CENTRO UNIVERSITÁRIO DO PARÁ - CESUPA
ESCOLA DE NEGÓCIOS, TECNOLOGIA E INOVAÇÃO - ARGO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ANTONIO EDUARDO DA SILVA NEVES JUNIOR
ARTHUR VICTOR FERREIRA DAMOUS
FRANKLIN GONÇALVES SOBRINHO
VICTOR HUGO SEREJO FERREIRA

UM ESTUDO SOBRE A BIBLIOTECA REACT JS E SEUS IMPACTOS NO AMBIENTE DE DESENVOLVIMENTO WEB

BELÉM
2019

ANTONIO EDUARDO DA SILVA NEVES JUNIOR
ARTHUR VICTOR FERREIRA DAMOUS
FRANKLIN GONÇALVES SOBRINHO
VICTOR HUGO SEREJO FERREIRA

UM ESTUDO SOBRE A BIBLIOTECA REACT JS E SEUS IMPACTOS NO AMBIENTE DE DESENVOLVIMENTO WEB

Trabalho de conclusão de curso apresentado à Escola de Negócios, Tecnologia e Inovação do Centro Universitário do Estado do Pará como requisito para obtenção do título de Bacharel em Ciência da Computação na modalidade MONOGRAFIA.

Orientador: MSc. Theo Carlos Flexa Ribeiro Pires.

BELÉM
2019

Dados Internacionais de Catalogação-na-publicação (CIP)
Biblioteca do Cesupa, Belém – PA

Neves Junior, Antônio Eduardo da Silva.

Um estudo sobre a biblioteca React JS e seus impactos no ambiente de desenvolvimento web / Antonio Eduardo da Silva Neves Junior, Arthur Victor Ferreira Damous, Franklin Gonçalves Sobrinho, Victor Hugo Serejo Ferreira; orientador Theo Carlos Flexa Ribeiro Pires. – 2019.

Trabalho de Conclusão de Curso (Graduação) – Centro Universitário do Estado do Pará, Ciência da Computação, Belém, 2019.

1. Software – Desenvolvimento. 2. Framework (Programa de computador). I. Damous, Arthur Victor Ferreira. II. Sobrinho, Franklin Gonçalves. III. Ferreira, Victor Hugo Serejo. IV. Pires, Theo Carlos Flexa Ribeiro, *orient*. V. Título.

CDD 23ª ed. 005.1

ANTONIO EDUARDO DA SILVA NEVES JUNIOR
ARTHUR VICTOR FERREIRA DAMOUS
FRANKLIN GONÇALVES SOBRINHO
VICTOR HUGO SEREJO FERREIRA

UM ESTUDO SOBRE A BIBLIOTECA REACT JS E SEUS IMPACTOS NO AMBIENTE DE DESENVOLVIMENTO WEB

Trabalho de conclusão de curso apresentado à Escola de Negócios, Tecnologia e Inovação do Centro Universitário do Estado do Pará como requisito para obtenção do título de Bacharel em Ciência da Computação na modalidade MONOGRAFIA.

Data da aprovação: / /
Nota final aluno I:_____
Nota final aluno II:_____
Nota final aluno III:_____
Nota final aluno IV:_____

Banca examinadora

_____

Prof. MSc. Theo Carlos Flexa Ribeiro Pires
Orientador e Presidente da banca

_____

Prof Msc. Ricardo Melo Casseb Do Carmo
Examinador

AGRADECIMENTOS

Agradeço, primeiramente, à Deus por sempre me guiar pelos melhores caminhos em meio aos obstáculos da vida e por sempre estar do meu lado me apoiando nas horas mais difíceis.

Agradeço ao meus pais, Eduardo Neves e Soraia Cristina por todo amor, apoio e dedicação para que eu tivesse a melhor educação possível ao longo da vida.

Agradeço a minha noiva Marcelly Moraes, que sempre esteve ao meu lado nos últimos 9 anos e sempre me apoiou em toda a trajetória deste curso.

Agradeço ao meu irmão, Evandro e minhas irmãs Debora e Evelyn, por fazerem parte da minha vida e me motivarem a ser um irmão e pessoa cada vez melhor.

Agradeço aos meus amigos, em especial aos colegas de turma, com os quais formamos verdadeiras equipes e amizades para toda a vida.

