

REACT NA PRÁTICA

FELIPE N. MOURA

SOBRE O FELIPE

Apaixonado por desenvolvimento há quase 20 anos.

Co-fundou a BrazilJS.

Co-fundou a On2.

Pai, trekkers, toca um violão, joga um video game.



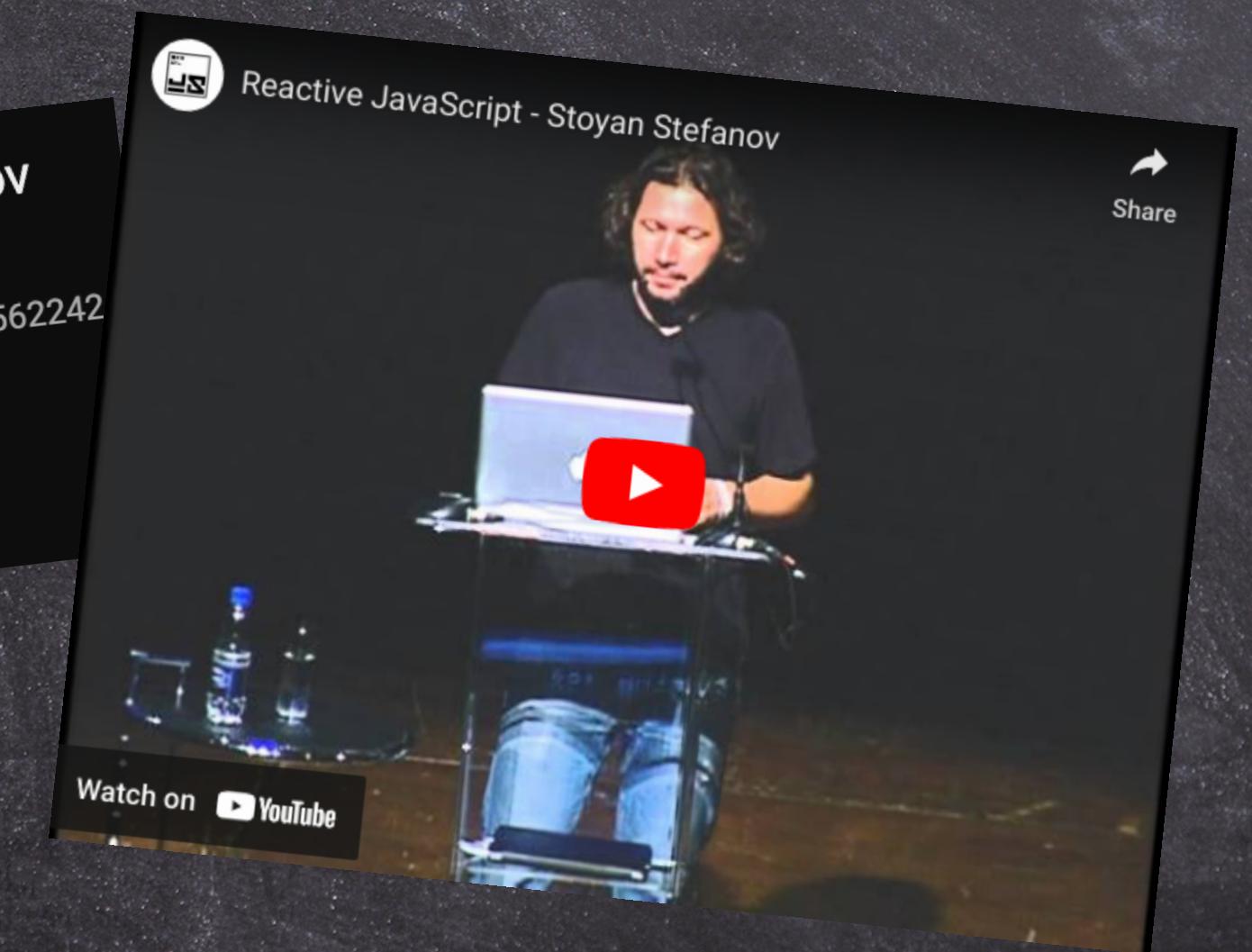
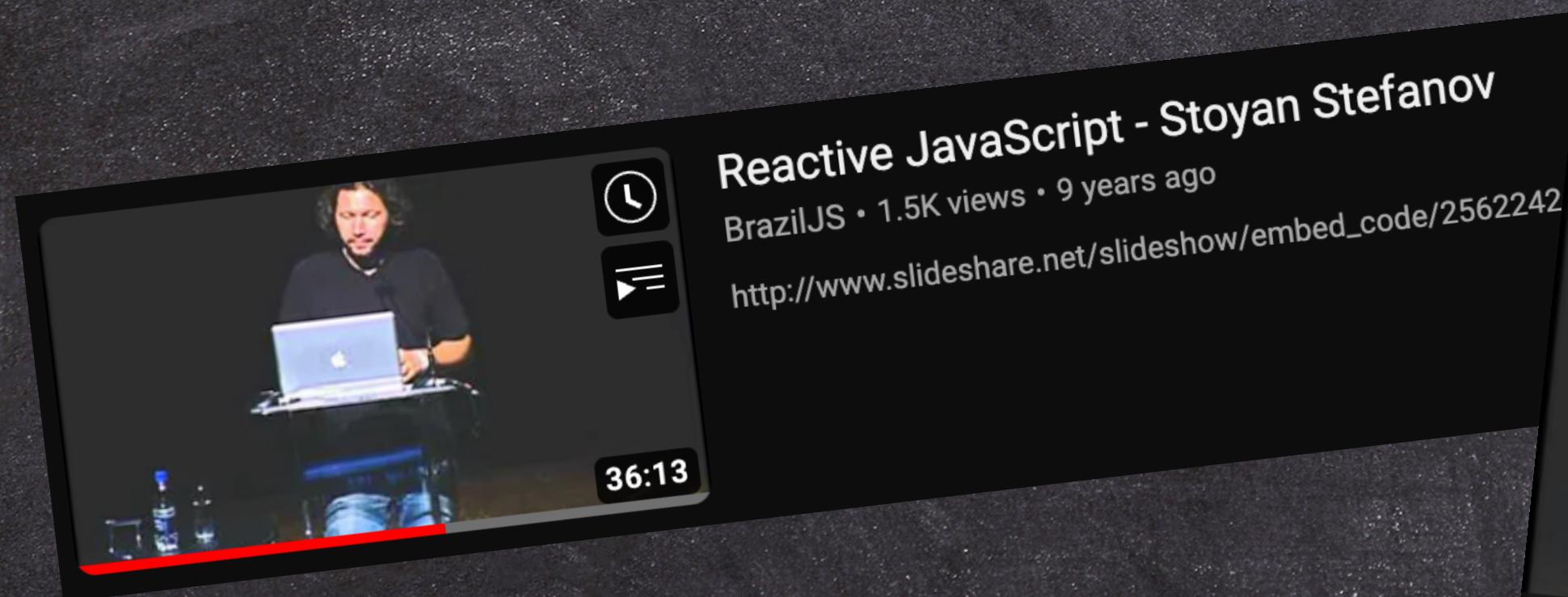
SOBRE O WORKSHOP

