Eder Rosati Ribeiro — 8122585,
Leonardo de Almeida Lima Zanguetin — 8531866,
Victor Luiz da Silva Mariano Pereira — 8602444

# Projeto 2
# *React* PetShop

Brasil

2018

Eder Rosati Ribeiro — 8122585,
Leonardo de Almeida Lima Zanguetin — 8531866,
Victor Luiz da Silva Mariano Pereira — 8602444

# Projeto 2
# *React* PetShop

Universidade de São Paulo – USP
Instituto de Ciências Matemáticas e de Computação – ICMC
Introdução ao Desenvolvimento Web – SCC0219

Brasil
2018

# 1 Introdução

Nesta segunda fase da aplicação PetShop, nos foi solicitado que continuássemos com o desenvolvimento de uma página *web* utilizando as linguagens HTML5, CSS3, as quais havíamos utilizado para desenvolver a primeira parte do projeto, e, as novidades, JavaScript e o uso de *frameworks*.

O objetivo deste trabalho ainda é o desenvolvimento de uma página no estilo *Single-Page Application* de um PetShop. Por termos feito já o *mockup* na primeira parte do trabalho e com a utilização dessa nova linguagem, acreditamos que será mais simples de fazer essa segunda parte.

O HTML5, como sabemos, cuida da parte estrutural do nosso *site*, o CSS3, pelo estilo da página. Já o JavaScript é responsável pelo comportamento da nossa página, ou seja, tudo que acontecer na página sem precisar atualizá-la, é parte do JavaScript.

Para facilitar o desenvolvimento nessa nova linguagem, escolhemos o React, uma biblioteca declarativa de JavaScript que serve para criar interfaces visuais. Com ela, podemos motrar mais dinamismo utilizado React, pois ele além de ser reativo, ou seja, ao mudar o estado de um componente o que ele representa muda também, podemos economizar tempo e minimizar em linhas de código, porque podemos usar componentes reutilizáveis.

# 2 Funcionamento

Como foi dito anteriormente, todo o *site* foi desenvolvido utilizando o React, uma biblioteca responsável apenas pelo desenvolvimento do *front-end* em *Single Page Application* (SPA). Tudo que julgamos conveniente à reutilização foi componentizado para reaproveitamento, tanto nos itens que precisavam ser reutilizáveis quanto nas partes que tinham papéis diferentes dentro do *site*.

Essa biblioteca JavaScript roda através do Node, portanto é necessário a instalação do Node e do NPM (do inglês, *Node Package Manager*) para iniciar um projeto. Com o NPM é possível baixar e instalar o *Create React App*, um pacote responsável por configurar várias tecnologias e trazer um projeto pronto para o desenvolvimento com todas as ferramentas iniciais mais utilizadas no React. Uma dessas tecnologias é o JSX, capaz de transformar os códigos JavaScript em códigos parecidos com HTML5.

Pode haver estranhamento com o código desenvolvido por aparentar utilizar HTML misturado com JavaScript, porém, é tudo JavaScript e o Babel nos auxilia nessa etapa. Todo o código pode ser realizado em ECMAScrip 6 pois é transpilado e convertido em ECMAScript 5 para funcionar nos navegadores atuais que ainda não o suportam.

O *site* todo consome uma API que hospedamos no GitHub servindo como interface entre nosso *front-end* e *back-end*. O projeto já está preparado para consumir esta API hospedada *online*, mas caso queira que o projeto rode totalmente *offline*, deixamos preparado para isso. Na seção 4.4 será abordada como podermemos rodar *offline* utilizando Json Server.

# 3 Estrutura do *site*

Como o nosso *site* é baseado praticamente em componentização, neste capítulo passaremos uma noção de como o estruturamos. No começo de tudo, temos a nossa aplicação, mais conhecida como APP, componente a qual consiste o nosso *site* inteiro e que será responsável por formar nosso único arquivo `.html`, chamado `index.html`.

## 3.1 APP

### 3.1.1 Topo

#### 3.1.1.1 Busca

#### 3.1.1.2 Login

### 3.1.2 Navbar

### 3.1.3 Conteúdo

### 3.1.4 Rodapé

# 4 Instruções

Neste capítulo vamos mostrar como utilizar a nossa aplicação, podendo rodar tanto *online* como *offline*. Os comandos estão listados em ordem, portanto a execução deles resulta no funcionamento do projeto. Na sequência, teremos a instalação das ferramentas necessárias no Linux e no Windows e posteriormente como será o procedimento para rodar a aplicação.

## 4.1 Linux

### 4.1.1 Instalação do Node

```
1 sudo apt-get install nodejs #Distros baseadas em Debian ou
2 sudo pacman -S nodejs       #Distros baseadas em Arch Linux
```

### 4.1.2 Instalação do NPM

```
1 sudo apt-get install npm #Distros baseadas em Debian ou
2 sudo pacman -S npm       #Distros baseadas em Arch Linux
```

### 4.1.3 Instalação do *Create React App*

```
1 npm install -g create-react-app
```

## 4.2 Windows

### 4.2.1 Instalação do Node

Pode ser baixado através do *site* oficial: https://nodejs.org/en/download/

### 4.2.2 Instalação do NPM

Como o NPM vem no pacode do Node e é instalado por padrão, não é preciso instalá-lo.

### 4.2.3 Instalação do *Create React App*

```
npm install -g create-react-app
```

Com estes comandos temos todas as ferramentas necesssárias para o funcionamento da nossa aplicação, tanto no Windows como no Linux.

## 4.3 Rodando a aplicação

Agora precisamos que entre na pasta do projeto pelo terminal.

```
cd ReactPetShop #Tanto Windows quanto Linux
```

Rode o projeto instalando suas dependências:

```
npm install && npm start      #Windows
sudo npm install && npm start #Linux
```

O projeto é aberto no navegador padrão assim que tudo estiver pronto, caso isso não ocorra, é possível acessá-lo através do *link*: http://localhost:3000/

## 4.4 Opção *offline*

Caso queira rodar a API *offline* siga estes comandos, mas nenhum deles é necessário para o funcionamento do projeto. Mantenha o terminal do processo passado e utilize outra aba/janela do terminal para realizar as próximas ações.

### 4.4.1 Instalação do Json Server

```
npm install -g json-server      #Windows
sudo npm install -g json-server #Linux
```

Vá até a pasta do projeto.

```
cd ReactPetShop #Tanto Windows quanto Linux
```

### 4.4.2 Rodando a API *offline*

```
json-server --watch --port 3001 db.json #Tanto Windows quanto Linux
```

Na pasta do projeto, abra o arquivo `src > service > http.js`. Nesse arquivo existem duas linhas de código:

```
baseURL: 'https://my-json-server.typicode.com/ederrr/ReactPetShop/'
//baseURL: 'http://localhost:3001'
```

Comente a primeira linha e descomente a segunda linha, ficando da seguinte forma:

```
//baseURL: 'https://my-json-server.typicode.com/ederrr/ReactPetShop/'
baseURL: 'http://localhost:3001'
```

Pronto, seu servidor *offline* está rodando a API e seu projeto está consumindo-a.