT = 3000;

// Configuração do Swagger UI
app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swagger
Document));

// Restante da configuração da sua aplicação Express...

app.listen(PORT, () => {
  console.log(`Servidor Express rodando na porta ${PORT}`);
});
```

Este exemplo configura o Swagger UI no endpoint `/api-docs` e serve a documentação gerada a partir do arquivo `swagger.json`.

1. Execute o seu aplicativo:

```
bashCopy code
node seu-arquivo-principal.js
```

Agora, você pode acessar a documentação Swagger UI em http://localhost:3000/api-docs para visualizar e interagir com a sua API.

Lembre-se de ajustar as configurações conforme necessário para a sua aplicação e documentação específica.

Prático:

- Escrever Documentação completa da ultima API desenvolvida
- Configurar swagger-ui-express
- Implementar rota /docs para expor swagger

Referências:

https://editor.swagger.io/

https://swagger.io/docs/specification/basic-structure/