Aplicação RESTful com Node.js

Desenvolver uma aplicação RESTful com Node.js que controla as operações CRUD em um banco de dados MySQL/MariaDB.

CONFIGURAÇÃO DO AMBIENTE DE DESENVOLVIMENTO

Recursos Mínimos

- NODEJS - VERSÃO 12.14.1LTS: https://nodejs.org/en/
- Visual Studio Code - Versão 1.4.1 ou superior: https://code.visualstudio.com/download
- CMDER - Emulador de linha de comando: https://cmder.net/ (opcional)
- XAMPP versão 7.4.1: https://www.apachefriends.org/pt_br/index.html
- GIT BASH versão 2.25.0: https://gitforwindows.org/

Iniciando um servidor com Node.js

- Em local de sua preferencia cria a pasta *node-spa;*
- Utilizando VSCode crie novo arquivo com nome de index.js;
- Acesse o site https://nodejs.org/en/, clique em About
- Utilizaremos o código de exemplo para nossa primeira aplicação:

```
const http = require('http');


const hostname = '127.0.0.1';
const port = 3000;


const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});


server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

- Salve o arquivo;
- Agora no terminal do próprio VSCode digite:
  - node index.js [pressione enter]
- Isso executará o script e ficara acessível no navegadormno endereço:
  - http://localhost:3000/

Nodemon – Monitorando alterações no servidor

Módulo para manter o servidor funcionando: https://github.com/remy/nodemon

```
npm install -g nodemon
```

E o nodemon será instalado globalmente no caminho do sistema.

A partir de agora iniciaremos nosso projeto com comando:

```
nodemon [your node app]
```

Framework para rotas REST

https://github.com/restify/node-restify

O restify é uma estrutura, utilizando o middleware de estilo connect para criar APIs REST.
Antes de instalar o Restify vamos habilitar a gestão de pacotes de dependência em nosso projeto, criando o arquivo package.json.

```
npm init
```

Agora instalando Restify

```
npm install restify --save
```

Plugin para definir mensagens de erro: https://github.com/restify/errors

```
npm install restify-errors --save
```

Pronto agora podemos manipular as rotas de nossa aplicação.
Vamos para prática:

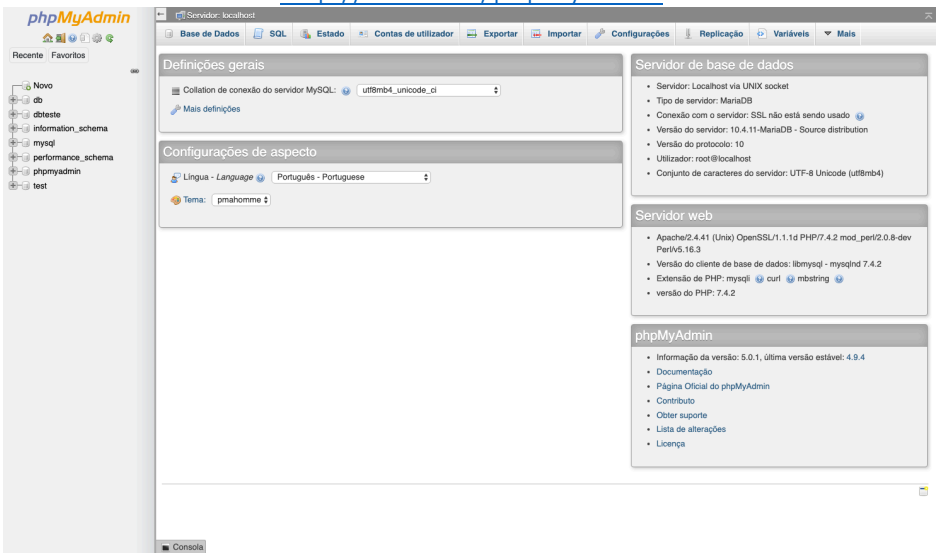
```
var restify = require('restify');

