



Curso Presencial Programação Fullstack

Aula 05

Prof. MSc. Kelson | Senior Software Engineer



PROFKELSON.DEV
BORA CODAR?

Cronograma

01

JSON Server

02

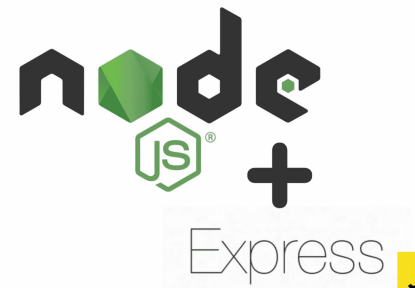
React Router

Back-end

- Estrutura que possibilita a operação do sistema
- Como as funções do meu sistema vão se comportar?
- Tudo que dá estrutura e apoio às ações do usuário
- Por trás daquela tela do site / app bonitinha... Quem é o responsável pela lógica interna?
- Nosso amigo back-end :)



Back-end



Front-end

- Interface em que o usuário pode interagir com o sistema
- HTML, CSS, JavaScript são as principais tecnologias envolvidas neste universo
- Facilitar a usabilidade do sistema
- Já imaginou o usuário ter que entender as requisições HTTPs para usar um sistema?





PROFKELSON.DEV

BORA CODAR?

Front e Back? Back e Front?



Auu!
Esse carinha
se chama
Bob...





PROFKELSON.DEV

BORA CODAR?

Front e Back? Back e Front?



Auu!
Bob decidiu
procurar no
Google cursos
de dev
presencial





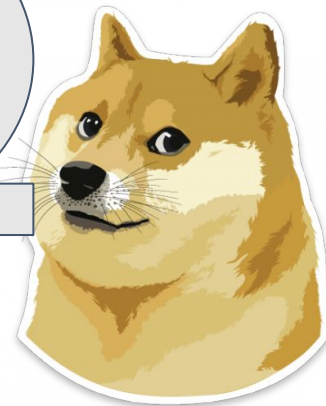
PROFKELSON.DEV

BORA CODAR?

Front e Back? Back e Front?



Auu!
Bob achou o
curso
presencial do
profkelson.
The Best 🥰



Exemplo de Comunicação Front e Back? Back e Front?



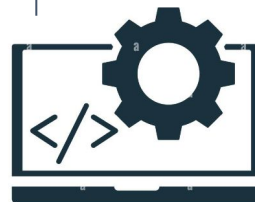
GET Request



GET Request



GET Request



PostgreSQL



O Servidor Web do profkelson recebe o *request* [GET] proveniente do *browser* do Bob.

Front end
O frontend ficará responsável por fornecer a página web (*interface*) com as informações do curso

BACK END
Mas quem busca as informações no banco de dados é o nosso amigo *back-end*

Atenção: Esta é apenas uma exemplificação, não necessariamente condiz com a realidade das tecnologias utilizadas pelo referido site.

Bob acessa <https://www.profkelson.dev>

Exemplo de Comunicação

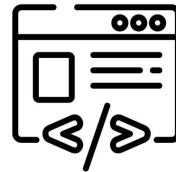
Bob acessa <https://www.profkelson.dev>



Response 200 OK



Response 200 OK



Front end

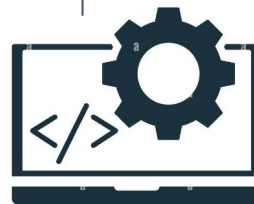
Response 200 OK



Body Response com o JSON com as infos

PostgreSQL

Java Spring



BACK END



Eu tenho esses dados!



Em 12 encontros, mergulhe em uma das áreas mais requisitadas. Desenvolva suas habilidades, receba orientação direta durante e

A era do "anywhere office" transformou a programação em uma carreira global. Empresas do Brasil e do mundo estão contratando profissionais de qualquer lugar. Não se restringe a um mercado local, de o profissional está na sua jornada no universo do desenvolvimento, abrindo portas para oportunidades nacionais e internacionais. Explore o potencial limitado da programação onde quer que você esteja.

```
json Copiar código
{
  "curso": {
    "nome": "Curso de Fullstack",
    "instrutor": "Prof. Kelson",
    "encontros": 12,
    "tecnologias": ["Java", "ReactJS"],
    "descricao": "Um curso abrangente que cobre desenvolvimento Fullstack usando Java para",
    "conteudo": [
      {
        "encontro": 1,
        "topico": "Introdução ao Desenvolvimento Fullstack"
      },
      {
        "encontro": 2,
        "topico": "Fundamentos de HTML, CSS e JavaScript"
      },
      {
        "encontro": 3,
```

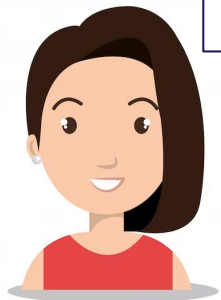


PROFKELSON.DEV

BORA CODAR?

HTTP

Eita! Vai abrir inscrições do curso do profkelson. Vou acessar o site dele....



Deva

GET REQUEST

<http://www.profkelson.dev>

Precisa de um PROTOCOLO de rede conseguir acessar o site.

Opa! Vou processar essa solicitação aí.





PROFKELSON.DEV

BORA CODAR?

HTTP

Show! Valeu



Em 12 encontros, mergulhe em uma das áreas mais requisitadas. Desenvolva suas habilidades, receba orientação direta durante e após o curso e transforme-se

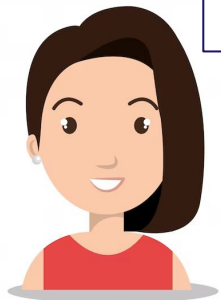
A sua vida "sempre off-line" transforme-se em uma carreira global. Engenheiro de Backend do mundo inteiro conectando profissionais de qualquer lugar. Não se limita a um mercado local, ele é capaz de atuar na sua própria rede mundial de desenvolvimento, através de uma conexão direta com os melhores profissionais internacionais. Explore o potencial ilimitado da programação onde quer que você esteja! 004

Garanta a sua vaga antes de mais! Último Início: Programação Fullstack em

RESPONSE 200 OK

SERVIDOR RESPONDEU QUE
ACHOU O CONTEÚDO
REQUISITADO.

Achei o site!
Vou te
responder!



Deva

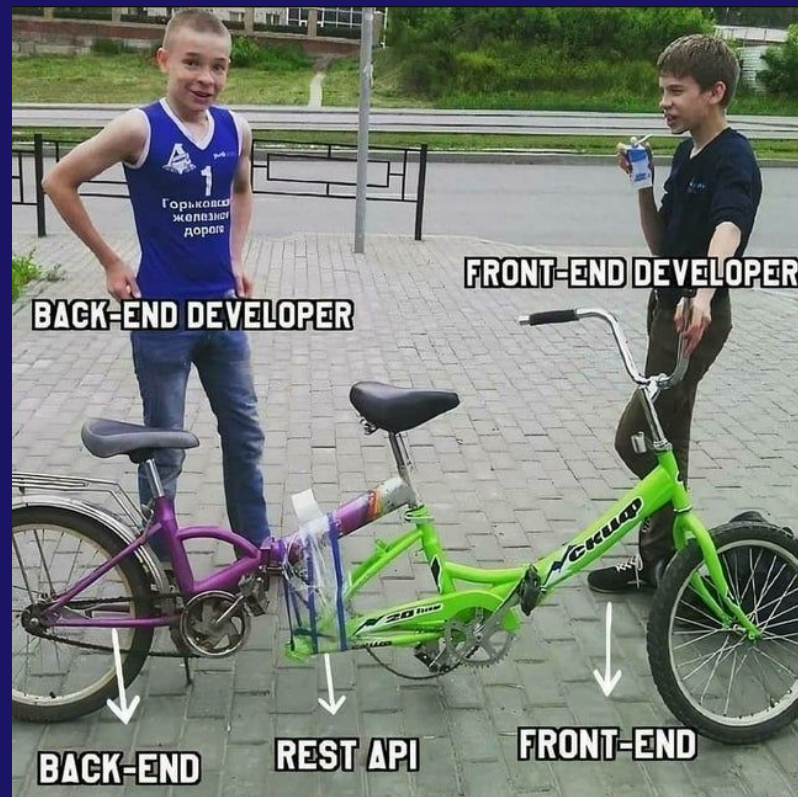


PROFKELSON.DEV

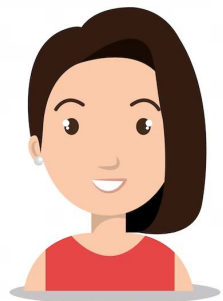
BORA CODAR?

