APIs REST

Labenu_

Sumário

Labenu_

O que vamos ver hoje? 👀

- Conceitos de APIs, Rest e GraphQL
- Criação de uma API REST com o Express
- Conceitos de CRUD na prática

Relembrando APIs

Labenu_

Relembrando APIs 🏗️

- Como vimos, uma API é uma **interface** de acesso à alguma fonte de dados, passando por certas regras. Ex: caixa de blocos com formas.

- Os tipos de API mais comuns são:
  - APIs REST
  - APIs GraphQL

APIs REST

- APIs **REST (Representational State Transfer)** são as que criamos até aqui, e são conhecidas por utilizar na maior parte das vezes os **verbos HTTP** para gerar a comunicação entre a aplicação e as fontes de dados.

- Cada endpoint trata de uma **entidade**, executando transações pré-estabelecidas pelos **métodos**.

- **GET**, **POST**, **PUT**, **PATCH**, **DELETE** são seus métodos mais comuns

APIs GraphQL

- APIs **GraphQL** são bem mais recentes, tendo sido criadas pelo Facebook em 2012, e são vistas como alternativa para APIs REST.

- Utilizam um único endpoint como uma **transação** de consulta ou alteração, normalmente atribuído como “/graphql”

- Utilizam **schemas**, **queries** e **mutations**.

APIs REST

Labenu_

Por que APIs REST? 🤔

- Maior maturidade, comunidade e suporte, diminuindo a curva de aprendizado (o conceito de Rest foi apresentado nos anos 2000 por Roy Fielding)

- Suporta múltiplos tipos de retorno (**JSON**, XML, YAML)

- Estruturas mais bem definidas de transação devido aos múltiplos endpoints

Entidades de uma API

Labenu_

Entidades da API 🛣️

- Uma entidade em uma API é um caminho para acessar as tabelas no banco.

- No **express**, a representação de uma entidade se dá pelo *path* que passamos em cada endpoint

```
app.get("/users", (req: Request, res: Response) => {

  //código para buscar todos os usuários
});
```

Neste exemplo, a **entidade** é users e o **método** é get

Entidades da API 🛣️

- Uma boa prática é usar o nome da entidade, trocando somente o método na hora de criar um endpoint. É comum também ver o nome da entidade no singular.

```
app.get(“/users”, (req: Request, res: Response) => {

  //código para buscar todos os usuários
});
```

```
app.put(“/users”, (req: Request, res: Response) => {

  //código para criar um usuário
});
```

Entidades da API 🛣️

- Caso a gente tenha alguma entidade dentro de outra, podemos adicionar valores ao caminho. Aqui vemos um exemplo de como buscar pedidos de um usuário específico.

```
app.get(“/users/:id/orders”, (req: Request, res: Response) => {

  //código para buscar todos os pedidos de um usuário por sua id
});
```

Entidades da API 

- O express suporta múltiplas entidades sendo manipuladas ao mesmo tempo. Isto é, endpoints que possuem caminhos diferentes e podem ser agrupados. Nesses casos, o agrupamento costuma ser por:
  - **Entidade**
  - **Método**

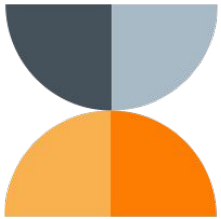
```
app.get("/users", (req: Request, res: Response) => {

  //código para buscar todos os usuários
});

app.get("/users/:id", (req: Request, res: Response) => {

  //código para buscar usuários por sua id
});

app.get("/users/:id/orders", (req: Request, res: Response) => {

  //código para buscar todos pedidos de um usuário por sua id
});

app.get("/stores", (req: Request, res: Response) => {

  //código para buscar todas as lojas
});

app.put("/stores", (req: Request, res: Response) => {

  //código para adicionar lojas
});
```

Exemplo - Entidades

Vamos criar um endpoint que busca informações de uma entidade Users. Para isso vamos usar:

	PATH	*POSTMAN*
- **Query** parameters	**“/users”**	**“/users?id=1”**
- **Path** parameters	**“/users/:id”**	**“/users/1”**

Labenu_



- APIs são interfaces de acesso aos dados, e normalmente são **REST** ou **GraphQL**
- **Entidades** de uma API são as áreas que manipulamos na aplicação
- Costumamos organizar nossos endpoints por **entidade** e **método**.