React se tornou padrão no desenvolvimento Web. O projeto segue evoluindo e influenciando a comunidade tech no mundo.

Neste workshop serão apresentados os principais conceitos do React, incluindo suas últimas funcionalidades.

Também serão abordadas suas vantagens, bem como exemplos práticos e construção de aplicação demo.

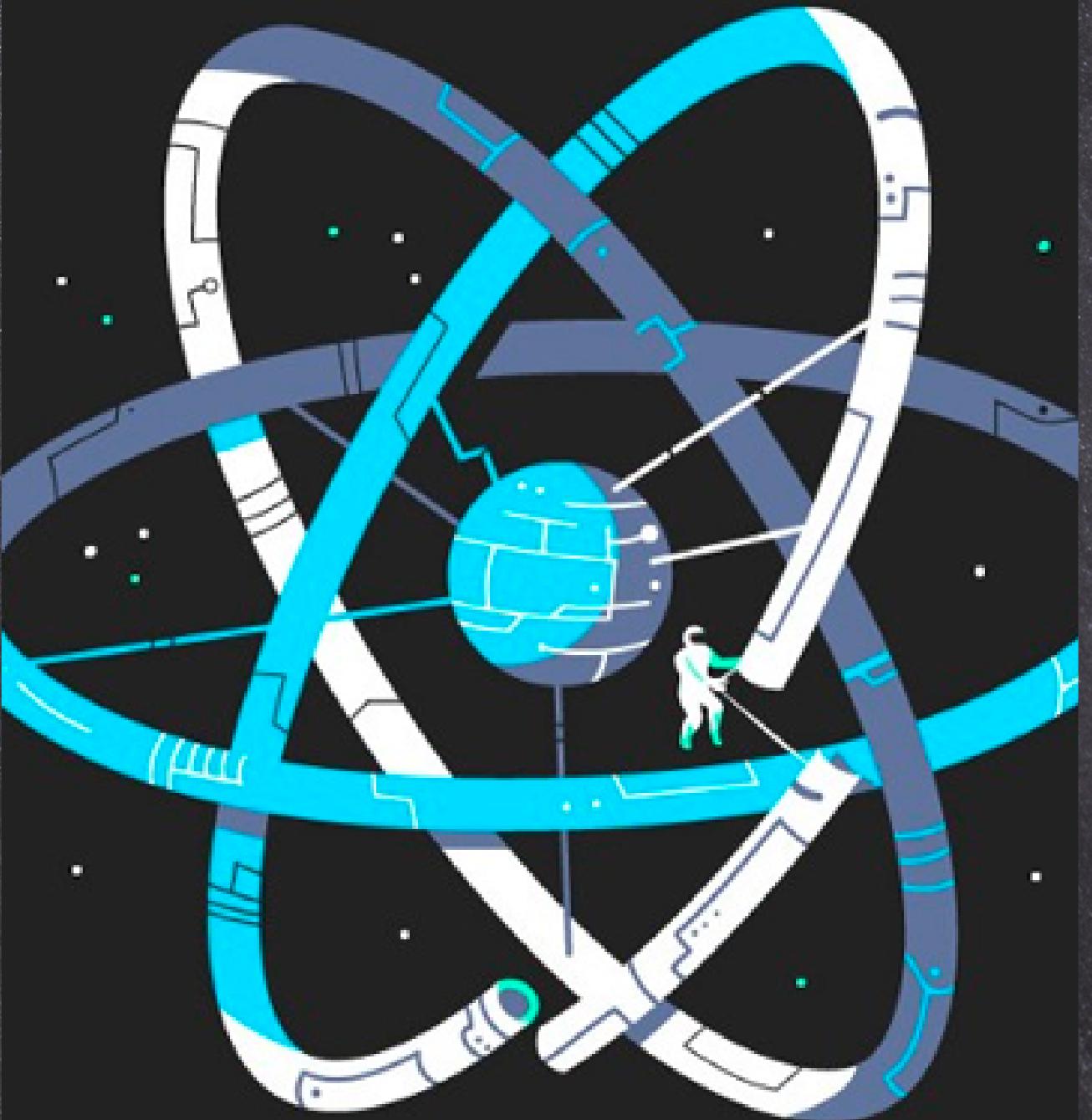
ANÚNCIO NA BRAZILJS 2013



COM O TEMPO: MATURIDADE E MELHORIAS

Com o avanço do próprio JavaScript e seus padrões sendo mais amplamente suportados por cada vez mais browsers, o próprio React escutou a comunidade e evoluiu usufruindo destas funcionalidades.