Agradeço aos nossos professores orientadores Theo Pires e Odlaniger Lourenço por colaborarem para que esse projeto de TC ocorresse da melhor maneira.

Agradeço aos meus professores do CESUPA por todos os ensinamentos adquiridos na graduação.

A todos, o meu muito obrigado.

Antonio Eduardo da Silva Neves Junior.

AGRADECIMENTOS

Agradeço aos meus pais Edna do Socorro Damous e Antonio Jorge Damous, por todo o apoio, suporte e amor oferecidos a mim. Nos momentos mais difíceis foi a eles quem recorri. Devo muito a eles por todas as minhas conquistas.

Também agradeço ao meu irmão Jorge Antonio, por tudo que passamos juntos e conquistamos. Acredito que ainda vamos compartilhar muitas experiências boas em nossas vidas.

À minha família, minhas tias, tios, primos e avós, por todo o carinho oferecido, por todo o apoio. Me conforta muito saber que sempre posso contar com todos.

Agradeço ao CESUPA, uma instituição que ganhou muito o apreço e respeito, pois é uma instituição séria e de renome, capaz de formar profissionais excelentes e oferecendo um dos melhores suportes da região.

Agradeço também aos meus colegas de turma, por todos os momentos que vivemos juntos, compartilhando histórias, ideias e soluções. Sempre em parceria ajudando mutuamente cada um.

Também agradeço a todos os meus amigos que fizeram parte desta trajetória, como Victor Mendes, Victória Mendes, Otávio Vieira, Jadiel Neto, Maliria Gisele, Alder Furtado, Eduardo Neves, Franklin Sobrinho, Victor Serejo, Luan Ferreira, Filipe Castro, Yussef Toutenge, Lucas Miguins, Lucas Davi, Lucas Uchoa e Renan Mello.

Agradeço aos nossos professores orientadores Theo Pires e Odlaniger Lourenço por colaborarem para que esse projeto de TC ocorresse da melhor maneira.

Agradeço à todo o corpo docente do CESUPA, principalmente aqueles envolvidos diretamente com o curso de Ciência da Computação. A todos, o meu muito obrigado por todos os ensinamentos e aprendizados que recebi dos mesmos.

No mais, só tenho a agradecer a Deus. Por ter me proporcionado todos esses momentos incríveis em minha vida. Por cada dia ter a oportunidade de olhar para o futuro e sonhar com o melhor.

Obrigado.

Arthur Victor Ferreira Damous.

AGRADECIMENTOS

Agradeço aos meus pais José de Sousa Sobrinho e Marivalda de Aragão Gonçalves e meus familiares por todo o apoio, incentivo e amor oferecidos ao longo dos anos.

Agradeço a todos os meus amigos por todo o suporte oferecido durante minha trajetória.

Agradeço aos nossos professores orientadores Theo Pires e Odlaniger Lourenço por colaborarem para que esse projeto de TC ocorresse da melhor maneira.

Agradeço ao CESUPA por todo conhecimento oferecido ao longo dos 4 anos de graduação.

A todos, o meu muito obrigado.

Franklin Gonçalves Sobrinho.

AGRADECIMENTOS

Agradeço à Deus por sempre me abençoar, guiar e dar forças nas escolhas da minha vida, pois, sem ele, este momento não seria possível.

Agradeço a minha família, em especial meu pai Carlos Emílio, minha mãe Mônica Roseane e minha irmã Camille Serejo, por sempre me apoiarem e incentivarem a sempre buscar meus objetivos e a nunca desistir nos momentos difíceis,

Agradeço aos meus colegas de turma, por sempre me ajudarem quando precisei e por todos os conhecimentos compartilhados durantes esses 4 anos de curso que, além disso, fizemos grandes amizades que levarei pro resto da vida.

Agradeço á todos meus amigos particulares que, também, sempre me incentivaram e deram forças durante minha trajetória de vida, compartilhando momentos bons e difíceis comigo.

Agradeço aos nossos professores orientadores Theo Pires e Odlaniger Lourenço por colaborarem para que esse projeto de TC ocorresse da melhor maneira.

Agradeço ao corpo docente do CESUPA, por todos esses 4 anos de ensinamentos que me tornaram um profissional muito bem capacitado, e que levarei pro resto da vida profissional.

A todos, o meu muito obrigado.

“Dificuldades preparam pessoas comuns para destinos extraordinários.”
C. S. Lewis

Victor Hugo Serejo Ferreira.

RESUMO

Várias tecnologias de desenvolvimento de aplicativos surgem com uma velocidade muito grande atualmente, porém, quais dessas tecnologias são boas o suficiente para se investir tempo e esforço, ainda é um grande questionamento da comunidade de programadores. O avanço das tecnologias de desenvolvimento traz novas linguagens de programação, novos frameworks, novas bibliotecas e novos padrões, então os programadores devem manter a tecnologia que por anos tem dado certo, ou devem migrar e tentar otimizar seus serviços? Desde 2016, a biblioteca React traz a premissa de otimizar a construção de interfaces de usuário e vem ganhando novos adeptos, por este motivo, faz-se necessário o estudo dessa biblioteca de código JavaScript, para que os programadores possam analisar melhor as opções que tem hoje em dia para migrar para uma tecnologia nova ou para que um aluno de programação possa iniciar nos estudos e ter boas expectativas de sucesso no mercado profissional.

Palavras-chave: JavaScript. React. Desenvolvimento de software.

ABSTRACT

Many application development technologies are emerging at a very high speed today, but which of these technologies is good enough to invest time and effort is still a big question for the developer community. The advancement of development technologies brings new programming languages, new frameworks, new libraries and new standards, so should programmers maintain the technology that has worked for years, or should they migrate and try to optimize their services? Since 2016, the React library has been premised on optimizing the construction of user interfaces and has been gaining new followers, so it is necessary to study this JavaScript code library, so that programmers can better analyze the options they have. nowadays to migrate to new technology or so that a programming student can start their studies and have good expectations of success in the professional market.

Keywords: JavaScript; React, Software development.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

CSS	Cascading Style Sheets
DOM	Document Object Model
ES	ECMAScript
ES6	ECMAScript 6
ES2015	ECMAScript 2015
JSON	Javascript Object Notation
JSX	JavaScript XML eXtension
HTML	HyperText Markup Language
PHP	Pre processed Home Page
SEO	Search Engine Optimization
VPS	Virtual Private Server
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Desde o início do mercado de desenvolvimento de software Web ocorreram diversas mudanças nos paradigmas das tecnologias que o englobam. Segundo uma pesquisa realizada com diversos executivos à frente de grandes cargos em empresas de computação, as mudanças nas tecnologias estão demorando cada vez menos tempo para acontecer (SMITH, 2016). A evolução da área da tecnologia de desenvolvimento Web permitiu mais do que apenas acesso à informação. Pessoas e empresas utilizam estes tipos de sistemas todo dia como um recurso para fazer negócios, seja como um blog ou uma plataforma completa de serviços. Este avanço tecnológico possibilitou o surgimento de novas modalidades de desenvolvimento de software (SILVA, 2010). Neste contexto, este trabalho propõe-se a estudar as principais características de um ambiente de tecnologias com potencial crescimento e relevância de mercado dentro desta categoria e seus impactos no mercado de desenvolvimento de software.

1.2 SITUAÇÃO PROBLEMA

Hoje, no processo de produção de sistemas é necessário muito esforço integrado de profissionais das áreas de desenvolvimento. À medida que aumentam a complexidade e a sofisticação dos novos sistemas, uma melhor abordagem de desenvolvimento, qualidade e segurança nos serviços são exigidos, e uma melhor metodologia contribui para um desenvolvimento harmônico, estruturado e bem integrado (PRESSMAN, 2006).

Diante das necessidades de softwares cada vez mais funcionais e sofisticados, os desenvolvedores e estudantes de programação precisam encontrar soluções para superar os obstáculos e atender a demanda de um público mais exigente, incluindo as grandes corporações. Contudo, a mudança de linguagens, estruturas e ambiente de desenvolvimento como um todo, nos tempos atuais ainda é um tabu, tornando dificultosa a tomada de decisão por uma parcela de profissionais da área (JUNIOR, 2017).

1.3 OBJETIVOS DO ESTUDO

1.3.1 Objetivo geral

Estudar as características da biblioteca React e as ferramentas que são atreladas ao seu ecossistema, através de pesquisa e utilizando o desenvolvimento de um sistema com a biblioteca

para auxiliar na escolha dos interessados em migrar para novas tecnologias como o React e estudar sobre desenvolvimento de softwares aplicativos.

