API RESTful Node.js + PostgreSQL

Guilherme Gomes
guilherme.gomes@academico.ifpb.edu.br

Danilo Marques
danilo.marques@academico.ifpb.edu.br

Objetivo

- Desenvolver um aplicação capaz de atender as requisições HTTP e realizar o CRUD no banco PostgreSQL

API

- **API** (*Application Programming Interface)*, trata-se de um conjunto de rotinas e padrões estabelecidos e documentados por uma aplicação A, para que outras aplicações consigam utilizar as funcionalidades desta aplicação A, sem precisar conhecer detalhes da implementação do software.
- Desta forma, entendemos que as APIs permitem a comunicação entre aplicações e entre os usuários.

REST x RESTful

- **REST** significa *Representational State Transfer*.
- Trata-se de uma abstração da **arquitetura da Web**. Resumidamente, o REST consiste em **princípios e/ou regras** que permitem a criação de um projeto com interfaces bem definidas.
- **RESTful** é como se chama o sistema que aplica os os princípios de REST.

HTTP

- ***Hypertext Transfer Protocol (HTTP)*** é um protocolo de camada de aplicação para transmissão de documentos hipermídia, como o HTML.
- Foi desenvolvido para comunicação entre navegadores web e servidores web.
- Segue um modelo cliente-servidor clássico, onde um cliente abre uma conexão, executa uma requisição e espera até receber uma resposta.

HTTP Request Methods

- O protocolo HTTP define um conjunto de **métodos de requisição** responsáveis por indicar a ação a ser executada para um dado recurso.
- Eles são conhecidos como ***Verbos HTTP***.
- Vamos trabalhar com o **GET**, **POST**, **PUT** e **DELETE**.

GET

- O método GET solicita a representação de um recurso específico, se bem sucedido devem retornar dados.
- Código de Status:
  - 200 OK
  - 400 BAD REQUEST

POST

- O método POST é utilizado para submeter uma entidade a um recurso específico.
- Código de Status:
  - 201 CREATED
  - 400 BAD REQUEST

PUT

- O método PUT atualizado todos os campos do recurso de destino.
- Código de Status:
  - 200 OK
  - 404 NOT FOUND

DELETE

- O método DELETE remove um recurso.
- Código de Status:
  - 200 OK
  - 404 NOT FOUND

- O PostgreSQL é um poderoso sistema de banco de dados relacional de código aberto, com mais de 30 anos de desenvolvimento ativo, que ganhou uma forte reputação de confiabilidade, robustez de recursos e desempenho.
- Versão atual 12.1
- Controle Transacional (ACID), triggers e visões.

CRUD

- CRUD (**Create, Read, Update and Delete**) são as quatro operações básicas utilizadas em bases de dados relacionais fornecidas aos utilizadores do sistema.
- PADRÃO SQL
  - Create - INSERT
  - Read - SELECT
  - Update - UPDATE
  - Delete - DELETE

- O Node.js pode ser definido como um ambiente de execução Javascript *server-side*.
- Isso significa que com o Node.js é possível criar aplicações Javascript para rodar como uma aplicação *standalone* em uma máquina, não dependendo de um browser para a execução, como estamos acostumados.