- Promises
- Destructuring Assignment & Arguments
- Arrow Functions
- Template Strings
- aynsc/await
- etc...



O QUE EXERCITAREMOS

Vamos ver algumas das features mais importantes e entender elas na prática:

- props
- useState
- useEffect
- mount/unmount
- iterações & renderizações
- JavaScript



O DESAFIO

Usaremos o seguinte desafio como base para o estudo:

- Uma página que busque a lista de estados brasileiros
- Quando o usuário seleciona um estado, vamos listar todas as cidades

E se der tempo, um bonus:

- Teremos um campo de filtro para encontrar uma cidade específica



codesandbox

INSTALAÇÃO LOCAL

PNPM (opcional):
`npm i -g pnpm`

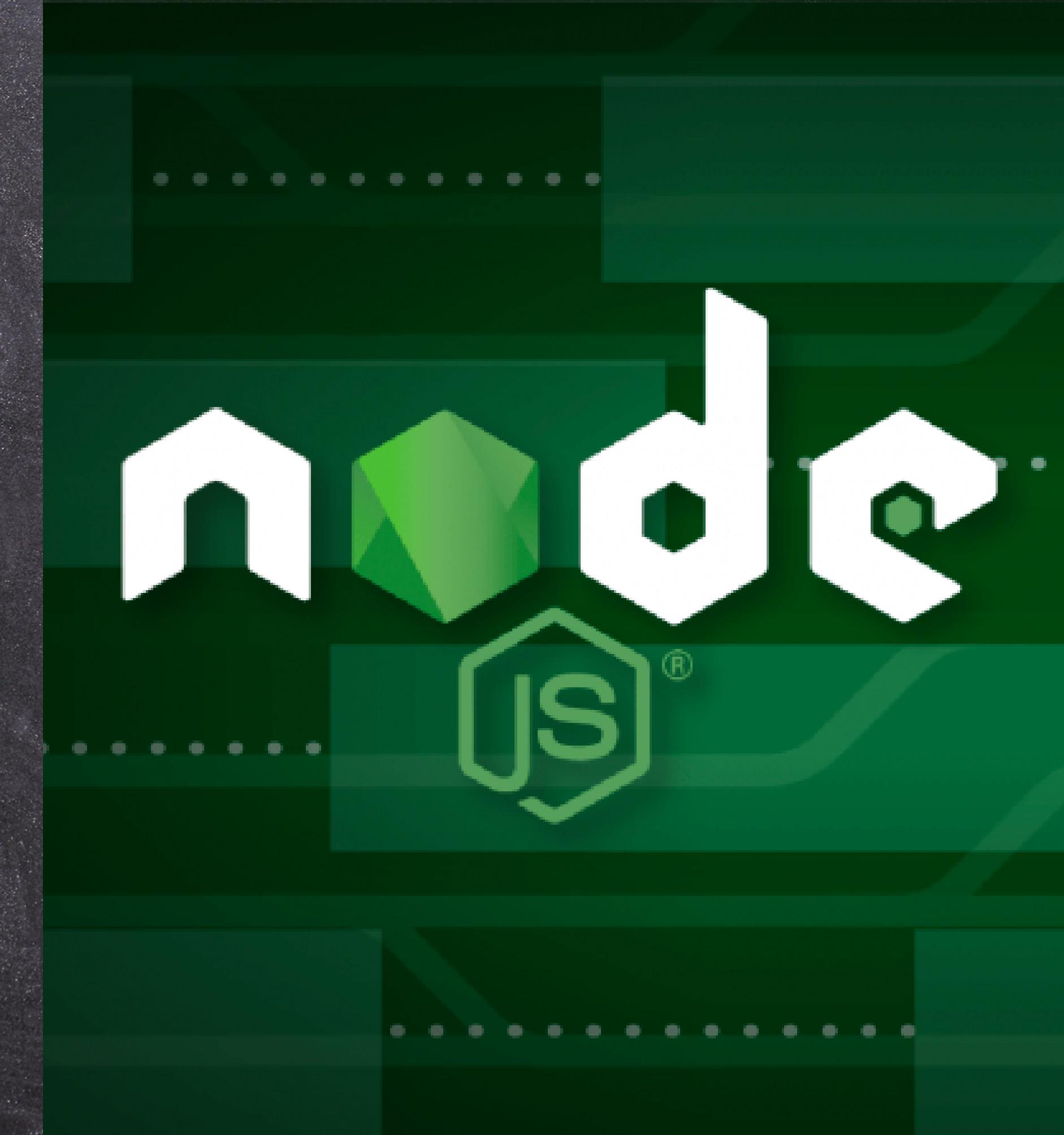
Criando a base do projeto:
`npx create-react-app@latest uf-cidades --use-pnpm`

