Revisão da Semana - Ambientes de Backend

Esse material tem como **objetivo** te dar uma orientação para realizar uma breve revisão sobre como utilizar as novas features de back aprendidas no último bloco

Materiais Complementares

▶ PT

▶ EN

Materiais de Revisão 👁

▼ Express.js

Express.js

Express é uma lib que permite receber as requisições através do protocolo HTTP, vindas de algum frontend.

Assim como o Axios e o próprio React, o Express é uma biblioteca de código JS, adotada por simplificar o uso de seus recursos nativos.

Adicionando o Express como dependência

Comando: `npm install express`

Como nossos programas serão escritos em Typescript, devemos também adicionar a versão tipada dessa biblioteca como dependência de desenvolvimento:

Comando: `npm install @types/express --save-dev`

Cors

A lib cors possibilita que qualquer frontend acesse nossa aplicação. (para fins educativos)

Comando: `npm install cors @types/cors--save-dev`

Criando o servidor

```
import express from 'express'
import cors from 'cors'
const app = express()
app.use(express.json())
app.use(cors())
```

Iniciando o servidor

```
app.listen(3003, () => {
 console.log("Server is running in http://localhost:3003");
```

```
});
```

Sintaxe dos endpoints

```
app.get('/path', (req: Request, res: Response) => {
 res.send('Hello, world!')
})
```

Request e Response

As principais propriedades do parâmetro req representam as diferentes formas pelas quais o front pode enviar dados:

- req.headers - Parâmetros de cabeçalho
- req.body - Parâmetros de corpo
- req.query - Parâmetros de consulta
- req.params - Parâmetros de caminho

De maneira análoga, os principais métodos do parâmetro res são:

- res.status(n) - envia uma resposta com status n
- res.send(x) - responde a requisição com o objeto x
- res.end() - encerra a requisição sem um body na resposta

Etapas do fluxo de dados do backend

1. Validação das entradas da requisição
2. Consulta à base de dados
3. Validação dos resultados da consulta
4. Envio da resposta

▼ API REST

API 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.

Entidades

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

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

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

Métodos da API

GET - Busca recursos
PUT - Atualiza recursos (cria se não existir)
PATCH - Atualiza parte dos recursos
POST - Cria novos recursos
DELETE - Remove recursos

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.

Os métodos POST, PUT e PATCH são utilizados para adicionar ou atualizar informações na entidade.

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.

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 Path Param (“/users/:id”).