O que é uma API?

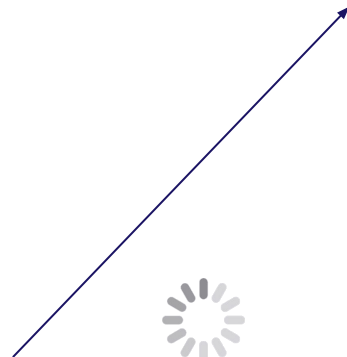
- **API** (Application Programming Interface): Conjunto de regras e definições que permitem que diferentes softwares se comuniquem entre si.
- **Como funciona:** Uma API age como um intermediário entre dois programas, permitindo que eles se comuniquem e troquem dados.
- Situação: "Meu back precisa se comunicar com o meu front!" - API nele!



HTTP



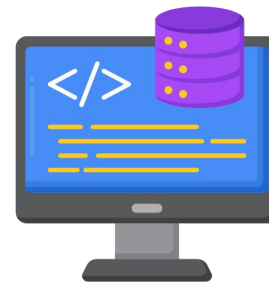
Deva



Fala brother,
backo! Blz?

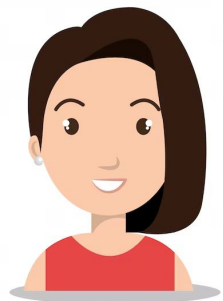


Frontend

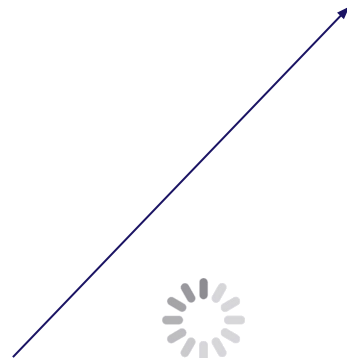


Backend

HTTP



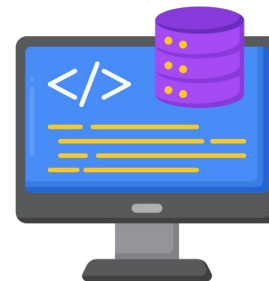
Deva



Deva quer ver as
informações do
curso do prof
kelson

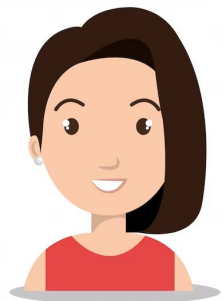


Frontend

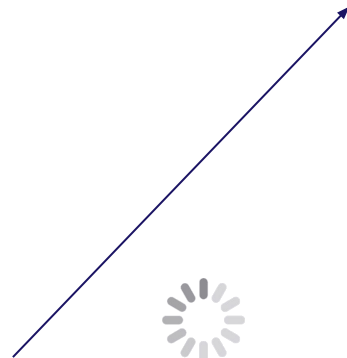


Backend

HTTP



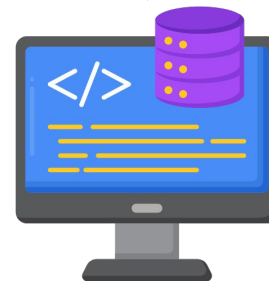
Deva



Processa aí a
regra de negócio
para isso, please.

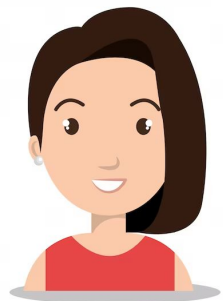


Frontend

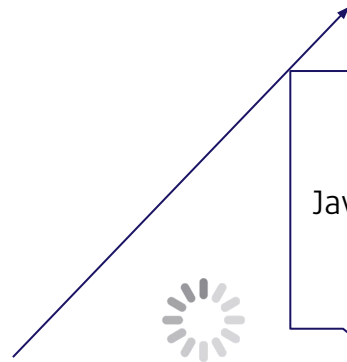


Backend

HTTP

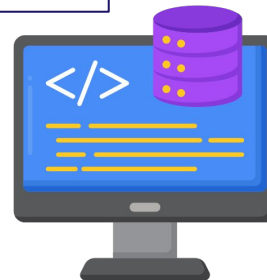


Deva



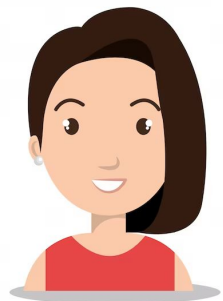
Frontend

Oxe, nem
JavaScript eu falo,
mano! 🤯

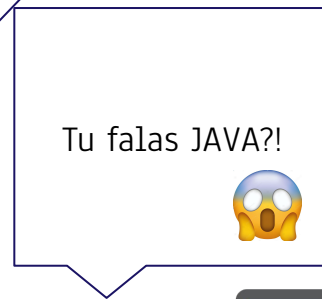
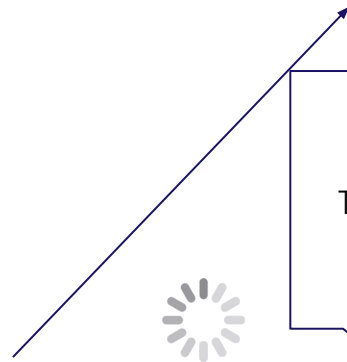


Backend

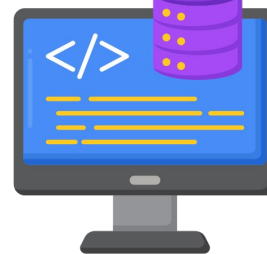
HTTP



Deva

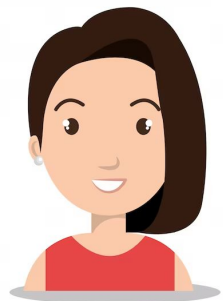


Frontend

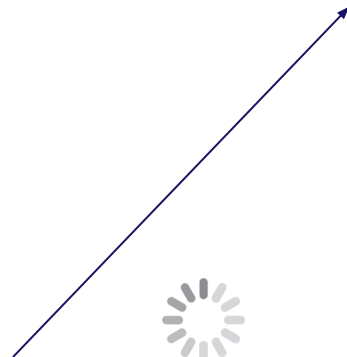


Backend

HTTP



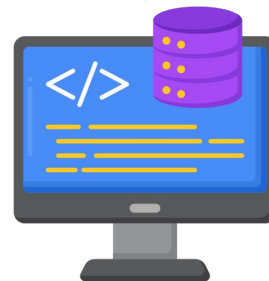
Deva



Falo p... nenhuma
de JAVA, e agora!?

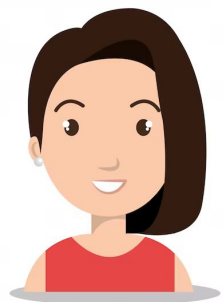


Frontend

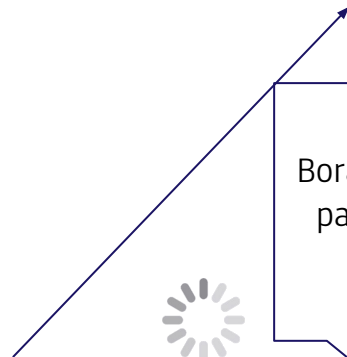


Backend

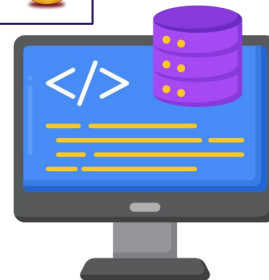
HTTP



Deva

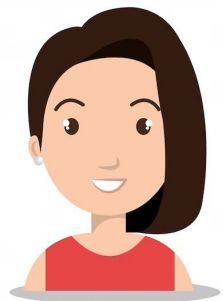


Frontend

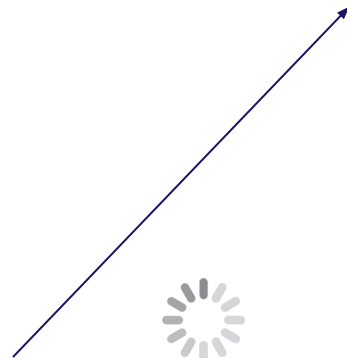


Backend

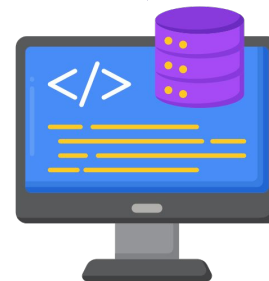
HTTP



Deva



Frontend



Backend



PROFKELSON.DEV

BORA CODAR?

O que é REST?

- REST (Representational State Transfer): É um estilo arquitetônico para criar APIs.
- Características de uma API RESTful:
 - Baseada em recursos: Tudo é considerado um recurso (ex: usuários, produtos).
 - Stateless: Cada solicitação do cliente ao servidor deve conter todas as informações necessárias para entender e processar o pedido.
 - Usa métodos HTTP: GET, POST, PUT, DELETE, etc.
 - Formato de dados: Normalmente utiliza **JSON** ou XML.

{ REST }

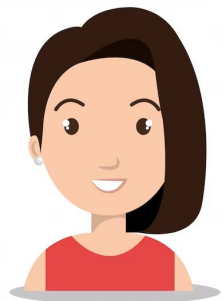
Estrutura de uma API RESTful

- Endpoint: URL que representa um recurso (ex: `http://api.meusite.com/usuarios`).
- Métodos HTTP: Definem a ação a ser realizada no recurso (ex: GET para obter dados, POST para criar novos dados).

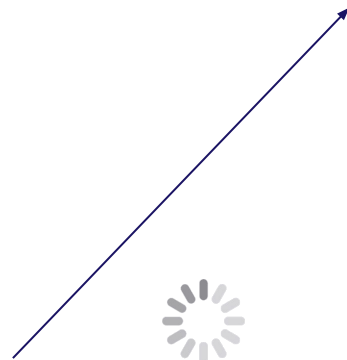


{ REST }

HTTP



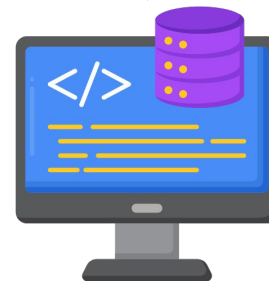
Deva



Show, API!
#partiu!
Mas, comofaz?

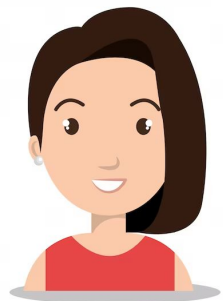


Frontend

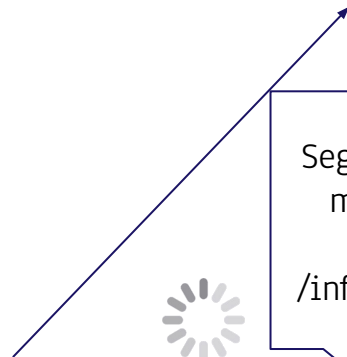


Backend

HTTP



Deva

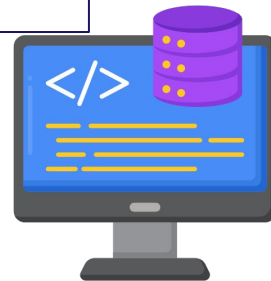


Show, API!
#partiu!
Mas, comofaz?



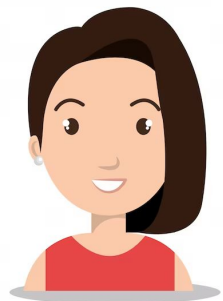
Frontend

Seguindo o REST,
manda aí pra
mim:
`/infos-curso [GET]`

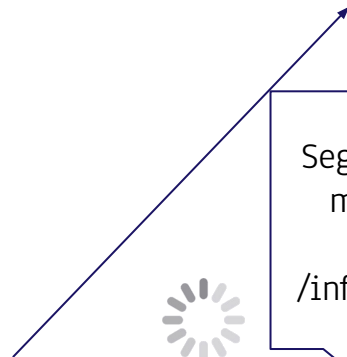


Backend

HTTP



Deva



Show, API!
#partiu!
Mas, comofaz?



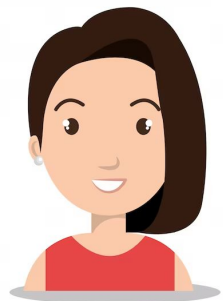
Frontend

Seguindo o REST,
manda aí pra
mim:
`/infos-curso [GET]`

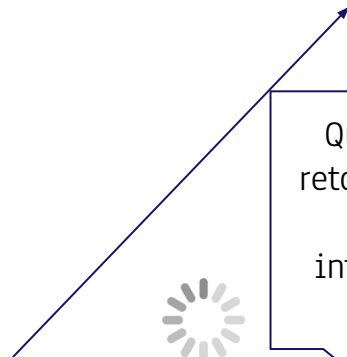


Backend

HTTP



Deva

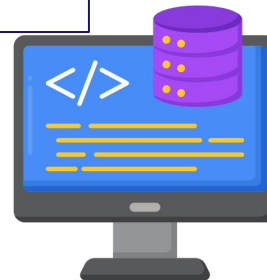


Show, API!
#partiu!
Mas, comofaz?

Que eu vou te
retornar um JSON
com as
informações do
curso.

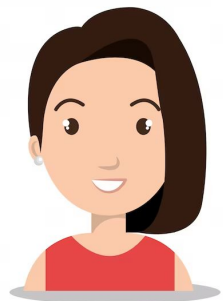


Frontend

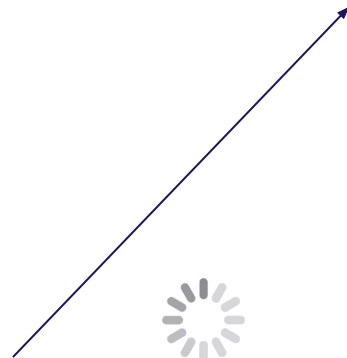


Backend

HTTP



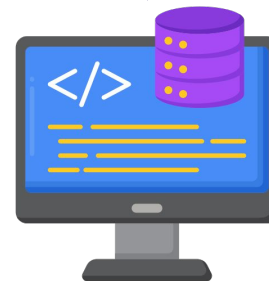
Deva



Demorô! JSON eu
entendo! Manda
aí, parça!



Frontend

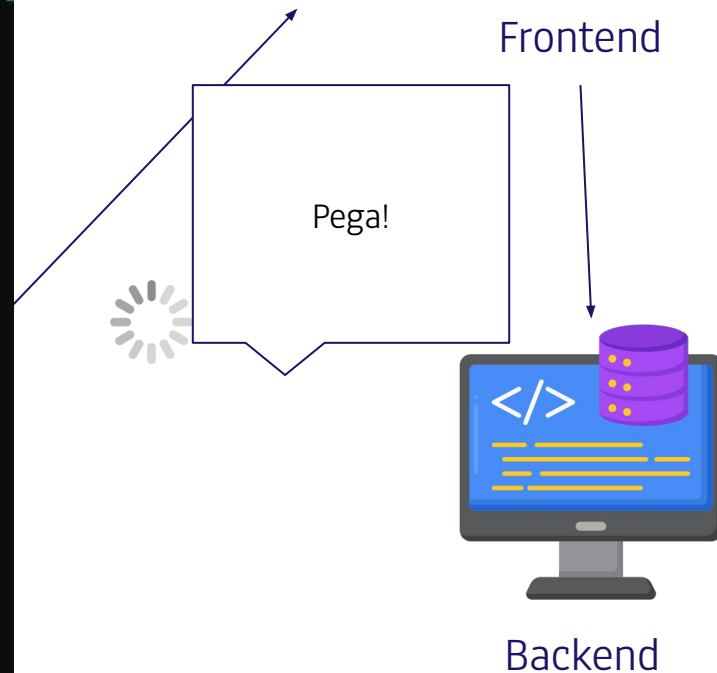


Backend

```
{
  "curso": {
    "nome": "Curso de Fullstack",
    "instrutor": "Prof. Kelson",
    "encontros": 12,
    "tecnologias": ["Java", "ReactJS"],
    "descricao": "Um curso abrangente que cobre desenvolvimento Fullstack usando Java para
    "conteudo": [
      {
        "encontro": 1,
        "topico": "Introdução ao Desenvolvimento Fullstack"
      },
      {
        "encontro": 2,
        "topico": "Fundamentos de HTML, CSS e JavaScript"
      },
      {
        "encontro": 3,
        "topico": "Introdução ao ReactJS"
      },
      {
        "encontro": 4,
        "topico": "Projeto Prático: Construindo uma Aplicação com ReactJS"
      }
    ]
  }
}
```



Frontend



Backend



PROFKELSON.DEV
BORA CODAR?

JSON Server

- JSON Server é uma biblioteca que permite criar rapidamente uma API RESTful fake (simulada) usando apenas um arquivo JSON.
- Muito útil para devs frontend que precisam testar e prototipar seus aplicativos antes de implementar o backend real.

```
\aula05> npm install json-server@0.16.3
```

```
{ } package.json X
http_com_react > { } package.json > { } devDependencies
  > Debug
6  "scripts": {
7    "dev": "vite",
8    "build": "vite build",
9    "lint": "eslint src --ext js,jsx --report-unused-c
10   "preview": "vite preview",
11   "server": "json-server --watch data/db.json"
12 },
```



PROFKELSON.DEV
BORA CODAR?

JSON Server

- Nos exemplos ao lado temos a instalação do pacote JSON Server através do comando:
 - `npm install json-server`
- É preciso que no arquivo "package.json" adicionemos a propriedade "server" em "scripts".
- Assim criaremos um "watch" do json-server para o arquivo data/db.json
- O servidor json-server é inicializado em outro terminal através do comando:
 - `npm run server`

```
\aula05> npm install json-server@0.16.3
```

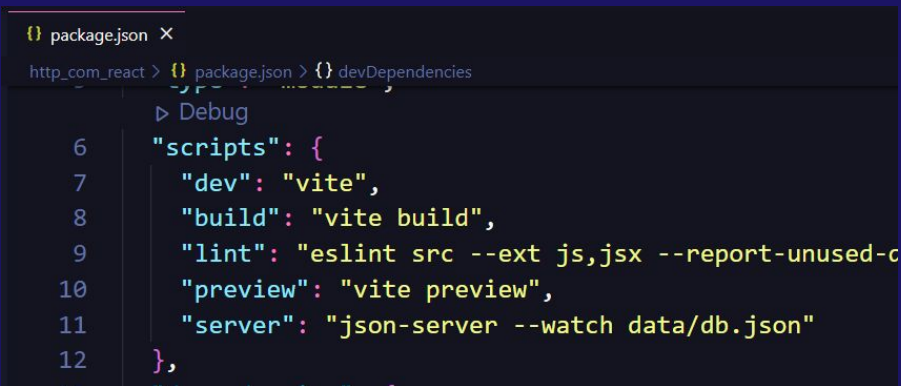
```
{} package.json X
http_com_react > {} package.json > {} devDependencies
  {} dependencies
  {} scripts
  {} server
  {} watch
  {} data/db.json
  {} data/db.json

  > Debug
6  "scripts": {
7    "dev": "vite",
8    "build": "vite build",
9    "lint": "eslint src --ext js,jsx --report-unused-c
10   "preview": "vite preview",
11   "server": "json-server --watch data/db.json"
12 },
```

JSON Server

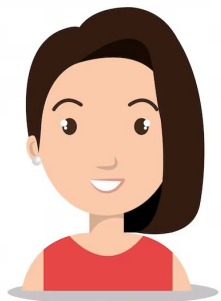
- No arquivo data/db.json, nós criamos um json que irá abstrair um possível banco de dados do qual queremos extrair os dados a serem retornados via API. (fake back-end)
- Nestes exemplos, o json-server será inicializado na porta 3000. E um endpoint `http://localhost:3000/products` ficará disponível para consulta.

```
\aula05> npm install json-server@0.16.3
```



```
{} package.json X
http_com_react > {} package.json > {} devDependencies
  > Debug
6  "scripts": {
7    "dev": "vite",
8    "build": "vite build",
9    "lint": "eslint src --ext js,jsx --report-unused-c
10   "preview": "vite preview",
11   "server": "json-server --watch data/db.json"
12 },
```



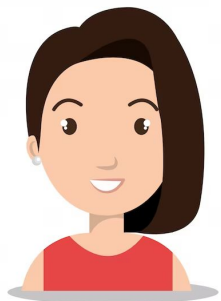



Dev
Front

Produtiva!
Já terminou
as telas!



Dev
Back



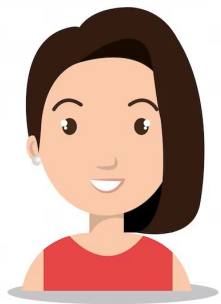
Dev
Front

Produtiva!
Já terminou
as telas!



Dev
Back

Tá moscando!
Atrasou!
Não terminou
a API

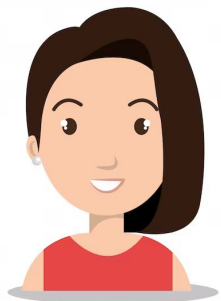


Dev
Front

Hoho! Poxa
vida!
Para não
ficar parada
vai fazer uma
API fake no
front



Dev
Back

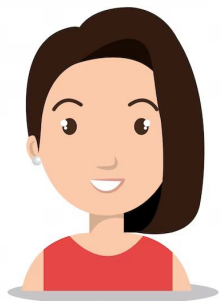


Dev
Front

Quando o
atrasão aí
terminar a
API
verdadeira,
só mudo a
URL da API
fake para a
verdadeira.



Dev
Back



Dev
Front



Dev
Back

Aí sim! #partiu!

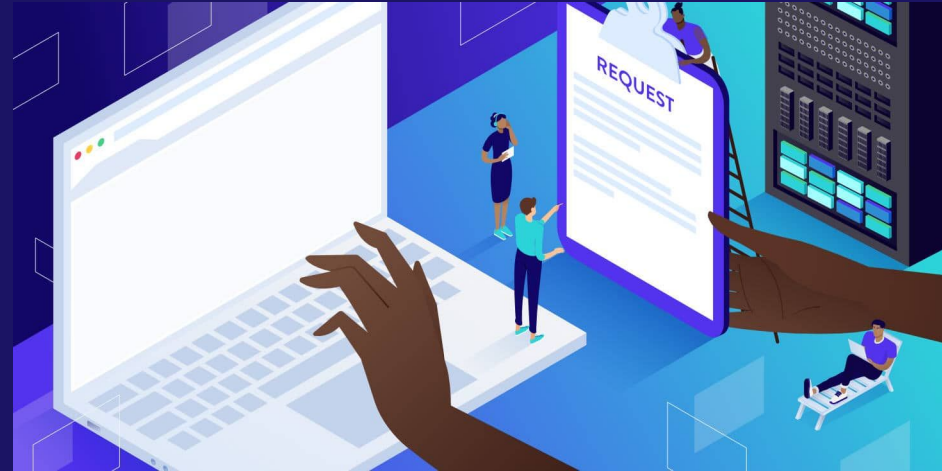
#prazoApertado
#mudançasDeEscopo
#burnOut



PROFKELSON.DEV
BORA CODAR?

HTTP

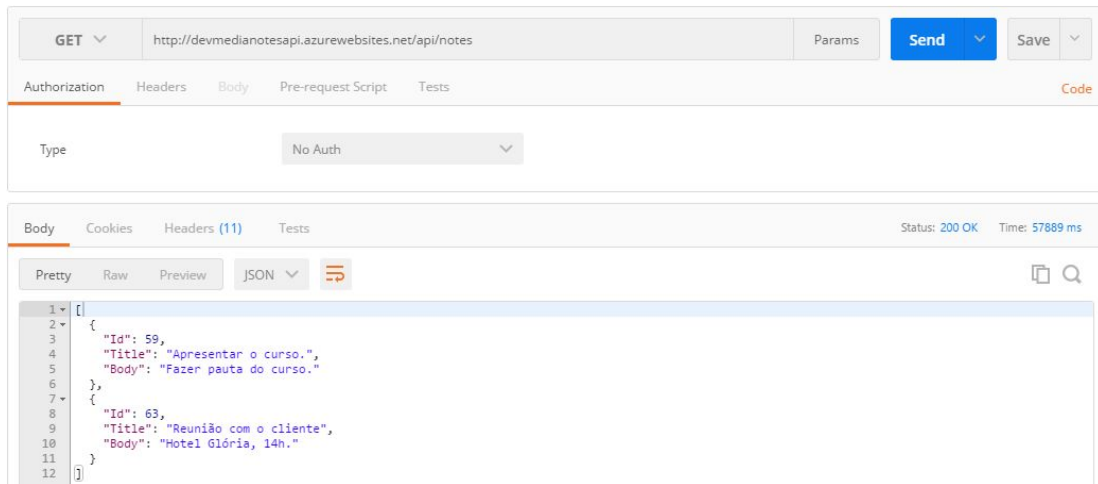
- HTTP (Hypertext Transfer Protocol): É o **protocolo** usado para comunicação na web. Permite a transferência de dados entre um cliente (como um navegador) e um servidor.
- **Como funciona:** Quando você digita um URL em seu navegador e pressiona Enter, uma solicitação HTTP é enviada ao servidor. O servidor processa essa solicitação e envia uma resposta de volta ao navegador.



Verbos HTTP

- **GET**

- Solicita a representação de um recurso específico.
- Requisições com GET retornam apenas dados.
- Ler dados, jamais alterar!



Get?



Bob
“solicitou”
as infos...



```
{
  "description": "Terminal JSON viewer",
  "websiteUrl": "https://fx.wtf",
  "isAwesome": true,
  "databaseId": 19872411,
  "repositories": {"totalCount": 55, ...},
  "sponsors": {"totalCount": 22, ...},
  "followers": {"totalCount": 1763, ...},
  "licenses": [{...}, ...]
}
```

.description

data.json

Verbos HTTP

- **POST**
- Criação de recursos
- Adiciona informações a um recurso
- Geralmente é utilizado para enviar dados de formulários

The screenshot shows a REST client interface with the following elements:

- Method:** POST (selected from a dropdown)
- URL:** http://devmedianotesapi.azurewebsites.net/api/notes
- Buttons:** Params, Send (blue), Save
- Tabs:** Authorization, Headers (1), Body (selected), Pre-request Script, Tests
- Body Type:** form-data, x-www-form-urlencoded, raw (selected), binary, JSON (application/json) (dropdown)
- Body Content:**

```
1 {  
2   "Title": "Pegar as crianças na escola.",  
3   "Body": "Centro, 10h30."  
4 }
```
- Code:** (link in the bottom right corner)

Post?



Bob preencheu
um formulário
no site e clicou
em salvar...

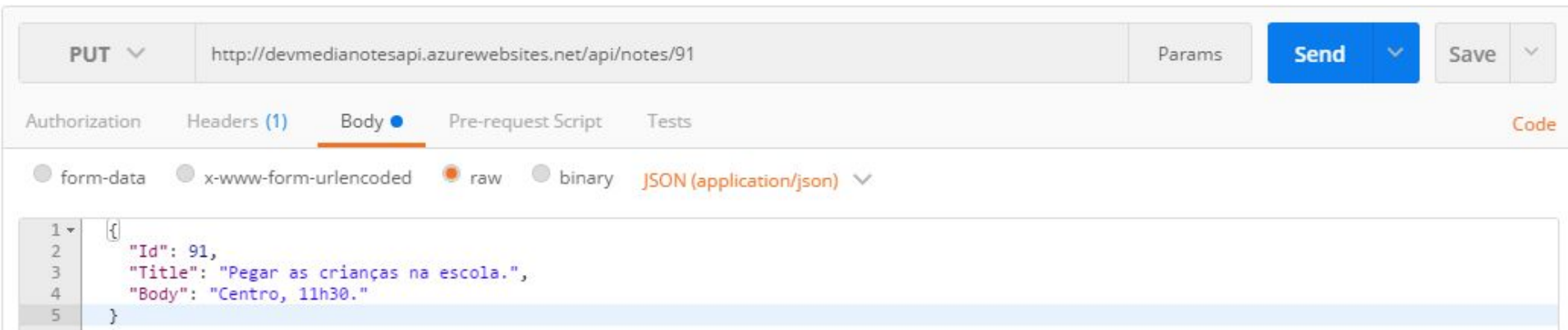


SIGN UP

CREATE ACCOUNT

Verbos HTTP

- **PUT**
- Atualiza informações de um recurso (update)



The screenshot shows an HTTP client interface with the following elements:

- Method:** PUT (selected from a dropdown)
- URL:** http://devmedianotesapi.azurewebsites.net/api/notes/91
- Params:** A button to manage query parameters.
- Send:** A blue button to execute the request.
- Save:** A button to save the request configuration.
- Tabs:** Authorization, Headers (1), Body (selected), Pre-request Script, Tests.
- Body Type:** radio buttons for form-data, x-www-form-urlencoded, raw (selected), and binary.
- Content Type:** JSON (application/json) (selected from a dropdown).
- Body Content:** A text area containing a JSON object:

```
{
  "Id": 91,
  "Title": "Pegar as crianças na escola.",
  "Body": "Centro, 11h30."
}
```

PUT?



**Bob preencheu
tudo errado e
precisa
atualizar todas
as
informações.**



IT Asset Request Form
Place requests for any IT assets required in the organization.

1 2 3 4 5

Employee Details

Name

First: Last:

Email

Select OS Type

☐ Windows

☐ MacOS

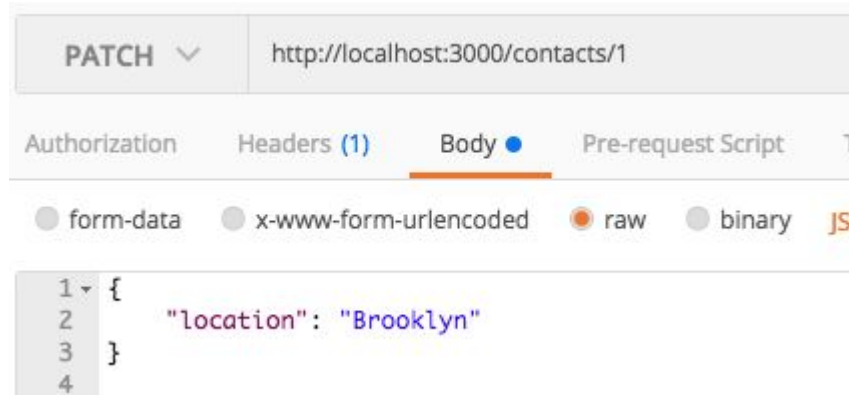
☐ Linux

☒

Next

Verbos HTTP

- **PATCH**
- Envia apenas o que precisa ser alterado, sem a necessidade de precisar enviar todos os dados.



Patch?



Bob preencheu apenas um campo errado, e precisou atualizar apenas aquela informação



IT Asset Request Form

Place requests for any IT assets required in the organization.

Employee Details

Name

First

Last

Email

daniel.hudson@zyklr.com

Select OS Type

☐ Windows

☐ MacOS

☐ Linux

Next

Verbos HTTP

- **DELETE**
- Remove um dado passado na URI

The screenshot shows a REST client interface with the following elements:

- Method:** DELETE (selected from a dropdown)
- URI:** http://devmedianotesapi.azurewebsites.net/api/notes/91
- Params:** A button to manage query parameters.
- Send:** A blue button to execute the request.
- Save:** A button to save the request configuration.
- Tabs:** Authorization, Headers (1), Body (selected), Pre-request Script, Tests.
- Body Type:** A row of radio buttons for form-data, x-www-form-urlencoded, raw (selected), and binary. A dropdown menu shows JSON (application/json).
- Body Editor:** A text area with a line number '1' on the left.
- Code:** A button in the top right corner.

Exemplo de Envio

Nome:

Email:

Telefone:

Enviar

Exemplo de Envio

Nome:

Email:

Telefone:

Atualizar

Exemplo de Envio

Nome:

Email:

Telefone:

Atualizar

Exemplo de Envio

Nome	Email	Telefone	Ação
Kelson Almeida 2	mentoria2@profkelson.dev	(83) 1111111111	

DELETE
Verb

Top 9 HTTP Request Methods

ByteByteGo

GET



GET /v1/products/iphone

Response

Retrieve a single item
or a list of items

PUT



PUT /v1/users/123

Request Body

Response

Update an item

POST



POST /v1/users

Request Body

Response

Create an item

DELETE



DELETE /v1/users/123

Response

Delete an item

PATCH



PATCH /v1/users/123

Request Body

Response

Partially modify an item

HEAD



HEAD /v1/products/iphone

Response

Identical to GET but no
message body in the response

CONNECT



CONNECT xxx.com:80

Request

Response

Create a two-way connection
with a proxy server

OPTIONS



OPTIONS /v1/users

Response

Return a list of supported
HTTP methods

TRACE



TRACE /index.html

Response

Perform a message loop-
back test, providing a
debugging mechanism

Códigos de Status HTTP

- A cada response, o protocolo HTTP nos retorna um código de “status” referente a cada tipo de retorno.
- Mas, prof. Esses códigos são aleatórios?
 - Não... São numerações pré-definidas e separadas por classes.



404. That's an error.

The requested URL /does_not_exist was not found on this server. That's all we know.



Códigos de Status HTTP

- Estão divididos em 5 classes:
 - **100s:** A solicitação iniciada pelo cliente HTTP continua...
 - **200s:** O pedido feito no *request* foi recebido, compreendido e processado pelo servidor.
 - **300s:** Redirecionamento. Um novo recurso foi substituído pelo recurso solicitado.
 - **400s:** Erros. Houve problema com o pedido.
 - **500s:** A solicitação foi aceita, mas aconteceu algum erro no servidor que impede o processamento completo da solicitação

HTTP STATUS CODES

2xx Success

200 Success / OK

3xx Redirection

301 Permanent Redirect

302 Temporary Redirect

304 Not Modified

4xx Client Error

401 Unauthorized Error

403 Forbidden

404 Not Found

405 Method Not Allowed

5xx Server Error

501 Not Implemented

502 Bad Gateway

503 Service Unavailable

504 Gateway Timeout

18 CÓDIGOS DE STATUS HTTP

(essenciais para desenvolvedores)

200
OK

Solicitação bem-sucedida



201
CREATED

Recurso criado



202
ACCEPTED

Pedido aceito



204
NO CONTENT

Sem conteúdo



301
MOVED PERM.

Nova URL permanente



302
FOUND

Nova URL temporária



304
NOT MODIFIED

Inalterado desde última req.



400
BAD REQUEST

Erro do lado do cliente



401
UNAUTHORIZED

Necessita autenticação



403
FORBIDDEN

Acesso proibido



404
NOT FOUND

Recurso não encontrado



405
NOT ALLOWED

Metódo não permitido



408
REQ TIMEOUT

Solicitação expirou



500
I.S. ERROR

Erro de servidor



501
NOT IMPLEMENTED

Não implementado



502
BAD GATEWAY

Gateway inválido



503
S UNAV.

Serviço indisponível



504
GATEWAY TIMEOUT

Tempo limite excedido





PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (1) [COM PROF]

- Através do seu projeto React crie uma RESTful api fake que retorne uma lista de Alunos, onde cada aluno terá nome, email e curso.
- Faça um get nessa lista de alunos, onde o mesmo retornará a lista desses alunos.
- Mostre o resultado em um cliente HTTP.
 - Através do insomnia:
 - Obtenha a lista de alunos
 - Obtenha um aluno pelo id
 - Atualize um aluno
 - Delete um aluno
 - (CRUD)





PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (1_1) [SOZINHO(A)]

- Através do seu projeto React, crie uma API RESTful fake que retorne uma lista de Cursos. Cada curso terá um nome, descrição, duração e instrutor. Utilize o Insomnia para realizar operações CRUD na lista de cursos.
 - Através do insomnia:
 - Obtenha a lista de cursos
 - Obtenha um curso pelo id
 - Atualize um curso
 - Delete um curso
 - (CRUD)





PROFKELSON.DEV

BORA CODAR?