Abrindo no editor:
`code ./uf-cidades`

Opcionalmente, podemos adotar o Sass (SCSS)

Instale a dependência de Sass
`pnpm add node-sass`

Aplique o uso das extensões:
Renomeie o arquivo **App.css** para **App.scss** e altere o
import no arquivo **App.js**.



URLS ÚTEIS

Lista de estados:

[https://servicodados.ibge.gov.br/api/v1/localidades/estados?
orderBy=nome](https://servicodados.ibge.gov.br/api/v1/localidades/estados?orderBy=nome)



Lista de cidades de um Estado:

[https://servicodados.ibge.gov.br/api/v1/localidades/estados/
\\${UF}/distritos](https://servicodados.ibge.gov.br/api/v1/localidades/estados/{UF}/distritos)



REACT HOOKS

React Hooks nos permitem disparar ou escutar alterações no resultado final a ser renderizado.

REACT HOOKS

O hook useState nos retorna uma array onde:

- [0] uma variável contendo o valor atual do estado
- [1] uma função que altera o valor do estado

Este hook pode receber um valor inicial por parâmetro.

```
export default function App() {  
  const [statesList, setStatesList] = React.useState(null);  
  
  return <div></div>;  
}
```

REACT HOOKS

Já o hook `useEffect` nos permite escutar alterações em states para dispararmos ação.

Para este hook, passamos dois parâmetros, o primeiro é a função a ser disparada, e o segundo, é uma lista de estados/valores a serem observados.

Nossa função pode retornar `undefined` ou uma nova função, que será executada quando seu componente for desmontado.

O segundo parâmetro pode ser usado como um "filtro". Se ele for:

- **`undefined`**: A função será disparada a cada render
- **`[]`**: A função será chamada apenas uma vez, no primeiro render
- **`[state1, state2, etc]`**: A função será chamada sempre que um dos estados for alterado.

```
React.useEffect(_ => {
  // executado apenas uma vez
  // equivalente a onMount

  return function () {
    // equivalente a onUnmount
  }
}, []);
```

```
React.useEffect(_ => {
  // executado sempre que o componente renderizar
  // deve acabar sendo executada muitas vezes
}); // <-- por que não estamos passando nenhum filtro
```

```
React.useEffect(_ => {
  // executado sempre que o valor em statesList for alterado
}, [statesList]);
```

REACT HOOKS

Vamos usar o `useEffect` para garantir que a requisição da lista de estados será feita apenas uma vez.
E vamos guardar esta lista em um estado.

```
const API_STATES_UF = `https://servicodados.ibge.gov.br/api/  
  
export default function App() {  
  const [statesList, setStatesList] = React.useState(null);  
  
  React.useEffect(() => {  
    fetch(API_STATES_UF)  
      .then(async (response) => {  
        const data = await response.json();  
        setStatesList(data);  
      })  
      .catch((error) => {  
        alert("Failed loading countries data!");  
        console.error(error);  
      });  
  }, []);  
  
  return <div></div>  
}
```

RENDERIZANDO

Vamos renderizar os dados usando chaves.

```
// enquanto não temos a lista
if (!statesList) {
  return <div className="no-data">Building up...</div>;
}

// quando tivermos a lista
return <div>
  {statesList?.length}
</div>;
```

RENDERIZANDO

Podemos assim renderizar as opções dentro do loop.

```
<select onChange={onStateChange}>
  <option value=""></option>
  {
    statesList.map((state) => {
      return (
        <option
          key={state.sigla}
          value={state.sigla}
        >
          {state.sigla} - {state.nome}
        </option>
      );
    })
  }
</select>
```

RENDERIZANDO

E teremos alguns estados a serem controlados.

```
const [selectedState, setSelectedState] = React.useState(null);
const [citiesForState, setCitiesForCurrentUF] = React.useState(null);

function onStateChange(event) {
  setCitiesForCurrentUF(null);
  setSelectedState(event.target.value);
}
```

```
<select onChange={onStateChange}>
  <option value=""></option>
  {
    statesList.map((state) => {
      return (
        <option
          key={state.sigla}
          value={state.sigla}
        >
          {state.sigla} - {state.nome}
        </option>
      );
    })
  }
</select>
```

RENDERIZANDO

Um botão para solicitar pelas cidades do estado selecionado.

Esse botão somente estará habilitado quando houver um estado selecionado.

E quando clicarmos no botão, vamos buscar a lista de cidades.

```
<button disabled={!selectedState} onClick={getCities}>  
  Ok  
</button>
```

BUSCANDO AS CIDADES

Vamos criar um cache local das cidades já carregadas, e no estado da lista de cidades, mantemos apenas as cidades do estado selecionado.

Teremos um estado para informar se estamos buscando a lista de cidades (loading).

E usaremos uma templateString para montar a url da chamada para a API

```
const CACHED_CITIES = {};
```

```
const [loadingCities, setLoadingCities] = React.useState(null);
```

```
const getCitiesAPI_URL = function (UF = "") {
  UF = UF.substring(0, 2);
  return `https://servicodados.ibge.gov.br/api/v1/localidades/estados/${UF}/distritos`;
};
```

BUSCANDO AS CIDADES

Usaremos o `fetch` para buscar a lista de cidades, mas somente se já não estava na nossa variável de cache.

Usaremos o status de loading para que usuários saibam que a busca está sendo feita.

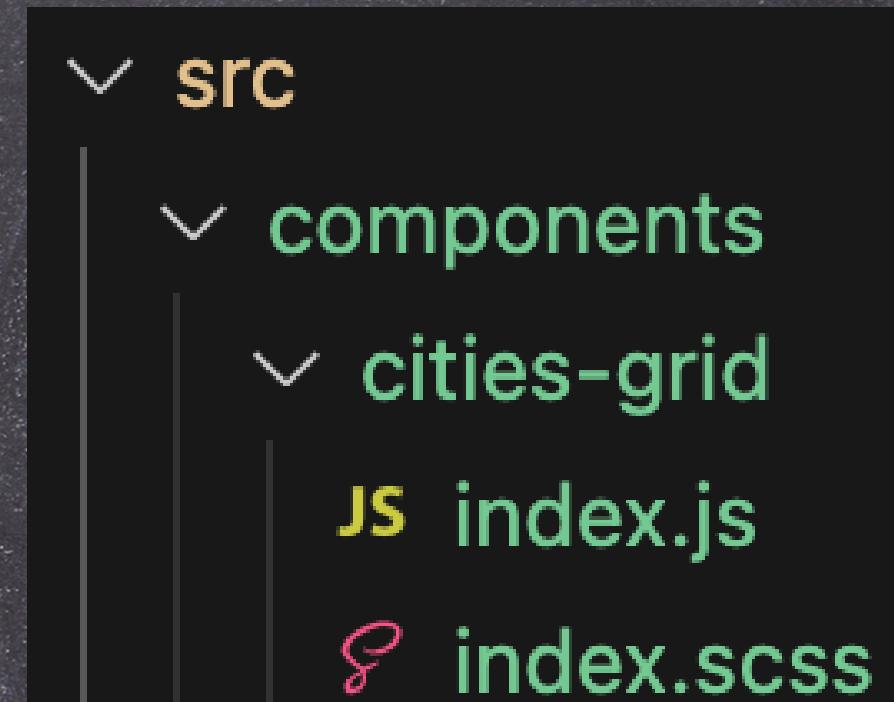
```
function getCities() {  
  if (CACHED_CITIES[selectedState]) {  
    return setCitiesForCurrentUF(CACHED_CITIES[selectedState]);  
  }  
  
  setLoadingCities(true);  
  fetch(getCitiesAPI_URL(selectedState))  
    .then(async (response) => {  
      const data = await response.json();  
      CACHED_CITIES[selectedState] = data;  
      setCitiesForCurrentUF(data);  
    })  
    .catch((error) => {  
      alert("Erro ao buscar a lista de cidades!");  
      console.error(error);  
    })  
    .finally(() => {  
      setLoadingCities(false);  
    });  
}
```

RENDERIZANDO CIDADES

Vamos criar um novo componente em uma pasta `src/components`, para uma melhor organização.

Daí importaremos ele na nossa App e usaremos ele no render.

Vamos passar para este componente alguns dos estados que estamos controlando.



```
import CitiesGrid from "./components/cities-grid";
```

```
<div className="main">
  <CitiesGrid
    loading={loadingCities}
    state={selectedState}
    cities={citiesForState}
  />
</div>
```

RENDERIZANDO CIDADES

Neste componente, vamos decidir o que mostrar quando estivermos em estado de loading e quando tivermos cidades para exibir ou não.

```
export default function CitiesGrid (props) {
  const {
    className = '',
    children,
    loading=false,
    cities=[],
    state=null,
    ...other
  } = props;

  if (loading) {
    return <div>Carregando: {state}</div>;
  }

  if (!loading && !cities?.length) {
    return <div>Selecione um estado e clique em OK</div>;
  }

  return (
    <div
      className={"cities-grid-container " + className}
      {...other}
    >
      {children}
    </div>
  );
}
```