1.3.2 Objetivos específicos

- Pesquisar sobre a linguagem JavaScript e a biblioteca React.
- Pesquisar tecnologias que utilizam o JavaScript e servem de suporta ao React.
- Desenvolver um protótipo de aplicativo com React.
- Analisar situação e expectativa de mercado para as tecnologias da pesquisa.

1.4 JUSTIFICATIVA

O JavaScript é uma linguagem de programação que já está presente no mercado desde 1995 e teve um crescimento considerável nos últimos anos na utilização de usuários programadores (STACKOVERFLOW, 2018) e pode se tornar uma excelente escolha para a migração de programadores já estabelecidos e para adição de novos conhecimentos.

Figura 1. Pesquisa realizada em 2018 pelo Site StackOverflow

,

Fonte: STACKOVERFLOW, 2018.

Hoje o JavaScript possui uma biblioteca de códigos desenvolvida pelo Facebook, chamada React, que trouxe consigo a premissa de tornar o desenvolvimento de aplicativos Web mais fácil (REACT, 2019). Também há uma gama de bibliotecas disponíveis no maior repositório de bibliotecas do mundo atualmente, o NPM (*Node Package Manager*) (BRASIL, 2019). Segundo uma pesquisa realizada pelo website *SimilarTech*, que é referência nos tipos de pesquisa que realiza, a biblioteca React é utilizada em 865,132 *websites* atualmente (SIMILARTECH, 2019). Devido à crescente demanda por aplicações cada vez mais seguras e melhores, faz-se necessário o estudo sobre essas novas tecnologias e suas qualidades para os novos programadores e grandes empresas que desejam investir tempo e dinheiro no aprendizado da mesma.

1.5 METODOLOGIA DA PESQUISA

A metodologia de pesquisa utilizada é a exploratória, fazendo-se uso de artigos científicos publicados em grandes veículos, artigos da web e de instituições confiáveis para embasar o conhecimento sobre as tecnologias.

Será estudado a biblioteca React, através de pesquisa bibliográfica e utilizando o desenvolvimento de um sistema construído com a ferramenta, produzido para auxiliar no entendimento do funcionamento das tecnologias.

Por fim, será feita uma análise de mercado atual através de pesquisas e indicadores de uso feitas por sites renomados, buscando medir o quanto a tecnologia tem crescido e abordar um possível futuro para a tecnologia e seus usuários. O enfoque é apresentar o que está sendo desenvolvido como novas tecnologias em arquiteturas de software ressaltando as principais características destas para que se possa compreender a significância dela no mercado atualmente e um prospecção futura.

1.6 ESTRUTURA DO TRABALHO

Além deste capítulo introdutório, que trata sobre o contexto do estudo, os seus objetivos e a metodologia utilizada, a estrutura dos demais capítulos desta dissertação é descrita a seguir.

O Capítulo 2 - apresenta o estudo das tecnologias ligadas ao JavaScript e a biblioteca React, capítulo que explica a linguagem JavaScript e sua popularização no mercado de desenvolvedores a partir do ano de 2015, O Node js, ambientes de desenvolvimento de código JavaScript para plataformas *desktop*, na arquitetura de *software* Flux e em descrever suas principais características e funcionalidades e as bibliotecas React e Redux.

O Capítulo 3 - apresenta um estudo de caso, com a construção de uma aplicação do mundo real, com a utilização das ferramentas estudadas na pesquisa do capítulo 2, como o Node js, React e Redux.

O Capítulo 4 - apresenta o mercado e a situação atual e prospecções futuras das tecnologias antes mencionadas.

E finalmente, o Capítulo 5 apresenta as considerações finais desta monografia.

2 ESTUDO DAS TECNOLOGIAS

2.1 O JAVASCRIPT

A linguagem de programação JavaScript foi criado na década de 90 por Brendan Eich. Quando foi lançado em 1995, seu objetivo era validar entradas de dados na parte do cliente em sistemas servidor/cliente. Antes dela era necessária uma chamada ao servidor para determinar se um campo de entrada havia sido deixado em branco ou se um valor inserido era inválido. Com isso, o navegador Netscape procurou alterar isso, inserindo então, o JavaScript. A capacidade de lidar com algumas validações básicas no cliente foi um recurso singular para a época. Deixando também as validações mais rápidas, uma vez que não era mais preciso enviar uma requisição para o lado do servidor para validar os dados (ZAKAS, 2012).

O JavaScript passou por diversas mudanças desde sua criação. No início, ele tornava a adição de elementos interativos a páginas da Web muito mais simples, agora, com o novas tecnologias atreladas e linguagem como o Node.js (Ambiente de desenvolvimento para desktop), o JavaScript se tornou uma linguagem usada para criar aplicativos de pilha completo, abrangendo *frontend*, *backend*, *desktop*, web e *mobile*. Hoje é a linguagem de programação da Web mais relevante no mercado de desenvolvimento de *software* web. A esmagadora maioria dos sites modernos faz uso do JavaScript, e todos os navegadores modernos em desktops, consoles de jogos, tablets e smartphones incluem intérpretes JavaScript, tornando-o a linguagem de programação mais onipresente da história. O JavaScript faz parte da tríade de tecnologias que todos os desenvolvedores da Web devem aprender: HTML para especificar o conteúdo de páginas da Web, CSS para especificar a apresentação de páginas da Web e JavaScript para especificar o comportamento das páginas da Web (FLANAGAN, 2011).

2.1.1 ECMAScript

A linguagem JavaScript teve seu primeiro protótipo criado em apenas 10 dias, sendo batizada com o nome Mocha e quando teve seu primeiro lançamento oficial em setembro de 1995, juntamente com a versão 2.0 do navegador Netscape, foi chamada de LiveScript e em dezembro do mesmo ano seu nome foi alterado para JavaScript. Antes que o JavaScript se tornasse popular, para que a linguagem evoluísse obedecendo a determinados padrões e normativas, em 1996, os criadores do JavaScript se associaram ao ECMA (*European Computer Manufactures Association*). Como o nome JavaScript já havia sido patenteado pela Sun

Microsystems (atualmente Oracle), optou-se por se definir um novo nome à linguagem utilizando a junção das palavras ECMA e JavaScript, surgindo então o ECMAScript. Logo em seguida a essa junção com a ECMA foi criado um comitê de gestão das versões do ECMAScript, o ECMA-262, grupo criado na ECMA para a padronização do JavaScript e que conta com participação de grandes empresas de tecnologia como Microsoft, Google, dentre outras (MALAVASI, 2017).

2.1.2 O ES6/ECMASCRIPT 2015

O ECMAScript (ES) é a especificação da linguagem de script que o JavaScript implementa, ou seja, é a descrição formal e estruturada de uma linguagem de script, sendo padronizada pela Ecma. No dia 17 de junho de 2015, foi definida a sexta edição da especificação, a ES6 (também chamada de ECMAScript 2015). Diferentemente das edições anteriores, o ES6 trouxe a maior mudança para a linguagem JavaScript desde a sua criação. O principal objetivo da nova versão, especificamente, foi tornar a linguagem mais flexível, enxuta e fácil de se aprender e trabalhar, tornando-a mais próxima a outras linguagens orientadas a objeto, como Java e Python (PINHO, 2017).

Na Figura 2, um exemplo de código escrito em JavaScript, com uma sintaxe de até meados de 2014.

Figura 2. Código escrito em ES5

```
1
2   function soma(n1, n2){
3       return n1 + n2;
4   }
5
```

Fonte: Autores

Essa sintaxe do JavaScript foi substituída em 2015, pelo ES6, e trouxe mais maleabilidade para os programadores em escrever a linguagem. Dentre as principais mudanças, temos:

- Criação de novos tipos de dados (Map, WeakMap, Set, WeakSet);
- Novas maneiras de iterar objetos e coleções;
- Declaração de variáveis com let e const;
- Modularização e estrutura de classes;

- Geradores e símbolos;
- Operadores rest e spread.

Na Figura 3 vemos um exemplo da sintaxe ES6 em comparação com a ES5:

Figura 3. Sintaxe da ES6

```
1
2   soma = (n1, n2) ⇒ n1 + n2;
3
```

Fonte: Autores.

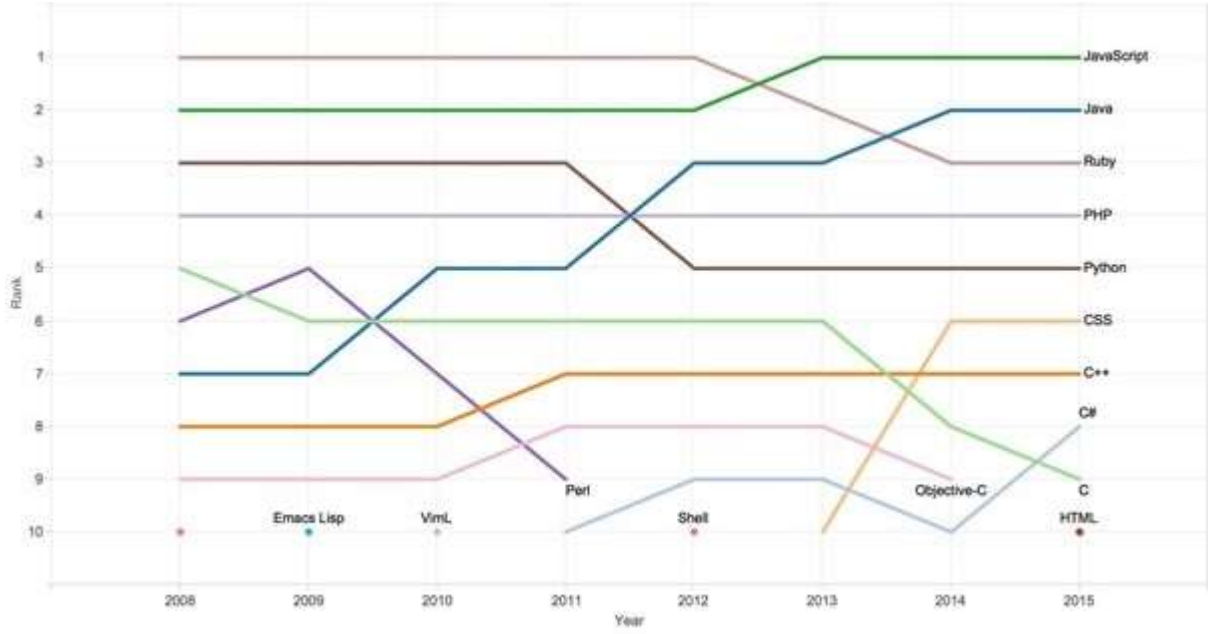
Note que a sintaxe ES6 no mesmo exemplo demonstrado de “soma”, o número de linhas é consideravelmente menor. O que torna o código mais simples, enxuto e agradável a vista do programador.

Ainda segundo PINHO, para que a linguagem não parasse novamente no tempo, a ECMA se comprometeu a lançar uma nova versão da especificação todos os anos. No ano de 2017, na sétima edição da especificação trouxe ainda mais melhorias, tais como:

- Novas operações em Array;
- Operador de exponenciação (**);
- Decoradores;
- Propriedades estáticas de classe;
- Async-await.

Foi graças e essas atualizações de estrutura da linguagem e também graças ao grande “BOOM” das tecnologias Node.Js e npm (Serão vistas mais para frente), que o JavaScript começou a sua popularização em 2015 segundo pesquisa feita pelo GITHUB em 2015, vista na Figura 4:

Figura 4. As linguagens mais populares de todos os tempos.



Fonte: GITHUB,2015.

Esse interesse na linguagem se refletiu na criação de uma imensa diversidade de bibliotecas e *frameworks*. Alguns exemplos mais famosos que têm tido grande aceitação na comunidade e no mercado:

- React: Biblioteca criada pela equipe do Facebook para criação de componentes;
- React Native: *Framework* para criação de aplicativos *mobile* usando o React;
- Angular 2: *Framework* MVC mantido pelo Google;
- js: Biblioteca para criação de componentes;
- Meteor: *Framework* para criação de aplicações web *singlepage*;
- Electron: *Framework* para criação de aplicações *desktop* multiplataforma.

Até a data atual da publicação desta monografia a maioria dos browsers ainda não dá suporte para a versão do ES6 do JavaScript, tão pouco, para versões posteriores, assim, para que o programador JavaScript possa escrever código ES6, é preciso de um tradutor (transpiler).

