Desenvolver aplicações Backend para WEB

Prof. Renan Ponick

Senac

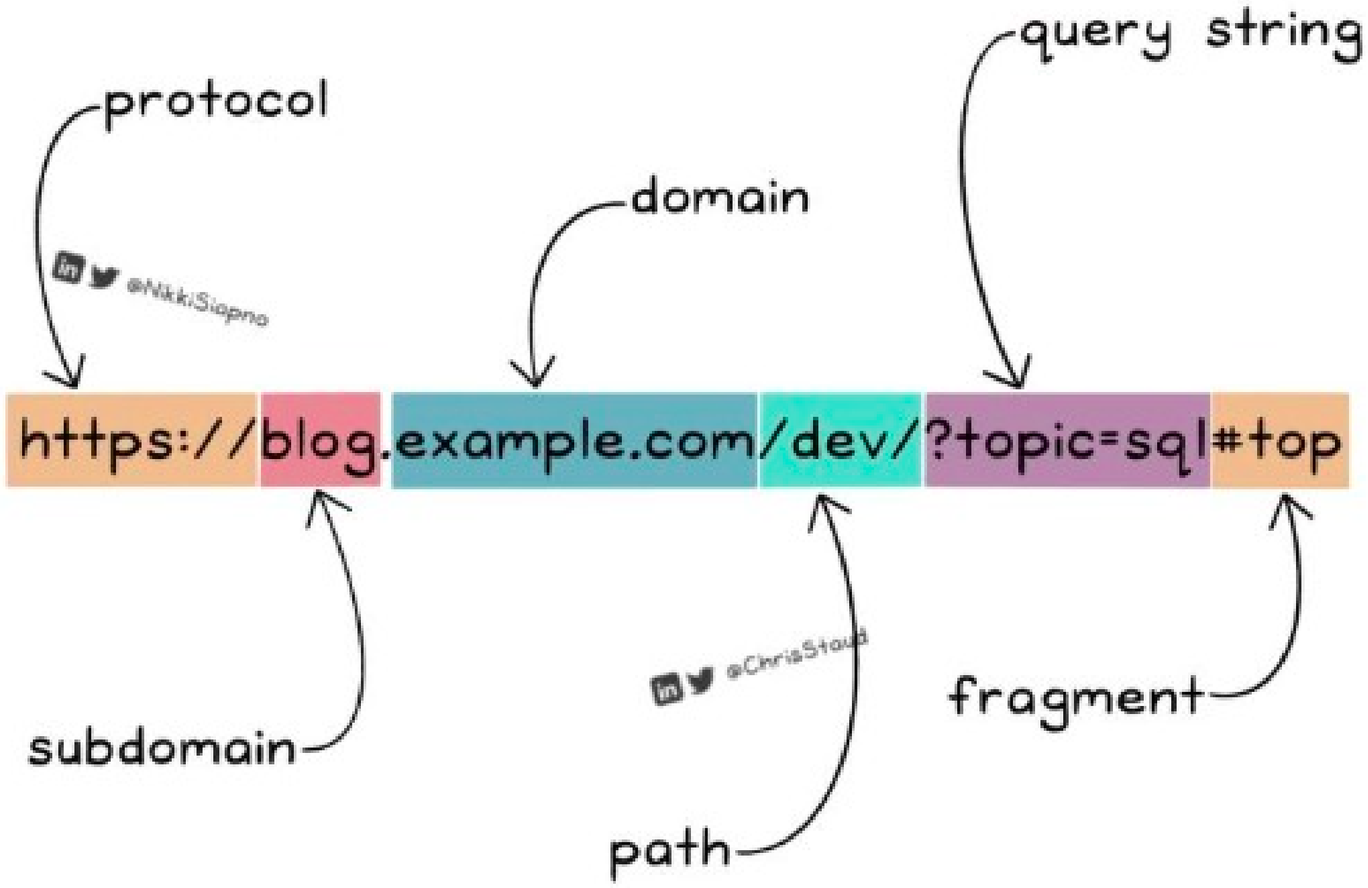
Lema

"Gastam enorme tempo nos detalhes e deixam passar coisas importantes. Guardam a atenção para as formigas e não enxergam os elefantes"

