Aula 3

Created	@October 28, 2023 11:22 AM
Data	@March 4, 2024
Status	Não iniciado
Tipo	Aula

Implementando e testando sua primeira API (O Padrão REST)

CRUD vs REST: Qual a diferença?;

CRUD (Create, Read, Update, Delete) e REST (Representational State Transfer) são conceitos relacionados, mas não são diretamente comparáveis, pois abordam diferentes aspectos no desenvolvimento de software.

CRUD (Create, Read, Update, Delete):

- **Definição:**
  - CRUD é uma abordagem padrão para as operações básicas de manipulação de dados em um sistema.
  - Representa as quatro operações principais que podem ser realizadas em dados: criar (Create), ler (Read), atualizar (Update) e excluir (Delete).
- **Objetivo:**
  - O CRUD foca nas operações básicas de um sistema que envolvem a persistência de dados.
- **Exemplo:**

- Um sistema de gerenciamento de usuários pode ter operações CRUD para adicionar um novo usuário, recuperar informações de um usuário, atualizar os dados de um usuário existente e excluir um usuário.

REST (Representational State Transfer):

- **Definição:**
  - REST é um estilo arquitetural que define um conjunto de restrições e princípios para o design de sistemas distribuídos.
  - Ele não é um protocolo, mas uma abordagem que utiliza os métodos e princípios do HTTP para criar serviços web escaláveis e interoperáveis.
- **Objetivo:**
  - O REST visa criar serviços web que são stateless (sem estado), escaláveis, e que seguem princípios como a representação de recursos e a utilização de métodos HTTP padrão.
- **Princípios do REST:**
  - **Recursos (Resources):** Tudo é considerado um recurso, e cada recurso é identificado por uma URI única.
  - **Representação:** Os recursos podem ter diferentes representações (JSON, XML, etc.).
  - **Métodos HTTP:** Utilização dos métodos HTTP padrão (GET, POST, PUT, DELETE) para operações em recursos.
  - **Stateless (Sem Estado):** Cada requisição do cliente contém toda a informação necessária; a sessão do cliente é mantida no cliente, não no servidor.
- **Exemplo:**
  - Uma API RESTful para um sistema de blog pode ter URIs como `/posts` para listar todos os posts, `/posts/{id}` para obter um post específico, e métodos HTTP como GET, POST, PUT e DELETE para manipular esses recursos.

Diferenças:

- **Escopo:**

    - CRUD é mais amplo e representa um conjunto básico de operações de manipulação de dados em um sistema.
    - REST é uma abordagem arquitetural específica para a criação de serviços web que seguem certos princípios.
- **Nível de Abstração:**
    - CRUD é mais focado nas operações diretamente relacionadas à persistência de dados.
    - REST está preocupado com a arquitetura geral do sistema, abordando questões de design para serviços web distribuídos.
- **Utilização de Métodos HTTP:**
    - CRUD não especifica necessariamente o uso específico de métodos HTTP.
    - REST baseia-se no uso dos métodos HTTP padrão para operações específicas (GET para leitura, POST para criação, PUT para atualização, DELETE para exclusão).

Em muitos casos, ao criar uma API RESTful, você estará implementando operações CRUD sobre recursos identificados por URIs específicas. Portanto, é comum ver esses conceitos sendo utilizados juntos em sistemas que seguem a abordagem RESTful.

RESTFUL vs REST: Qual a diferença?;

RESTful e REST são termos frequentemente usados de forma intercambiável, mas vale a pena destacar que REST é uma arquitetura e RESTful refere-se à implementação ou conformidade a princípios REST.

REST (Representational State Transfer):

- **Definição:**
    - REST é um estilo arquitetural que define um conjunto de princípios para a criação de serviços web escaláveis, interoperáveis e stateless (sem estado).
    - Foi proposto por Roy Fielding em sua tese de doutorado em 2000.
- **Princípios do REST:**

- Utilização de recursos identificados por URIs (Uniform Resource Identifiers).
- Representação dos recursos (JSON, XML, etc.).
- Uso dos métodos HTTP padrão (GET, POST, PUT, DELETE).
- Stateless: Cada requisição do cliente contém toda a informação necessária.