const server = restify.createServer({
  name: 'myapp',
  version: '1.0.0'
});

server.use(restify.plugins.acceptParser(server.acceptable));
server.use(restify.plugins.queryParser());
server.use(restify.plugins.bodyParser());

server.get('/echo/:name', function (req, res, next) {
  res.send(req.params);
  return next();
});

server.listen(8080, function () {
  console.log('%s listening at %s', server.name, server.url);
});
```

Depois da incluso do código inicie sua aplicação com o comando: nodemon index.js
Observe que a porta foi alterada para 8080, no navegador digite:

- http://localhost:8080/echo/SEUNOMEAQUI

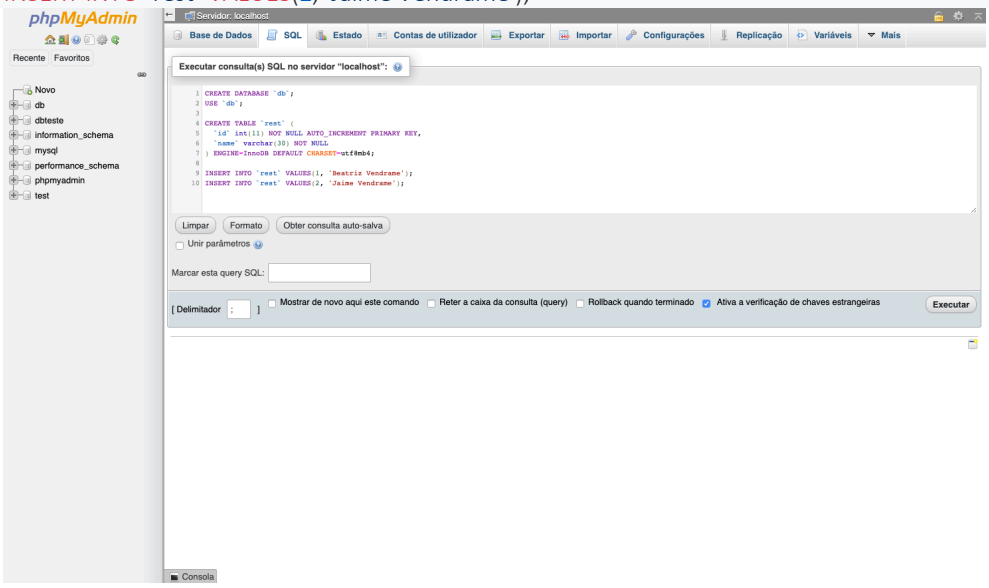
CONECTAR AO BANCO DE DADOS

Se tudo ocorreu bem com instalação do XAMPP, será possível acessar o gerenciador de banco através da url: http://localhost/phpmyadmin



Vamos criar um banco de dados e um tabela para realização de alguns testes.

```
CREATE DATABASE `db`;
USE `db`;

CREATE TABLE `rest` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO `rest` VALUES(1, 'Beatriz Vendrame');
INSERT INTO `rest` VALUES(2, 'Jaime Vendrame');
```



OK! Agora temos dados para serem manipulados.

Precisamos de um módulo ORM para MySQL/MariaDB.

Vamos usar o Knex um construtor de consultas para PostgreSQL, MySQL e SQLite3, projetado para ser flexível, portátil e divertido de usar.

Para isso precisamos instalar os pacotes referente ao Knex e o mysql.

```
npm install knex mysql --save
```

Incluindo o Código knex para conexão ao banco.

```
const restify = require('restify');

const server = restify.createServer({
  name: 'myapp',
  version: '1.0.0'
});

const knex = require('knex')({
  client: 'mysql',
  connection: {
    host : '127.0.0.1',
    user : 'root',
    password : '',
    database : 'db'
    }
});
server.use(restify.plugins.acceptParser(server.acceptable));
server.use(restify.plugins.queryParser());
server.use(restify.plugins.bodyParser());

server.get('/echo/:name', function (req, res, next) {
  res.send(req.params);
  return next();
});

server.listen(8081, function () {
  console.log('%s listening at %s', server.name, server.url);
});
```

FAZENDO CONSULTAS – TESTE COM POSTMAN

O que é o Postman?

O Postman é uma plataforma de colaboração para o desenvolvimento de API. Os recursos do Postman simplificam cada etapa da criação de uma API e agilizam a colaboração para que você possa criar APIs melhores - mais rapidamente.

Cliente de acesso ao servidor: https://www.getpostman.com/

Rota de Consulta

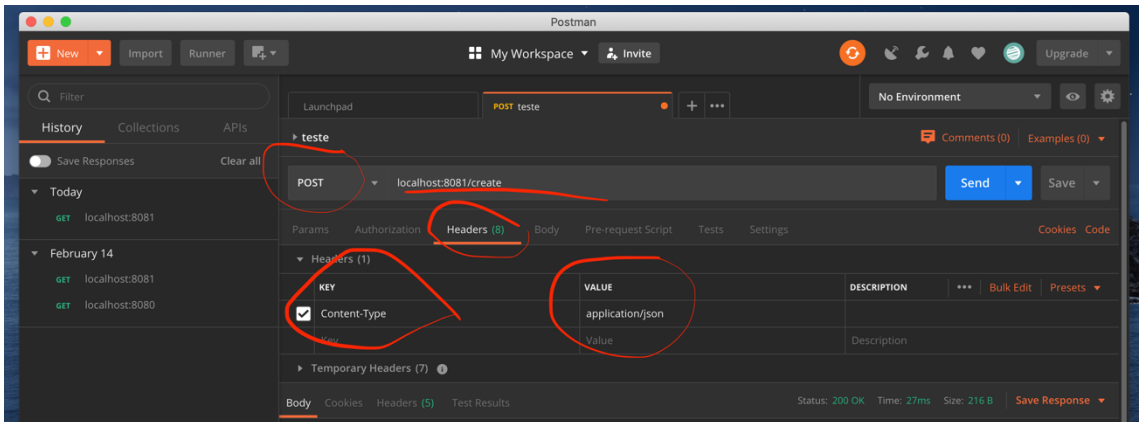
```
server.get('/', function (req, res, next) {

 knex('rest').then((dados)=>{
  res.send(dados)
 },next)