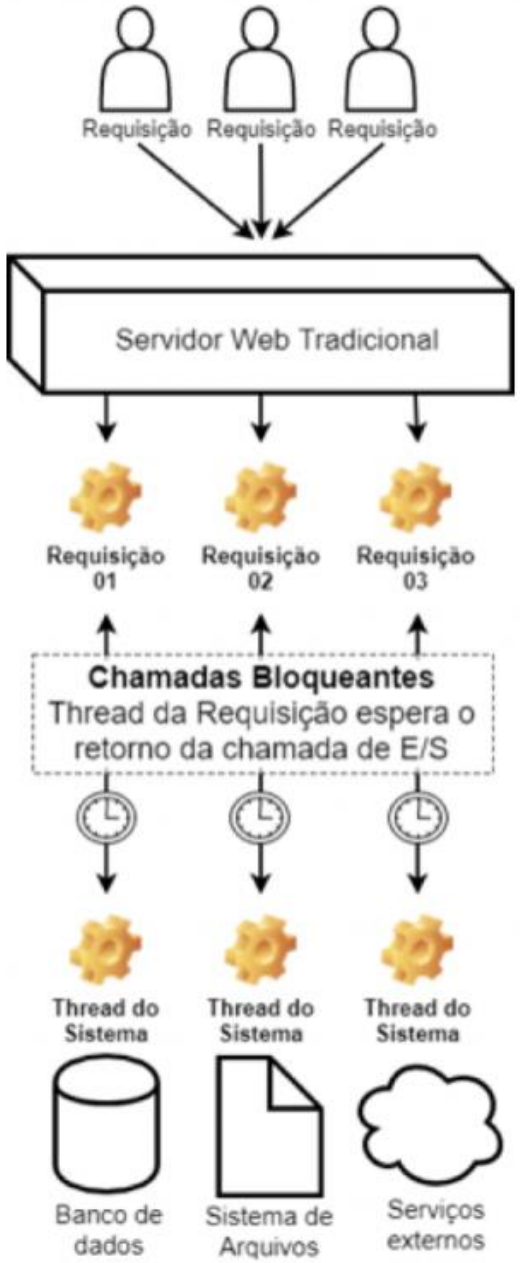
- Apesar de recente, o Node.js já é utilizado por grandes empresas no mercado de tecnologia, como Netflix, Uber e LinkedIn.
- O principal motivo de sua adoção é a sua alta capacidade de escala. Além disso, sua arquitetura, flexibilidade e baixo custo, o tornam uma boa escolha para implementação de Microsserviços e componentes da arquitetura *Serverless*.

- A principal característica que diferencia o Node.JS de outras tecnologias, como PHP, Java, C#, é o fato de sua execução ser *single-thread*.
- Apenas uma thread é responsável por executar o código Javascript da aplicação, enquanto que nas outras linguagens a execução é *multi-thread*.

- A cada requisição, serão demandados recursos computacionais (memória RAM, por exemplo) para a criação dessa nova *thread*.
- Uma vez que esses recursos são limitados, as *threads* não serão criadas infinitamente, e quando esse limite for atingido, as novas requisições terão que esperar a liberação desses recursos alocados para serem tratadas.

- No modelo Node.js, apenas uma thread é responsável por tratar as requisições. Essa thread é chamada de *Event Loop*, e leva esse nome pois cada requisição é tratada como um evento.
- O *Event Loop* fica em execução esperando novos eventos para tratar, e para cada requisição, um novo evento é criado.

Modelo **Node.js**	Modelo **Tradicional**



- No servidor Node.js, o *Event Loop* é a única *thread* que trata as requisições, enquanto que no modelo tradicional uma nova *thread* é criada para cada requisição.
- Enquanto o *Event Loop* delega uma operação de E/S para uma *thread* do sistema de forma assíncrona e continua tratando as outras requisições que aparecerem em sua pilha de eventos.
- As *threads* do modelo tradicional esperam a conclusão das operações de E/S, consumindo recursos computacionais durante todo esse período de espera.

- Apesar do Node.js ser *single-thread*, sua arquitetura possibilita um número maior de requisições concorrentes sejam tratadas em comparação com o modelo tradicional.

Vantagens

- Flexibilidade - O NPM (Node Package Manager) é o gerenciador de pacotes do Node.js e também é o maior repositório de softwares do mundo. Isso faz do Node.js uma plataforma com potencial para ser utilizada em qualquer situação.
- Leveza - Criar um ambiente Node.js e subir uma aplicação é uma tarefa que não exige muitos recursos computacionais em comparação com outras tecnologias mais tradicionais.
- Produtividade – NPM e mesma linguagem frontend e backend

Usos mais comum

- Aplicações de Tempo Real
- Ambientes Escaláveis
- Camada de Entrada do Servidor
- Mocks e Protótipos
- API com noSQL

Express.js

- O Express é um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel.

Referências

- Postgres Documentação: https://www.postgresql.org/docs/
- Node.js Documentação: https://nodejs.org/docs/latest-v10.x/api/
- Getting Started Express: http://expressjs.com/en/starter/installing.html
- MDN Web Docs: HTTP: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview