

SW-II

SISTEMAS WEB II

Prof. Anderson Vanin

AULA 18 – Conectando o Frontend ao Backend – Parte 3

Criando um novo usuário com os dados do formulário

Rodar o Servidor e o FrontEnd

- Abra uma aba do VS code com os arquivos de seu servidor
- Abra outra aba do VS code com os arquivos de seu FRONT END

etecmcm-cadastro-usuarios > src > pages > Home > index.jsx > Home

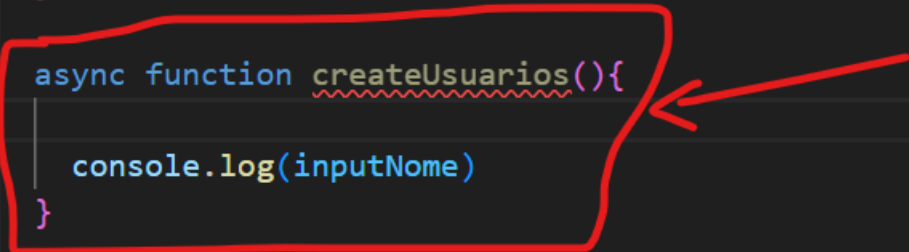
```
1 import { useEffect, useState, useRef } from 'react'
2 import './style.css'
3 import Lixeira from '../../../assets/lixeira_peq.svg'
4 import api from '../../../services/api'
5
6
7 function Home() {
8   const [usuarios, setUsuarios] = useState([])
9   //let usuarios = []
10
11   const inputNome = useRef()
12   const inputEmail = useRef()
13   const inputIdade = useRef()
14
15
16   async function getUsuarios(){
17     const usuariosDaApi = await api.get('/cadastro')
18     //setUsuarios = usuariosDaApi.data
19     setUsuarios(usuariosDaApi.data)
```

```
return (  
  <div className='container'>  
    <form>  
      <h1>Cadastro de Usuários</h1>  
      <input placeholder='Digite seu nome' name='nome' type='text' ref={inputNome} />  
      <input placeholder='Digite sua idade' name='idade' type='number' ref={inputIdade} />  
      <input placeholder='Digite seu email' name='email' type='email' ref={inputEmail}/>  
      <button type='button'>Cadastrar</button>  
    </form>  
  
    {usuarios.map(usuario => (  
      <div key={usuario.id} className='card'>  
        <div>  
          <p>Nome: <span>{usuario.nome}</span></p>  
          <p>Idade: <span>{usuario.idade}</span></p>  
        </div>  
      </div>  
    ))}  
  </div>  
)
```





```
async function getUsers(){
  const usuariosDaApi = await api.get('/cadastro')
  //setUsuarios = usuariosDaApi.data
  setUsuarios(usuariosDaApi.data)
  console.log(usuarios)
}
```

```
async function createUsuarios(){
  console.log(inputNome)
}
```



```
useEffect(()=>{
  getUsers()
},[])
```

```
return (
  <div className='container'>
    <form>
      <h1>Cadastro de Usuários</h1>
      <input placeholder='Digite seu nome' name='nome' type='text' ref={inputNome} />
      <input placeholder='Digite sua idade' name='idade' type='number' ref={inputIdade} />
      <input placeholder='Digite seu email' name='email' type='email' ref={inputEmail}/>
      <button type='button' onClick={createUsuarios}>Cadastrar</button>
    </form>
    {usuarios.map(usuario => (
```



Cadastro de Usuários

ANDERSON SILVA V

Digite sua idade

Digite seu email

Cadastrar

Download the React DevTools for a better development experience

Array(0)

Array(0)

{current: input} i

current: input

value: "ANDERSON SILVA VANIN"

__reactEvents\$4yrbc1awapm: Set(1) {'invalid__bubble'}

__reactFiber\$4yrbc1awapm: FiberNode {tag: 5, key: null}

__reactProps\$4yrbc1awapm: {placeholder: 'Digite seu email'}

_valueTracker: {getValue: f, setValue: f, stopTracking: f}

accept: ""

accessKey: ""

align: ""

alt: ""

ariaActiveDescendantElement: null

ariaAtomic: null

ariaAutoComplete: null

ariaBrailleLabel: null

ariaBrailleRoleDescription: null

ariaBusy: null

ariaChecked: null

ariaColCount: null

ariaColIndex: null

ariaColIndexText: null

ariaColSpan: null

ariaContentElement: null

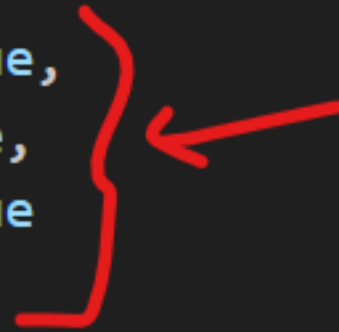
Relembrando o método Post no servidor

```
app.post('/cadastro', async (req, res) => {  
  await prisma.usuario.create({  
    data: {  
      email: req.body.email,  
      nome: req.body.nome,  
      idade: req.body.idade  
    }  
  })  
  res.status(201).json(req.body)  
})
```




```
async function getUsuarios(){  
  const usuariosDaApi = await api.get('/cadastro')  
  //setUsuarios = usuariosDaApi.data  
  setUsuarios(usuariosDaApi.data)  
  console.log(usuarios)  
}
```

```
async function createUsuarios(){  
  await api.post('/cadastro',{  
    email: inputEmail.current.value,  
    nome: inputIdade.current.value,  
    idade: inputIdade.current.value  
  })  
  //console.log(inputNome)  
}
```