 return next();
});
```

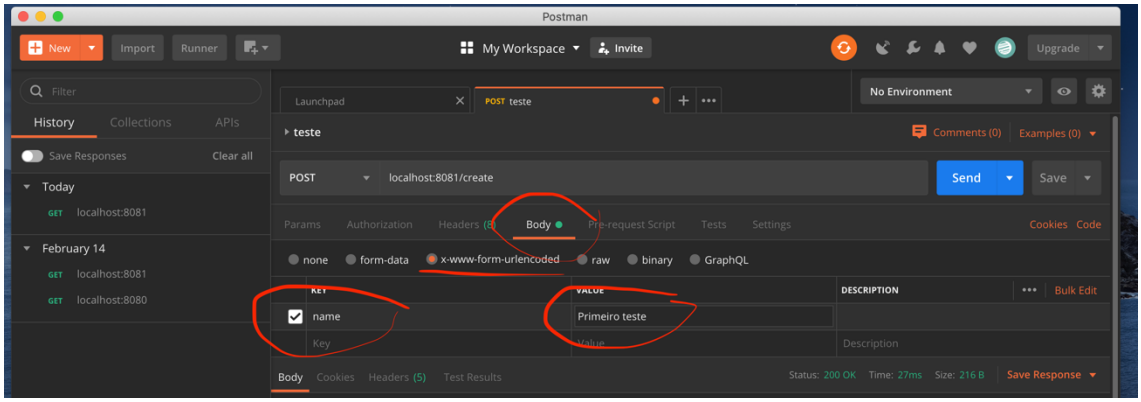
INSERINDO DADOS NA APLICAÇÃO

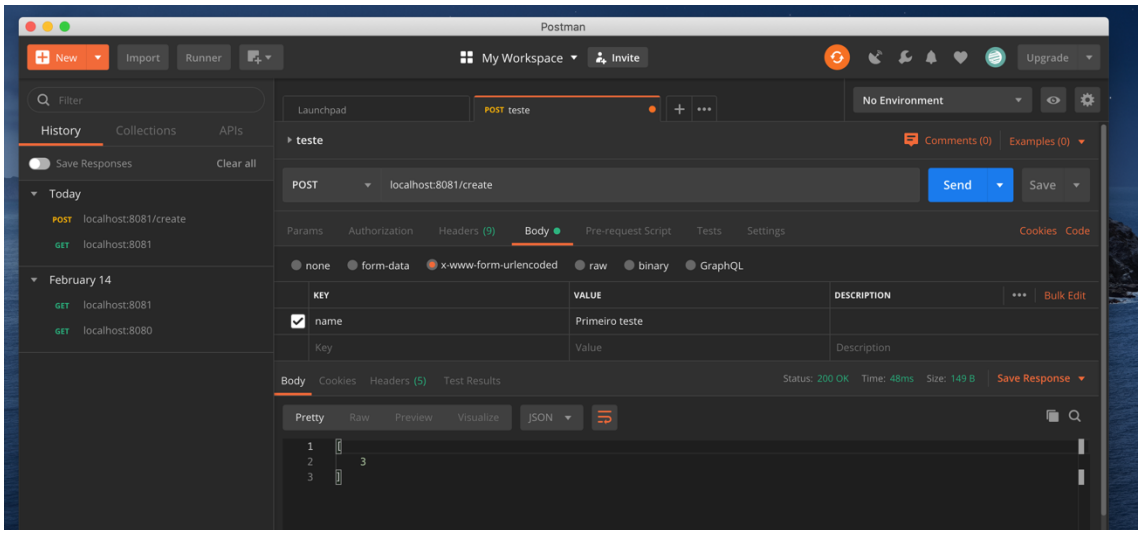
Criar a rota para o cadastro.

```
server.post('/create', function (req, res, next) {
 knex('rest')
  .insert(req.body)
  .then((dados)=>{
  res.send(dados)

 }, next)

 return next();
});
```

Configurando Postman







CONSULTA POR ID

Instalar o modulo de erros do Restify-errors.

```
const errs = require('restify-errors')
```

Criar rota de consulta por id

```
server.get('/show/:id', function (req, res, next) {

 const { id } = req.params

 knex('rest')
  .where('id', id)
  .first()
  .then((dados)=>{
   if(!dados) return res.send(new errs.BadRequestError('nada foi encontrado'))

    res.send(dados)

 }, next)

 return next();
});
```

ATUALIZANDO DADOS

```
server.put('/update/:id', function (req, res, next) {

  const { id } = req.params

  knex('rest')
    .where('id', id)
    .update(req.body)
    .then((dados)=>{
      if(!dados) return res.send(new errs.BadRequestError('nada foi encontrado'))

      res.send('dados atualizados!')

  }, next)

  return next();
});
```

EXCLUINDO DADOS

```
server.del('/delete/:id', function (req, res, next) {

  const { id } = req.params

  knex('rest')
    .where('id', id)
    .delete()
    .then((dados)=>{
      if(!dados) return res.send(new errs.BadRequestError('nada foi encontrado'))

      res.send('dados excluidos!')

