

Configurando o frontend

Para criar a aplicação, dentro da pasta 4oanos, digite o comando:

yarn create react-app frontend

Depois que o comando finalizar, digite os comandos `cd frontend` para entrar na pasta e o `dir` para visualizar os arquivos da pasta, o create react app é um boilerplate de projetos para react que vem com toda configuração do babel, webpack. Ele transpila o código JS de versões mais atualizadas para versões antigas. Adicione a pasta frontend dentro do visual studio code.

Clique no arquivo package.json para visualizar o seu conteúdo e veja que foi criado quatro scripts: start, build, teste e eject.

```
frontend > {} package.json > ...
1  {}
2  "name": "frontend",
3  "version": "0.1.0",
4  "private": true,
5  "dependencies": {
6    "react": "^16.9.0",
7    "react-dom": "^16.9.0",
8    "react-scripts": "3.1.1"
9  },
10 "scripts": {
11   "start": "react-scripts start",
12   "build": "react-scripts build",
13   "test": "react-scripts test",
14   "eject": "react-scripts eject"
15 },
16 "eslintConfig": {
17   "extends": "react-app"
18 },
19 "browserslist": {
20   "production": [
21     ">0.2%",
22     "not dead",
23     "not op_mini all"
```

Observe também que foi criado a pasta `node_modules`, a pasta `src` que contém o código que será manipulado e a pasta `public` que armazena arquivos que serão disponibilizados publicamente. Visualize o conteúdo do arquivo `index.html` e nele uma `div` com o id `root`. Quando a página for carregada o react encontrará esta `div` e todo o conteúdo html da aplicação será exibido nesta posição.

Apagando arquivos desnecessários

Dentro da pasta src, apague os arquivos: **App.css**, **App.test.js**, **index.css**, **logo.svg** e o **serviceWorker.js**, deixando somente o **App.js** e o **index.js**.

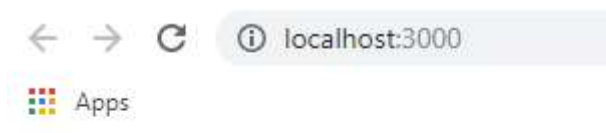
Configure o arquivo **src/index.js** conforme print a seguir

```
JS index.js  X
frontend > src > JS index.js
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import App from './App';
4
5  ReactDOM.render(<App />, document.getElementById('root'));
6
```

Configure o arquivo **src/App.js**, conforme print a seguir.

```
JS App.js  X
frontend > src > JS App.js > ...
1  import React from 'react';
2
3  function App() {
4    return (
5      <div className="App">
6        <h1>Hello World</h1>
7      </div>
8    );
9  }
10
11  export default App;
```

Abra o terminal dentro do visual studio code e digite o comando **yarn start** para iniciar a aplicação, observe que a página abaixo será exibida no navegador.



Hello World

Entendendo o arquivo src/index.js

- Na linha 1 importa o react, pois será necessário toda vez que utilizar a sintaxe, por exemplo, <App />, parecida com o html, mas é o jsx (uma mistura de JS com HTML).
- Na linha 2 importa o react-dom que é a integração do react com o browser, com o mobile é react-native.
- Na linha 3 importa o comando App.
- Na linha 5 será renderizado no navegador o componente <App /> dentro da página index.html com o id root.

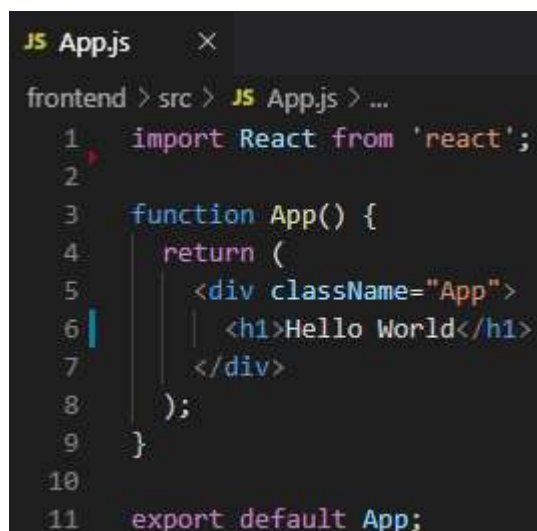


```
JS index.js ×
frontend > src > JS index.js
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import App from './App';
4
5  ReactDOM.render(<App />, document.getElementById('root'));
6
```

O arquivo de entrada do react é index.js.

Entendendo o arquivo src/App.js

O arquivo App.js é um componente, ou seja, um arquivo JS que tem como função retornar um arquivo JSX que é o html presente nas linhas 5, 6 e 7. Ele é um trecho de código que pode ser isolado da aplicação, por exemplo, se a aplicação tem um cabeçalho e este algumas funcionalidades que para o restante da aplicação não tem nenhuma finalidade, então ele pode ser um componente, ele é um conjunto isolado de código estruturado que seria o html, estilização (css) e código de lógica (js) e pode ser escrito no formato de uma função com o comando return ou pode ser uma classe.



```
JS App.js ×
frontend > src > JS App.js > ...
1  import React from 'react';
2
3  function App() {
4    return (
5      <div className="App">
6        <h1>Hello World</h1>
7      </div>
8    );
9  }
10
11  export default App;
```

A partir do momento que está sendo utilizado o react como frontend, tudo é javascript e nada é passado diretamente para o html, por exemplo, para criar uma estilização global para a aplicação que será utilizada em todos os elementos.

Crie o arquivo `src/global.css`

```
# global.css ×
frontend > src > # global.css > ...
1  * {
2      margin: 0;
3      padding: 0;
4      outline: 0;
5      box-sizing: border-box;
6  }
7
8  body {
9      background: #fafafa;
10     font: 14px Arial, Helvetica, sans-serif;
11     -webkit-font-smoothing: antialiased !important;
12 }
```

E você deve importá-lo no arquivo `index.js` ou `App.js` ao invés do `index.html`. Veja o exemplo abaixo importando-o no arquivo `src/index.js` na linha 4.

```
JS index.js ×
frontend > src > JS index.js
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import App from './App';
4  import './global.css'
5
6  ReactDOM.render(<App />, document.getElementById('root'));
7
```

Sua página deverá ficar conforme print a seguir.



Criando as páginas da aplicação

Será criado duas páginas. Crie uma pasta chamada pages e dentro dela os arquivos Feed.js para listar todos os eventos e New.js para criar.

Edite o arquivo src/pages/Feed.js

```
JS Feed.js  X
frontend > src > pages > JS Feed.js > ...
1  import React, { Component } from 'react'
2
3  class Feed extends Component {
4    render() {
5      return (
6        <h1>Feed</h1>
7      )
8    }
9  }
10
11  export default Feed
```

Edite o arquivo src/pages/New.js

```
JS New.js  X
frontend > src > pages > JS New.js > ...
1  import React, { Component } from 'react'
2
3  class New extends Component {
4    render() {
5      return (
6        <h1>New</h1>
7      )
8    }
9  }
10
11  export default New
```

Criando as rotas da aplicação

Digite o comando **yarn add react-router-dom** para instalar este pacote, ele será utilizado para tratar as rotas dentro da url.

Crie o arquivo src/routes.js

O switch garante que somente uma rota seja chamada a cada url que o usuário acessar.

O exact faz uma comparação totalmente igualitária, com ele ao acessar a rota localhost:3000/new será exibido o componente New ao invés do componente Feed.

```

JS routes.js  X
frontend > src > JS routes.js > ...
1  import React from 'react'
2  import { Switch, Route } from 'react-router-dom'
3
4  import Feed from './pages/Feed';
5  import New from './pages/New';
6
7  function Routes() {
8      return (
9          <Switch>
10             <Route path="/" exact component={Feed} />
11             <Route path="/new" component={New} />
12          </Switch>
13      )
14  }
15
16  export default Routes;

```

Importando as rotas no arquivo src/App.js

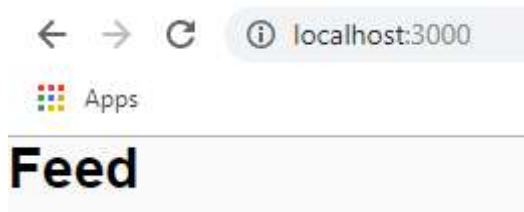
- Insira a linha 3 para importar as rotas.
- Insira a linha 7 para executar as rotas como um componente (Routes).
- O BrowserRouter deve ser usado para todos os componentes com acesso as rotas, ele está inserido no arquivo App.js ao invés do routes.js, pois o componente Header que será criado posteriormente terá acesso as rotas.

```

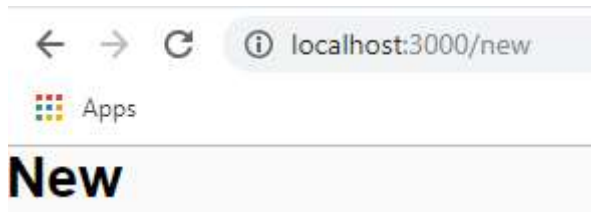
JS App.js  X
frontend > src > JS App.js > ...
1  import React from 'react';
2  import { BrowserRouter } from 'react-router-dom'
3
4  import Routes from './routes'
5
6  function App() {
7      return (
8          <BrowserRouter>
9              <Routes />
10          </BrowserRouter>
11      );
12  }
13
14  export default App;

```

Abra a aplicação e será exibida a página a seguir.

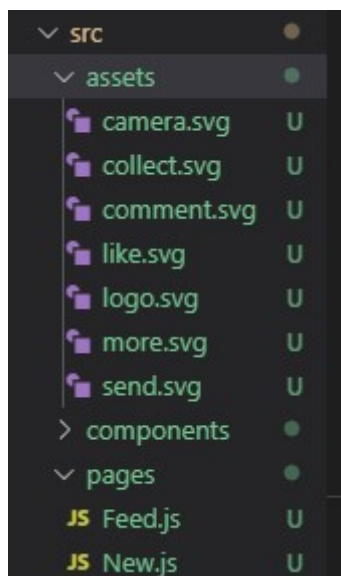


Se você acessar o endereço localhost:3000/new será exibida a página a seguir.



Criando a pasta de imagens

Dentro da pasta src crie a pasta assets e cole todas as imagens disponíveis para a aplicação. Os arquivos com a extensão svg, são um formato de vetor que o react consegue importar



Criando o componente Header

Este componente será exibido nas páginas Feed e New.

Crie uma pasta components e dentro dela o arquivo Header.js com o conteúdo abaixo.

```
JS Header.js x
1 import React from 'react'
2
3 import logo from '../assets/logo.svg'
4 import camera from '../assets/camera.svg'
5
6 export default function Header() {
7   return (
8     <header id='main-header'>
9       <div className='header-content'>
10         <img src={logo} alt='InstaFoa' />
11         <img src={camera} alt='Enviar publicação' />
12       </div>
13     </header>
14   )
15 }
```

Importando o Header no arquivo src/App.js

- Insira a linha 4 para importar o componente Header
- Insira a linha 10 para executar o componente Header

```
JS App.js x
frontend > src > JS App.js > ...
1 import React from 'react';
2 import { BrowserRouter } from 'react-router-dom'
3
4 import Header from './components/Header'
5 import Routes from './routes'
6
7 function App() {
8   return (
9     <BrowserRouter>
10       <Header />
11       <Routes />
12     </BrowserRouter>
13   );
14 }
15
16 export default App;
```

Abra o navegador e será exibida a página abaixo.