Pequeno ajuste

```
useEffect(()=>{  
  getUsuarios()  
  // eslint-disable-next-line react-hooks/exhaustive-deps  
},[])
```



Cadastro de Usuários

Teste front end

19

testedefront@email.com

Cadastrar

DevTools is now available in Portuguese

Don't show again Always match Chrome's language Switch DevTools to Portuguese

Network Elements Console Sources Performance Memory Application >>

Filter

Fetch/XHR Doc CSS JS Font Img Media Manifest Socket Wasm Other

Name	Status	Type	Initiator	Size	Time
cadastro	201	xhr	index.jsx:24	0.3 kB	

1 / 2 requests | 0.3 kB / 0.3 kB transferred | 0.1 kB / 0.1 kB resources

Console AI assistance

top Filter Default levels 1 Issue 2 hidden

react-dom_client.js?v=cdd415a4:1799

Download the React DevTools for a better development experience: <https://react.dev/link/react-devtools>

[] index.jsx:24

[] index.jsx:24

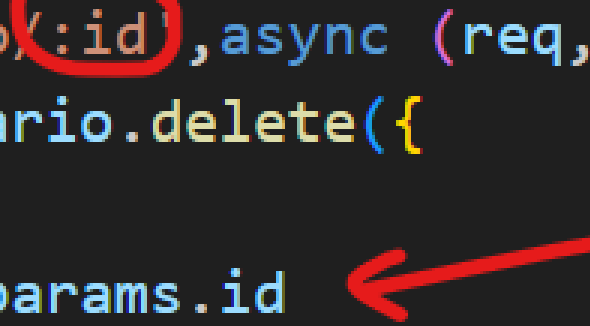
Recarregar o frontend de maneira
automática com novos dados

```
async function createUsuarios(){  
  await api.post('/cadastro',{  
    email: inputEmail.current.value,  
    nome: inputIdade.current.value,  
    idade: inputIdade.current.value  
  })  
  getUsuarios() ←  
}
```

Deletar um usuário

Relembrando o delete da api

```
app.delete('/cadastro/:id', async (req, res) => {  
  await prisma.usuario.delete({  
    where: {  
      id: req.params.id  
    }  
  })  
  res.status(200).json({ "message": "Usuário Deletado com sucesso!" })  
})
```

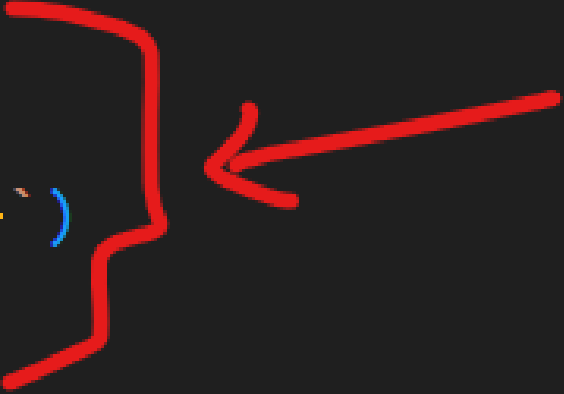


Criar a função para apagar

```
    idade: inputIdade.current.value
  })
  getUsuarios()
}

async function deleteUsuarios(id){
  await api.delete(`/cadastro/${id}`)
}

useEffect(()=>{
  getUsuarios()
  // eslint-disable-next-line react-hooks/exhaustive-dep
```

A red bracket is drawn on the right side of the code, grouping the `deleteUsuarios` function and its call inside the `useEffect` hook. A red arrow points from the right towards the bracket, highlighting the function.

Chamar a função de deletar e passar o valor do id

- É necessário o uso de uma arrow function na chamada da função, pois se fazer a chamada por exemplo: `deleteUsuarios(usuario.id)` o React irá travar!

```
</form>

{usuarios.map(usuario => (
  <div key={usuario.id} className='card'>
    <div>
      <p>Nome: <span>{usuario.nome}</span></p>
      <p>Idade: <span>{usuario.idade}</span></p>
      <p>Email: <span>{usuario.email}</span></p>
    </div>
    <button onClick={() => deleteUsuarios(usuario.id)}>
      <img src={Lixeira} />
    </button>
  </div>
))}
```

```
</div>
```

Tarefas

- Após deletar um usuário, faça com que a tela seja atualizada! (* critério de avaliação da nota de 3 Bim)
- Implementar a função de Update (* critério de avaliação da nota de 3 Bim)
 - Criar um botão para atualizar e a tela de atualizar
- Entregar (via github) dois links dos repositórios:
 1. Projeto da API (Backend) – link 1 (* critério de avaliação da nota de 3 Bim)
 2. Projeto do FrontEnd – link 2 (* critério de avaliação da nota de 3 Bim)