  }, next)

  return next();
});
```

INTERFACE

Single Page Application (SPA)

Criar a pasta *dist* para conter os códigos *html* estáticos de nossa aplicação.
Criar arquivo *index.html*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <p>Olá mundo!</p>
</body>
</html>
```

Criar rota padrão para retornar o index.html

```
server.get('/', restify.plugins.serveStatic({
 directory: './dist',
 file: 'index.html'
}))
```

https://www.bootstrapcdn.com/
https://www.bootstrapcdn.com/fontawesome

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
  <link href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet"
integrity="sha384-wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0sibVcOQVnN"
crossorigin="anonymous">

  <title>Document</title>
</head>
<body>
  <div class="container">
    <h1>Cadastro de Nomes</h1>
    <div class="row">
      <div class="col-6">
        <input type="text" id="texto" class="form-control">
        <button id="botao" class="btn btn-primary">
          Enviar
          <i class="fa fa-check" aria-hidden="true"></i>
```

```
            </button>
          </div>
        </div>

        <div class="row">
          <div class="">
            <ul class="list-group" id="lista"></ul>
          </div>
        </div>
      </div>
</body>
</html>
```

Melhorando!

```
<input type="text" id="texto" class="form-control"
                style="display:inline; vertical-align:middle; width:auto;">
```

```
      <hr>
      <h1 style="margin-top: 40px;">Lista de Nomes</h1>
```

Definindo as funções e as consultas no servidor

```
<script>

      window.onload = () => {
        read()
      }
      function read(){

      }

      function create(){

      }

      function edit(){

      }

      function update(){

      }

      function del(){

      }
  </script>
</body>
```

Chamadas Ajax com Axios

https://github.com/axios/axios

```
window.onload = () => {
        const lista = document.querySelector('#lista')
        read()
    }
```

```
function read(){
        lista.innerHTML = "teste"

        //chamada ajax na rota /read
        axios.get('/read')
          .then((response) =>{
            console.log(response);
          })
          .catch((error) =>{
            console.log(error);
          })


    }
```

```
//chamada ajax na rota /read
        axios.get('/read')
          .then((response) =>{
            // console.log(response);
            response.data.forEach(element => {
              lista.innerHTML += `<li>${element.name}</li>`
            })
          })
          .catch((error) =>{
            console.log(error);
          })
```

Melhorando!

```
function read(){
        lista.innerHTML = ""
        //chamada ajax na rota /read
        axios.get('/read')
          .then((response) =>{
            // console.log(response);
            response.data.forEach(element => {
              lista.innerHTML += `<li class="list-group-item">${element.name}</li>`
            })
          })
          .catch((error) =>{
            console.log(error);
          })
    }
```

Cadastrando dados com Axios

```
const botao = document.querySelector('#botao')
const texto = document.querySelector('#texto')

//criando evento de click
botao.addEventListener('click', create )
```

```
function create(){

    const name = texto.value

    axios.post('/create', {name})
        .then((response) =>{
            console.log(response);

        })
        .catch((error) =>{
            console.log(error);
        })
}
```

Refatorar

Criar template para a lista

```
function templateLi (id, name){
    return `
        <li class="list-group-item">${name}</li>
    `
}
```

Função read

```
function read(){
    lista.innerHTML = ""

    //chamada ajax na rota /read
    axios.get('/read')
        .then((response) =>{
            // console.log(response);
            response.data.forEach(element => {
                lista.innerHTML += templateLi(element.id, element.name)
            })
        })
        .catch((error) =>{
            console.log(error);
        })
}
```

Função create

```
function create(){

    const name = texto.value

    axios.post('/create', {name})
      .then((response) =>{
        console.log(response)
        lista.innerHTML += templateLi(response.data[0],name)

      })
      .catch((error) =>{
        console.log(error);
      })
  }
```

Exclusão de Dados

Refatorar a função templateLi

```
  function templateLi (id, name){
          return `
              <li class="list-group-item">${name}
                  <i class="btn btn-danger delete fa fa-trash"
                  style="float:right; margin-left: 5px;"
                  ></i>

              </li>
              `
      }
```

Criar evento que ouça o evento click no botão delete.
Obs. Criar único evento para todos botões delete.