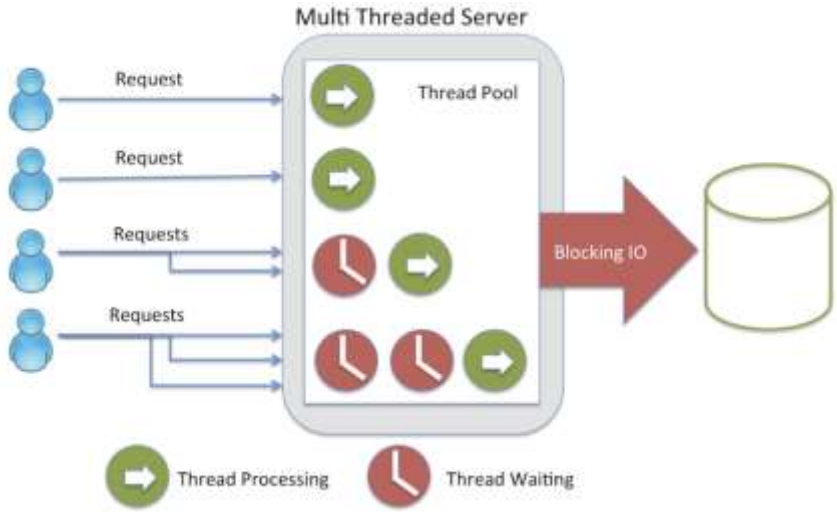
O transpiler mais famoso atualmente é o BABEL JavaScript, que transforma seu código ES6 em ES5 (versão antiga), que é suportada pela maioria dos navegadores (LIMA, 2019).

2.2 NODE JS

O Node.js foi desenvolvido por Ryan Dahl em 2009. Neste mesmo ano o Google lançou o V8, um interpretador de JavaScript, também chamado de Máquina Virtual JavaScript, para suprir uma necessidade da indústria e conseguir rodar uma máquina virtual de alta performance e que interpretasse códigos JavaScript para programação, o que não era feito até então, o JavaScript servia apenas como um linguagem de suporte ao ambiente de desenvolvimento web, facilitando no comportamento dos elementos HTML. O Node.js foi desenvolvido em cima do V8, essencialmente como uma forma de rodar programas JavaScript fora do contexto de um browser. O Node.js é usado para a criação de aplicativos JavaScript altamente escalonáveis. Empresas respeitadas como a Dow Jones, a LinkedIn e a Walmart estão entre as muitas organizações que têm visto o potencial do Node e o tem adotado em seus negócios (IHRIG, 2014).

A principal característica que diferencia o Node.JS de outras tecnologias, como PHP, Java, C#, é o fato de sua execução ser single-thread. Apenas uma thread é responsável por executar o código JavaScript da aplicação. Em um servidor web utilizando linguagens tradicionais, para cada requisição recebida é criada uma nova thread para tratá-la. A cada requisição, serão demandados recursos computacionais (memória RAM, por exemplo) para a criação dessa nova thread. Uma vez que esses recursos são limitados, as threads não serão criadas infinitamente, e quando esse limite for atingido, as novas requisições terão que esperar a liberação desses recursos alocados para serem tratadas (LENON, 2018), como podemos ver na Figura 5:

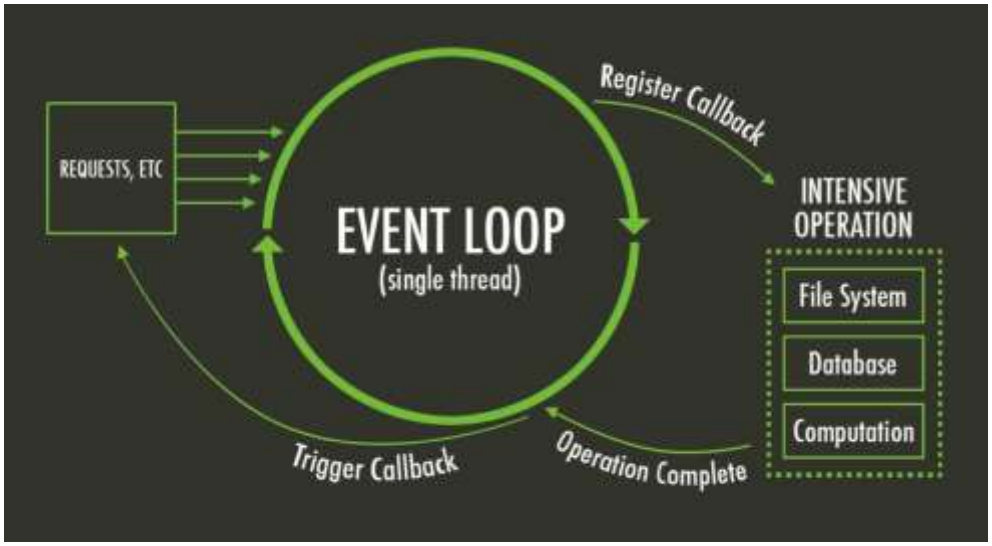
Figura 5. Modelo de multi-threads em servidores tradicionais.