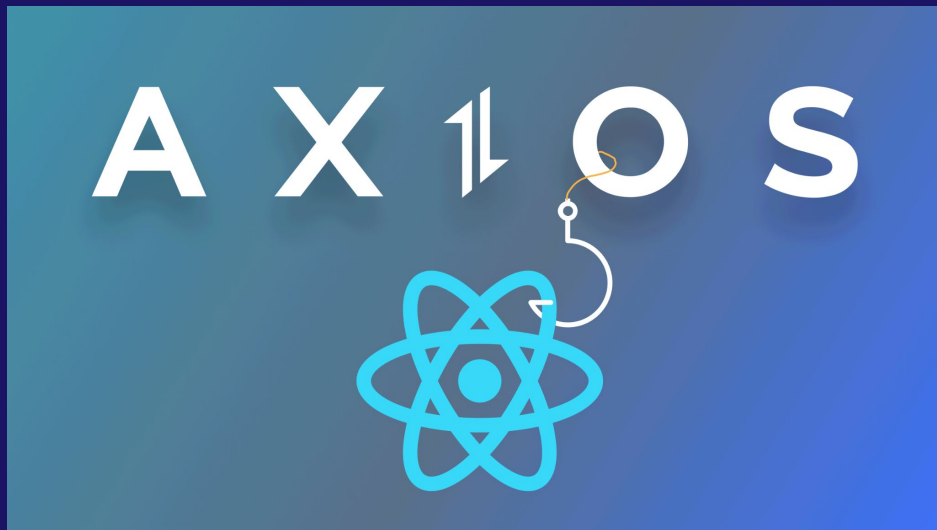
VAMOS CODAR? (1_2) [SOZINHO(A)]

- Através do seu projeto React, crie uma API RESTful fake que retorne uma lista de Projetos. Cada projeto terá um nome, descrição, data de início, data de término e status. Utilize o Insomnia para realizar operações CRUD na lista de projetos.
 - Através do insomnia:
 - Obtenha a lista de projetos
 - Obtenha um projeto pelo id
 - Atualize um projeto
 - Delete um projeto
 - (CRUD)



Axios

- Axios é uma biblioteca JavaScript popular usada para realizar requisições HTTP.
- Funciona tanto no navegador quanto em Node.js.
- Facilita o envio de requisições assíncronas para REST APIs.
- Integração simples com projetos React.
- Suporta interceptores de requisições e respostas.
- Transformação automática de dados JSON.
- Instalação:
 - `npm install axios`



Axios: Realizando um GET

- Exemplo de como buscar dados de uma API.
- Uso de `axios.get(url)`.
- Importamos `React`, `useEffect`, e `useState` do `React` para gerenciar o estado e o ciclo de vida do componente. Também importamos o `axios` para fazer a requisição HTTP.

```
jsx

import React, { useEffect, useState } from 'react';
import axios from 'axios';

function FetchData() {
  const [data, setData] = useState([]);

  useEffect(() => {
    axios.get('https://api.example.com/data')
      .then(response => {
        setData(response.data);
      })
      .catch(error => console.error("There was an error!", error));
  }, []);

  return (
    <div>
      {data.map(item => (
        <div key={item.id}>{item.title}</div>
      ))}
    </div>
  );
}
```

Axios: Realizando um POST

- Enviando dados para um servidor ou API.
- Exemplo de uso de `axios.post(url, data)`.
- Função `postData`: Declaramos uma função assíncrona `postData` que será responsável por enviar uma requisição POST.

jsx

```
import axios from 'axios';

const postData = async () => {
  try {
    const response = await axios.post('https://api.example.com/data', {
      title: 'Your Title',
      body: 'Your body content',
      userId: 1,
    });
    console.log(response.data);
  } catch (error) {
    console.error("There was an error!", error);
  }
};
```



PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (2) [COM O PROF]

- Utilizando o backend falso criado no exercício anterior, utilize o axios para:
 - Adicionar um novo aluno no banco de dados
 - Listar todos os alunos
- Crie os componentes AdicionarAluno e ListarAlunos





PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (2_1) [SOZINHO(A)]

- Aproveite o exercício do “Formulário de Contato” da aula anterior.
- Utilize o json-server + axios para enviar o contato para o banco de dados do axios.





PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (2_2) [SOZINHO(A)]

- Aproveite o exercício do “Formulário de Registro” da aula anterior.
- Utilize o json-server + axios para enviar o registro para o banco de dados do axios.





PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? (2_3) [SOZINHO(A)]

- Faça um componente que liste uma tabela com a listagem de alunos, contatos e registros.



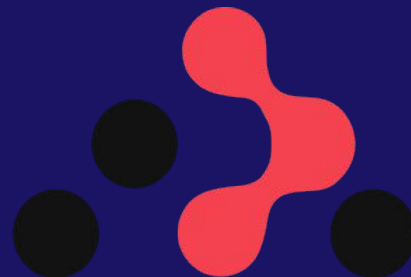


PROFKELSON.DEV

BORA CODAR?

React Router

- React Router é uma biblioteca JS de roteamento para aplicações React.
- Roteamento, prof? Sim!
- Apesar do React trabalhar com o SPA (Single Page Application), podemos criar várias rotas e URLs amigáveis para a nossa aplicação;



React Router

```
`npm install react-router-dom`
```



PROFKELSON.DEV

BORA CODAR?

React Router

- Precisaremos três importações:
 - BrowserRouter (área que vai trocar as páginas)
 - Routes (Definição das rotas)
 - Route (rota com path e componente desta rota)

```
1 import { BrowserRouter, Routes, Route } from 'react-router-dom'
2 import Inicial from './pages/Inicial'
3 import Contato from './pages/Contato'
4 import Produtos from './pages/Produtos'
5
6 function App() {
7   return (
8     <>
9     <BrowserRouter>
10    <h1>Título da Página</h1>
11    <Routes>
12      <Route path="/" element={<Inicial />} />
13      <Route path="/contato" element={<Contato />} />
14      <Route path="/produtos" element={<Produtos />} />
15    </Routes>
16    </BrowserRouter>
17  </>
18 )
19 }
20
21 export default App
```

React Router

- Com o Router é possível criar rotas que correspondem a diferentes URLs em sua aplicação e renderizar diferentes componentes React de acordo com o componente selecionado.
- Com isso, é possível que os usuários naveguem em diferentes partes da aplicação sem precisar recarregar a página.



```
1 import { BrowserRouter, Routes, Route } from 'react-router-dom'
2 import Inicial from './pages/Inicial'
3 import Contato from './pages/Contato'
4 import Produtos from './pages/Produtos'
5
6 function App() {
7   return (
8     <>
9     <BrowserRouter>
10    <h1>Título da Página</h1>
11    <Routes>
12      <Route path="/" element={<Inicial />} />
13      <Route path="/contato" element={<Contato />} />
14      <Route path="/produtos" element={<Produtos />} />
15    </Routes>
16    </BrowserRouter>
17  </>
18 )
19 }
20
21 export default App
```

React Router

- Com o Router é possível criar rotas que correspondem a diferentes URLs em sua aplicação e renderizar diferentes componentes React de acordo com o componente selecionado.
- Com isso, é possível que os usuários naveguem em diferentes partes da aplicação sem precisar recarregar a página.