```
      lista.addEventListener('click', del )
```

Usando a propriedade target do javascript rastrear onde ocorre click e dessa forma disparar a função.

```
      function del(element){
          console.log(element)
      }
```

```
function del(element){
          console.log(element)
          if (element.target.classList.contains('delete')){
              //id do elemento a excluir?
          }
      }
```

Refatorar a função templateLi
Adicionado propriedade dataset do javascript

```
function templateLi (id, name){
            return `
                <li class="list-group-item">${name}
                    <i class="btn btn-danger delete fa fa-trash"
                    style="float:right; margin-left: 5px;"
                    data-id="${id}"
                    ></i>
                </li>
                `
        }
```

Capturar o id do registro passado pelo dataset

```
function del(element){
            console.log(element)
            if (element.target.classList.contains('delete')){
                //id do elemento a excluir?
                const id = element.target.dataset.id
            }
        }
```

Remover registro usando a propriedade *path* do javascript para identificar o elemento pai.

```
function del(element){
            console.log(element)
            if (element.target.classList.contains('delete')){
                //id do elemento a excluir?
                const id = element.target.dataset.id

                axios.delete(`/delete/${id}`)
                .then((response) =>{
                    console.log(response)
                    if (response.status = 200){
                        lista.removeChild(element.path[1])
                    }
                })
                .catch((error) =>{
                    console.log(error);
                })
            }
        }
```

Editando os dados
Refatorar templateLI

```
       function templateLi (id, name){
            return `
                <li class="list-group-item">${name}
                    <i class="btn btn-danger delete fa fa-trash"
                    style="float:right; margin-left: 5px;"
                    data-id="${id}"
```

```
                    ></i>
                    <i class="btn btn-primary update fa fa-trash"
                    style="float:right;"
                    data-id="${id}"
                    ></i>
                </li>
                `
        }
```

Evento click no botão edit

```
            lista.addEventListener('click', edit )
```

Função edit

```
function edit(element){
            if (element.target.classList.contains('update')){
                const input = document.createElement('input')

                input.type = 'text'

                input.setAttribute('value','')

                const pai = element.target.parentElement;
                const id = element.target.dataset.id

                pai.innerHTML = ''
                pai.appendChild(input)
            }
        }
```

Atualizando os dados
Refatorar função edit

```
pai.innerHTML = ''
                pai.appendChild(input)
                input.addEventListener('keydown', update)
                input.focus()
            }
        }
```

Função update

```
function update(){
            axios.delete(`/update/${id}`, { name: input.value })
                .then((response) =>{

                })
                .catch((error) =>{
                    console.log(error);
                })
        }
```

Refatorar função edit

```
        input.addEventListener('keydown', update.bind(pai, id, input))
```

Refatorar update

```
function update(id, input){
            axios.delete(`/update/${id}`, { name: input.value })
                .then((response) =>{
                    if (response.status = 200){
                        this.innerHTML = templateLi(id, input.value)
                    }
                })
                .catch((error) =>{
                    console.log(error);
                })
        }
```

```
                        this.innerHTML = templateLi(id, input.value, false)
```

refatorar templateLi

```
function templateLi (id, name, element = true){
            return `
                ${element ? `<li class="list-group-item">` : ''}
                    ${name}
                    <i class="btn btn-danger delete fa fa-trash"
                    style="float:right; margin-left: 5px;"
                    data-id="${id}"
                    ></i>
                    <i class="btn btn-primary update fa fa-wrench"
                    style="float:right;"
                    data-id="${id}"
                    ></i>
                ${element ? `</li>` : ''}
                `
        }
```

Update quando teclar enter

```
function update(id, input){
            const x = event.key
            if (x == null || x != 'Enter') return

            axios.put(`/update/${id}`, { name: input.value })
                .then((response) =>{
                    if (response.status = 200){
                        this.innerHTML = templateLi(id, input.value, false)
                    }
                })
                .catch((error) =>{
                    console.log(error);
                })
        }
```