RENDERIZANDO CIDADES

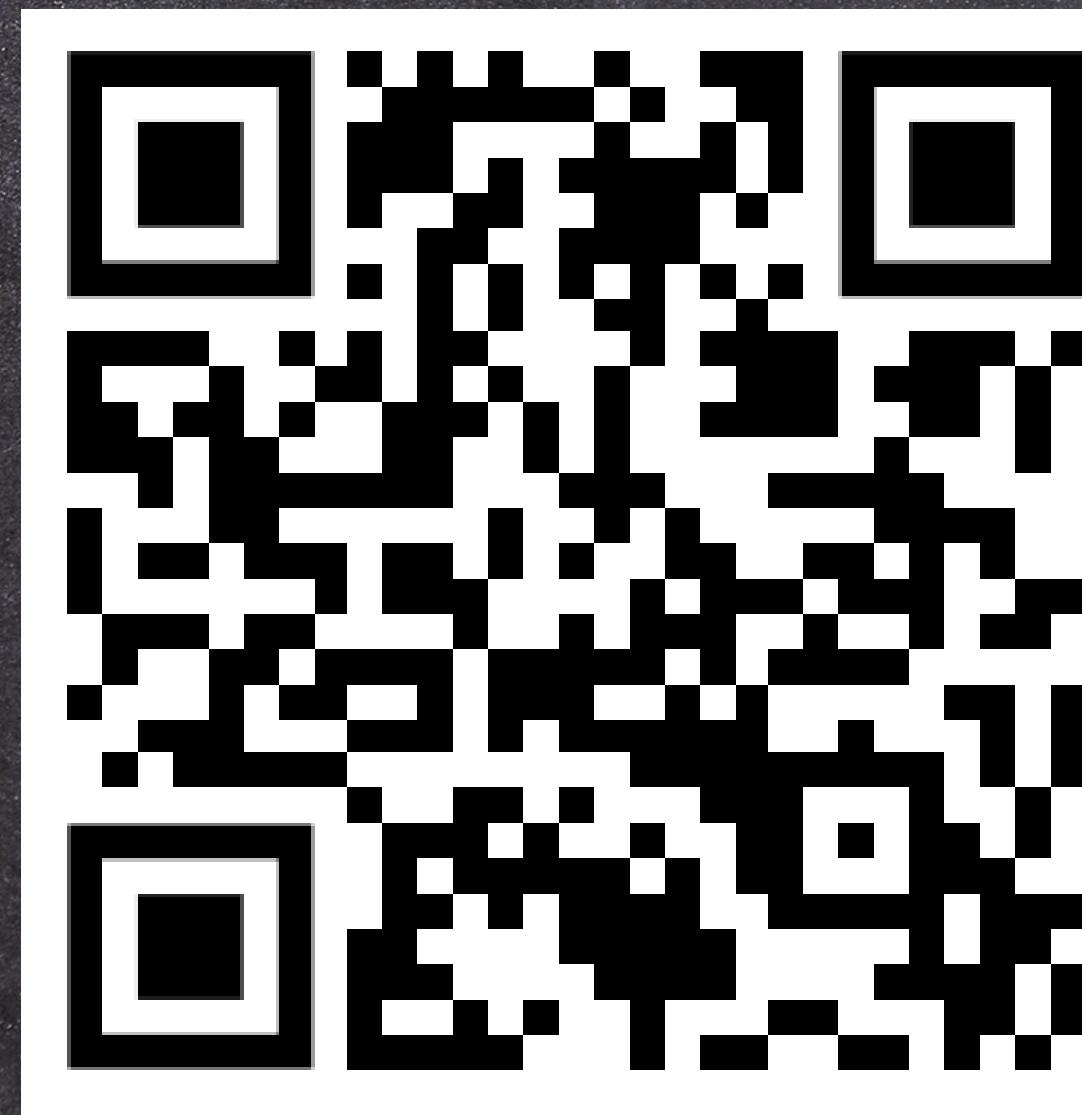
Neste componente, vamos decidir o que mostrar quando estivermos em estado de loading e quando tivermos cidades para exibir ou não.

```
return (
  <div
    className={"cities-grid-container " + className}
    {...other}
  >
    <table className="cities-table">
      <thead>
        <tr>
          <td>Nome</td>
          <td>Microrregiao</td>
        </tr>
      </thead>
      <tbody>
        {cities.map((city) => {
          return (
            <tr>
              <td>{city.nome}</td>
              <td>{city.municipio.microrregiao.nome}</td>
            </tr>
          );
        })}
      </tbody>
    </table>
  </div>
);
```

POSSÍVEIS MELHORIAS

- ORDENAR AS CIDADES
- CAMPO DE FILTRO

GITHUB REPOSITORY



[HTTPS://GITHUB.COM/ON2-DEV/WORKSHOP-REACT-BBDW](https://github.com/on2-dev/workshop-react-bbdw)

CONCLUSÃO

Este é um projeto simples, mas que visa apresentar o uso de funcionalidades das linguagens de forma integrada com o React.

React é muito poderoso, mas é fundamental entender suas limitações e desafios para saber como resolver determinados problemas.

Atividades como estas são comuns no dia a dia e usamos hoje como um teste para avaliação técnica também.

OBRIGADO