Baltasar Gracián

Components of a URL

by levelupcoding.co



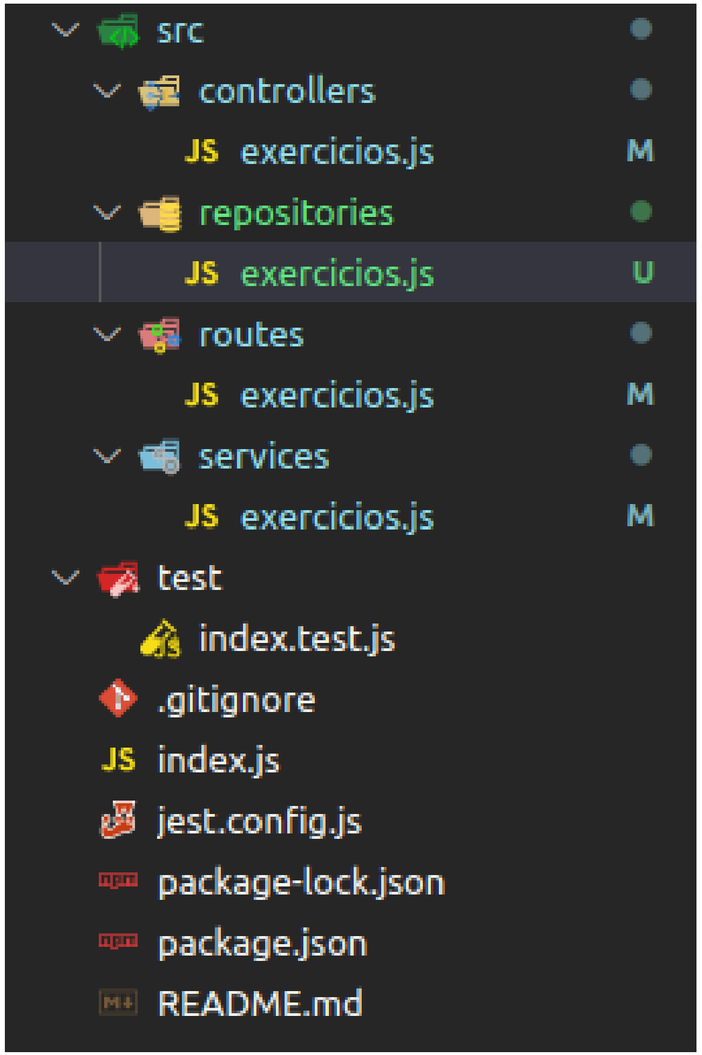
CRUD

Sem mais enrolações, da mesma forma que fizemos o C#, faremos em JS.

Faça o CRUD para listar nomes.

Estrutura

Adicionado o Repositores



Repositories

```
1  // const meuArray = new Array();
2  const nomes = new Array("Alice", "Bob", "Charlie");
3
4  export default class RepositorioExercicio {
5
6      PegarUm(index){
7          return nomes[index]
8      }
9
10     PegarTodos(){
11         return nomes
12     }
13
14     Adicionar(nome){
15         nomes.push(nome)
16     }
17
18     Alterar(index, nome){
19         nomes[index] = nome
20     }
21
22     Deletar(index){
23         nomes.splice(index, 1)
24     }
25
26 }
27
```

Service

```
import RepositorioExercicio from "../repositories/exercicios.js"

const repositorio = new RepositorioExercicio()
You, 1 second ago | 1 author (You)
export default class ServicoExercicio {

    PegarUm(index){
        return repositorio.PegarUm(index)
    }

    PegarTodos(){
        return repositorio.PegarTodos()
    }

    Adicionar(nome){
        if(!nome) {
          throw new Error("Favor preencher o nome.")
        }
        repositorio.Adicionar(nome)
    }

    Alterar(index, nome){
      if(!nome) {
        throw new Error("Favor preencher o nome.")
      } else if(!index || isNaN(index)) {
        throw new Error("Favor corretamente o index.")
      }

      repositorio.Adicionar(nome)
    }
```

Controllers

```
import ServicoExercicio from "../services/exercicios.js";

const servico = new ServicoExercicio()

You, 1 second ago | 1 author (You)
export default class ControllerExercicio {

    PegarUm(req, res){
      try {
        const index = req.params.index

        const result = servico.PegarUm(index)

        res.status(200).json({ nome: result});
      } catch (error) {
        res.status(500).json({ message: "Erro ao pegar um nome"});

      }
    }

    PegarTodos(req, res){
      try {
        const result = servico.PegarTodos()

        res.status(201).json({ nomes: result });
      } catch (error) {
        res.status(500).json({ message: "Erro ao listar nomes"});
      }
    }

    Adicionar(req, res){
      try {
        const nome = req.body.nome

        servico.Adicionar(nome)

        res.status(201).json({ message: "Adicionado com sucesso!"
      } catch (error) {
        res.status(500).json({ message: "Erro ao adicionar"});
      }
    }

    Alterar(req, res){
      try {
        const index = req.params.index
        const nome = req.body.nome

        servico.Alterar(index, nome)

        res.status(200).json({ message: "Alterado com sucesso!"})
      } catch (error) {
        res.status(500).json({ message: "Erro ao Alterar"});

      }
    }
```

Router

```
1   import express from "express";
2   import ControllerExercicio from "../controllers/exercicios.js";
3
4   const router = express.Router();
5
6   const controllers = new ControllerExercicio()
7
8   router.get("/api/nomes/", controllers.PegarTodos);
9   router.get("/api/nome/:index", controllers.PegarUm);
10  router.post("/api/nome", controllers.Adicionar);
11  router.put("/api/nome/:index", controllers.Alterar);
12  router.delete("/api/nome/:index", controllers.Deletar);
13
14  export default router;
15
```

Testes

```
1  import { describe, expect, it } from '@jest/globals';
2  import ServicoExercicio from "../src/services/exercicios";
3
4  describe('Testes do primeiro exercício', () => {
5
6      const servico = new ServicoExercicio()
7
8      // Executado antes de TODOS os testes
9      beforeAll(async () => {
10         console.info('Iniciando TDD com jest!');
11     });
12
13     // Executado após TODOS os testes
14     afterAll(() => {
15         console.info('Encerrados os testes');
16     });
17
18     it('Should add a name', () => {
19         const qtde = servico.PegarTodos().length
20         servico.Adicionar("Joao")
21         const qtdeAfter = servico.PegarTodos().length
22
23         expect(qtdeAfter).toBe(qtde + 1);
24     })
25
26     it('Should add a name and validate', () => {
27         const nome = "Lucas"
28         servico.Adicionar(nome)
29
30         const addedNome = servico.PegarUm(servico.PegarTodos().length - 1)
31         console.log(addedNome)
32         expect(nome).toBe(addedNome);
33     })
34
35 })
```

Exercicio

Realizem uma pesquisa para criar um objeto em JS

Depois da pesquisa faça o CRUD do Cachorro seguindo as mesmas regras de C#