RESTful:

- **Definição:**
  - RESTful refere-se à implementação prática ou à conformidade aos princípios do REST em um sistema ou serviço web.
  - Um serviço ou API que segue os princípios do REST é considerado RESTful.
- **Características de um Serviço RESTful:**
  - Utilização de URIs para identificação de recursos.
  - Utilização adequada dos métodos HTTP para realizar operações em recursos (GET para leitura, POST para criação, PUT para atualização, DELETE para exclusão).
  - Stateless: A sessão do cliente é mantida no cliente, não no servidor.
  - Representação dos recursos em formatos como JSON ou XML.

Diferenças:

- **REST é um Estilo Arquitetural:**
  - REST é um conjunto de princípios arquiteturais que guiam o design de sistemas distribuídos.
  - Envolve conceitos fundamentais, como recursos, URIs, representações e métodos HTTP.
- **RESTful é uma Implementação:**
  - RESTful refere-se à aplicação prática dos princípios do REST em um serviço web específico.
  - Um serviço RESTful implementa os princípios do REST de maneira concreta.
- **Níveis de Conformidade:**

- Uma implementação pode ser considerada mais ou menos RESTful, dependendo de quão bem ela adere aos princípios do REST.
- Um serviço pode ser descrito como mais ou menos RESTful com base em quão bem ele segue as práticas recomendadas.

Ambos os termos são frequentemente usados para descrever sistemas e serviços web que adotam os princípios do REST. No geral, quando alguém se refere a um serviço como "RESTful", ela está indicando que o serviço segue os princípios e práticas do REST na implementação de suas operações e interações.

Exemplo REST express.js

Implementar o padrão REST no Express.js envolve o uso dos métodos HTTP corretos, a definição de rotas e a manipulação de recursos de forma a seguir os princípios RESTful. Abaixo, vou fornecer um exemplo básico de como você pode criar uma API RESTful simples usando o Express.js.

1. **Instalação de Dependências:**
   Certifique-se de ter o Node.js instalado. Crie um novo diretório para o projeto e, dentro dele, execute:

   ```
   bashCopy code
   npm init -y
   npm install express
   ```

2. **Criação do Arquivo de Aplicação (app.js):**
   Crie um arquivo chamado
   `app.js` para sua aplicação Express.

   ```
   javascriptCopy code
   const express = require('express');
   const app = express();
   const PORT = process.env.PORT || 3000;

   // Middleware para facilitar a leitura do corpo das requis
   ições (parsing JSON)
   ```

```
app.use(express.json());

// Simulação de um banco de dados em memória (array)
const books = [
  { id: 1, title: 'The Great Gatsby', author: 'F. Scott Fi
tzgerald' },
  { id: 2, title: 'To Kill a Mockingbird', author: 'Harper
Lee' },
];

// Rota para obter todos os livros
app.get('/books', (req, res) => {
  res.json(books);
});

// Rota para obter um livro específico por ID
app.get('/books/:id', (req, res) => {
  const bookId = parseInt(req.params.id);
  const book = books.find(book => book.id === bookId);

  if (!book) {
    return res.status(404).json({ error: 'Book not found'
});
  }

  res.json(book);
});

// Rota para criar um novo livro
app.post('/books', (req, res) => {
  const newBook = req.body;
  books.push(newBook);

  res.status(201).json(newBook);
});
```

```
// Rota para atualizar um livro existente por ID
app.put('/books/:id', (req, res) => {
  const bookId = parseInt(req.params.id);
  const updatedBook = req.body;

  const index = books.findIndex(book => book.id === bookI
d);

  if (index === -1) {
    return res.status(404).json({ error: 'Book not found'
});
  }

  books[index] = { ...books[index], ...updatedBook };

  res.json(books[index]);
});

// Rota para excluir um livro por ID
app.delete('/books/:id', (req, res) => {
  const bookId = parseInt(req.params.id);
  const index = books.findIndex(book => book.id === bookI
d);

  if (index === -1) {
    return res.status(404).json({ error: 'Book not found'
});
  }

  const deletedBook = books.splice(index, 1);

  res.json(deletedBook[0]);
});

// Inicia o servidor
app.listen(PORT, () => {
```

```
  console.log(`Server is running on http://localhost:${POR
T}`);
});
```

3. **Execução da Aplicação:**
   Execute
   `node app.js` para iniciar o servidor. Acesse `http://localhost:3000` e teste as rotas definidas.

Prático:

- Implementar API Padrão RESTFUL