Métodos de uma API

Labenu_

Métodos da API

- O protocolo HTTP nos fornece **verbos/métodos** que são designados para executar funções específicas dentro da aplicação, seguindo uma convenção de boas práticas

- Os verbos/métodos não influenciam no comportamento do código, eles servem apenas como padronização de uso

- Os verbos/métodos que mais utilizamos são:
  - **GET** - Busca recursos
  - **PUT** - Atualiza recursos (cria se não existir)
  - **PATCH** - Atualiza parte dos recursos
  - **POST** - Cria novos recursos
  - **DELETE** - Remove recursos

Métodos da API

- O método **GET** é utilizado para buscar informações e não utiliza **body** em sua requisição, então todos os dados necessários na busca devem ser passados por:
    - **PathParams**, em caso de continuação de caminho, como no exemplo de pedidos de um usuário específico;
    - **QueryParams**, em caso de busca dentro de um mesmo conjunto, como em uma busca por nome;
    - **Headers**, em caso de informações externas à consulta, como informações sobre o dispositivo ou autenticação.

Métodos da API

- Os métodos **POST**, **PUT** e **PATCH** são utilizados para adicionar ou atualizar informações na entidade. No backend costumamos utilizar o seguinte padrão (exemplo no caso de entidade = “/users”):
    - **PUT** para modificar completamente um recurso existente;
    - **PATCH** para atualizar uma ou mais informações de um recurso existente (email, senha, etc);
    - **POST** para criar um novo recurso.

Métodos da API

- O método **DELETE** é utilizado para remover itens da entidade.

- Apesar de poder utilizar o body, o mais comum é que o identificador do que vai ser removido seja passado como PathParam (“/users/:id”).

Métodos da API

- Ao conjunto de operações possíveis de serem criados em uma API, damos o nome de **CRUD**.

    - **C**reate
    - **R**ead
    - **U**pdate
    - **D**elete

- Onde **Create** são nossos POSTs, **Read** são os GETs, **Update** são PUTs e PATCHes e **Delete** são DELETEs.

Exemplo - métodos

Vamos criar um novo item na nossa lista de Users e editá-lo posteriormente. Para isso, utilizaremos:

- **POST**
- **PATCH**

Extra

Labenu_

Rodando arquivos com ts-node e ts-node-dev

Uma biblioteca muito legal do *npm* é o **ts-node** e sua versão de desenvolvimento, o **ts-node-dev**. Com estas bibliotecas você pode rodar sua aplicação sem gerar o arquivo js diretamente na pasta.
É importante ressaltar que na hora de fazer o *build* ainda precisaremos fazer a transpilação com o tsc

```
npm install ts-node
npm install ts-node-dev --save-dev
```

```
"scripts": {
   "start": "ts-node index.ts",
   "build": "tsc"
   "dev": "ts-node-dev index.ts"
 }
```

Resumo

Labenu_

Resumo

- APIs são interfaces de acesso aos dados, e normalmente são **REST** ou **GraphQL**

- **Entidades** de uma API são as áreas que manipulamos na aplicação

- Costumamos organizar nossos endpoints por **entidade** e **método**.

Resumo

- Uma **API REST** utiliza os **métodos HTTP** para receber e executar os comandos;
- O verbo **GET** é usado para consultas e não utiliza **body** em sua requisição;
- Os verbos **POST, PATCH e PUT** são utilizados para inserir, atualizar ou substituir dados, respectivamente;
- O verbo **DELETE** remove itens, passando valores pelo **path**

Dúvidas?

Obrigado!