← → ↻ ⓘ 127.0.0.1:5173/contato

Título da Página

Contato

```
1 import { BrowserRouter, Routes, Route } from 'react-router-dom'
2 import Inicial from './pages/Inicial'
3 import Contato from './pages/Contato'
4 import Produtos from './pages/Produtos'
5
6 function App() {
7   return (
8     <>
9     <BrowserRouter>
10    <h1>Título da Página</h1>
11    <Routes>
12      <Route path="/" element={<Inicial />} />
13      <Route path="/contato" element={<Contato />} />
14      <Route path="/produtos" element={<Produtos />} />
15    </Routes>
16    </BrowserRouter>
17  </>
18 )
19 }
20
21 export default App
```

React Router

- Com o Router é possível criar rotas que correspondem a diferentes URLs em sua aplicação e renderizar diferentes componentes React de acordo com o componente selecionado.
- Com isso, é possível que os usuários naveguem em diferentes partes da aplicação sem precisar recarregar a página.



```
1 import { BrowserRouter, Routes, Route } from 'react-router-dom'
2 import Inicial from './pages/Inicial'
3 import Contato from './pages/Contato'
4 import Produtos from './pages/Produtos'
5
6 function App() {
7   return (
8     <>
9     <BrowserRouter>
10    <h1>Título da Página</h1>
11    <Routes>
12      <Route path="/" element={<Inicial />} />
13      <Route path="/contato" element={<Contato />} />
14      <Route path="/produtos" element={<Produtos />} />
15    </Routes>
16    </BrowserRouter>
17  </>
18 )
19 }
20
21 export default App
```




PROFKELSON.DEV

BORA CODAR?

Navbar será com componente em “components”

React Router

- Mas prof, eu preciso de uma navegação mais intuitiva para o usuário, ele não é obrigado a “decorar”, os links de navegação...
- Então podemos fornecer os links prontos na tela, para que ele possa clicar.

```
8 function App() {  
9   return (  
10     <>  
11     <BrowserRouter>  
12     <h1>Título da Página</h1>  
13     <Navbar />  
14     <Routes>  
15       <Route path="/" element={<Inicial />} />  
16       <Route path="/contato" element={<Contato />} />  
17       <Route path="/produtos" element={<Produtos />} />  
18     </Routes>  
19   )  
20 }
```

```
1 import React from 'react'  
2  
3 import { Link } from "react-router-dom"  
4  
5 const Navbar = () => {  
6   return (  
7     <nav>  
8       <Link to="/">Inicial</Link>  
9       <Link to="/contato">Contato</Link>  
10      <Link to="/produtos">Produtos</Link>  
11    </nav>  
12  )  
13 }  
14  
15 export default Navbar
```




PROFKELSON.DEV

BORA CODAR?

React Router

- Eita, prof... E se eu quisesse fazer uma rota dinâmica? Por exemplo... Abrir uma página (Componente) que carregasse apenas as informações de acordo com um parâmetro passado.
- Para esses casos pode usar o hook "useParams" e carregamos a informação que desejamos visualizar no componente.

```
22   return (  
23     <div>  
24       <h1>Lista de Produtos</h1>  
25       <ul>  
26         {products.map((product) => (  
27           <li key={product.id}>  
28             {product.name} - R$ {product.price} <br/>  
29             <Link to={` /detalhes-do-produto/${product.id}`} >Detalhes</Link>  
30           </li>  
31         )>  
32       </ul>  
33     </div>  
34   )  
35 }  
36  
37 export default Produtos
```

```
8   function App() {  
9     return (  
10       <>  
11         <BrowserRouter>  
12           <h1>Título da Página</h1>  
13           <Navbar />  
14           <Routes>  
15             <Route path="/" element={<Inicial />} />  
16             <Route path="/contato" element={<Contato />} />  
17             <Route path="/produtos" element={<Produtos />} />  
18             <Route path="/detalhes-do-produto/:id" element={<DetalhesDoProduto />} />  
19           </Routes>  
20         </BrowserRouter>  
21       </>  
22     )  
23   }
```

React Router

- Eita, prof... E se eu quisesse fazer uma rota dinâmica? Por exemplo... Abrir uma página (Componente) que carregasse apenas as informações de acordo com um parâmetro passado.
- Para esses casos pode usar o hook "useParams" e carregamos a informação que desejamos visualizar no componente.

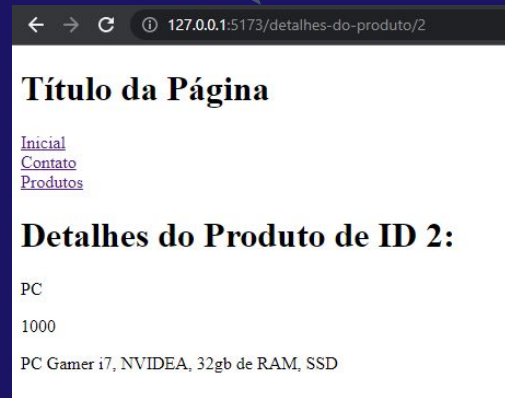


PROFKELSON.DEV

```
2 import { useParams } from 'react-router-dom'
3 import { useState, useEffect } from 'react'
4
5 const DetalhesDoProduto = () => { BORA CODAR?
6
7   const { id } = useParams()
8
9   const url = `http://localhost:3000/products/${id}`
10
11   const [productDetails, setProductDetails] = useState({})
12
13   // 1 - resgatando dados
14   useEffect(() => {
15     async function fetchData() {
16
17       const res = await fetch(url)
18
19       const data = await res.json()
20
21       setProductDetails(data)
22     }
23     fetchData()
24   }, [])
25
26   return (
27     <>
28     <h1>Detalhes do Produto de ID {id}</h1>
29     <p>{productDetails.name}</p>
30     <p>{productDetails.price}</p>
31     <p>{productDetails.details}</p>
32   )
33 }
```

React Router

- Eita, prof... E se eu quisesse fazer uma rota dinâmica? Por exemplo... Abrir uma página (Componente) que carregasse apenas as informações de acordo com um parâmetro passado.
- Para esses casos pode usar o hook "useParams" e carregamos a informação que desejamos visualizar no componente.





PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? [COM O PROF]

- Crie um sistema de informações de um site fictício chamado TechHub, que oferece artigos sobre tecnologia. A aplicação deve conter as seguintes funcionalidades:
- Uma Navbar com links para:
 - Página Inicial
 - Página de Artigos
 - Página "Sobre nós"
- Um componente que liste todos os artigos:
 - Este componente deve consumir os dados de um back-end fake.
 - No banco de dados, cada artigo deve ter as propriedades:
 - Título
 - Resumo (pequena introdução ao conteúdo)
 - Conteúdo (texto completo do artigo)
 - O componente de listagem deve exibir o Título e o Resumo de cada artigo, com links para visualizar os detalhes.
- Um componente para visualizar um artigo:
 - Este componente deve consumir os dados do back-end fake para buscar as informações do artigo pelo id.
 - Deve exibir todas as propriedades do artigo (exceto o id).
- Inclua também uma página "Sobre nós", explicando a missão do site TechHub.





PROFKELSON.DEV
BORA CODAR?

VAMOS CODAR? [SOZINHO(A)]

Crie um sistema para um site fictício chamado **BookSpot**, especializado em resenhas de livros. A aplicação deve conter as seguintes funcionalidades:

1. Uma **Navbar** com links para:
 - Página Inicial
 - Página de Resenhas
 - Página "Sobre o BookSpot"
2. Um **componente que liste todas as resenhas**:
 - Este componente deve consumir dados de um back-end fake.
 - No banco de dados, cada resenha deve ter as seguintes propriedades:
 - Título do Livro
 - Autor
 - Resumo (um pequeno trecho da resenha)
 - Conteúdo Completo (o texto completo da resenha)
 - A listagem deve exibir o **Título do Livro**, o **Autor** e o **Resumo** de cada resenha, com links para visualizar os detalhes.
3. Um **componente para visualizar uma resenha**:
 - Deve consumir os dados do back-end fake para buscar as informações pelo id.
 - Deve exibir todas as propriedades da resenha (exceto o id).
4. Inclua uma **página "Sobre o BookSpot"**, explicando a missão do site.