Fonte: LENON, 2018.

Com o Node.js, apenas uma thread é responsável por tratar as requisições. Esta é chamada de Event Loop, e leva esse nome pois cada requisição é tratada como um evento. O Event Loop fica em execução esperando novos eventos para tratar, e para cada requisição, um novo evento é criado e apesar de ser single-thread, é possível tratar requisições concorrentes em um servidor Node.js. Enquanto o servidor tradicional utiliza o sistema multi-thread para tratar requisições concorrentes, o Node.js consegue o mesmo efeito através de chamadas de E/S (entrada e saída) não-bloqueantes. Isso significa que as operações E/S (acesso a banco de dados e leitura de arquivos do sistema) são assíncronas e não ocorre o bloqueio da thread. Diferentemente dos servidores tradicionais, a thread não fica esperando que essas operações sejam concluídas para continuar sua execução (LENON, 2018). A figura 6 representa a diferença de funcionamento de um servidor web tradicional e um Node.JS:

Figura 6. Funcionamento de um servidor tradicional e Node js



Fonte: RAGAEY, 2017.

Apesar do Node.js ser single-threaded, sua arquitetura possibilita que um número maior de requisições concorrentes seja tratado em comparação com o modelo tradicional, que é limitado devido ao alto consumo computacional pela criação e manutenção de threads a cada requisição.

2.3 O NPM

O NPM (Node Package Manager) é um pacote dentro no Node.js que serve para gerenciar as dependências das aplicações. O NPM consiste muito mais em um kit de ferramentas para desenvolvimento do que somente um instalador de pacotes. Através do NPM

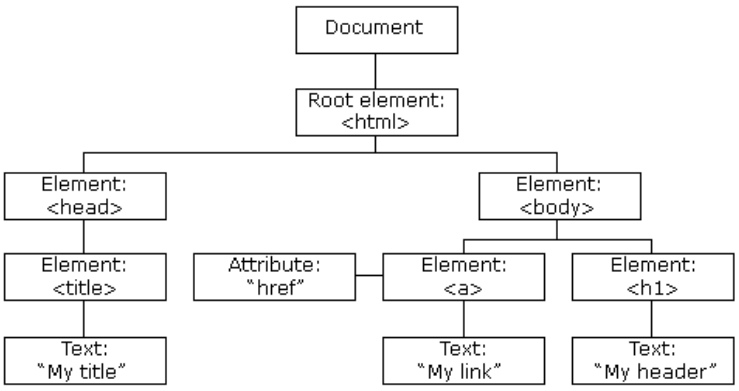
config é possível realizar uma série de tarefas, como efetuar o login/logout do npm.org para controlar pacotes próprios por exemplo (NPM, 2019).

O NPM é o maior ecossistema de bibliotecas open source do mundo (BRASIL, 2019), daí mais um motivo relevantes para se investir em tecnologias que têm como base a linguagem JavaScript.

2.4 DOM (DOCUMENT OBJECT MODEL)

O DOM (Document Object Model) é uma convenção multiplataforma e independente de linguagem para representar e interagir com objetos em HTML, XHTML, e documentos XML (MDN, 2019). Os nós de todos os documentos são organizados em uma estrutura de árvore, chamada de árvore DOM, como representada na Figura 7.

Figura 7. Árvore DOM



Fonte: W3SCHOOLS, 2019.

Objetos na árvore DOM podem ser endereçados e manipulados utilizando métodos nos objetos. A interface pública de um DOM é especificada na sua API (Application Programming Interface) (SILVA, 2010).

A história do DOM está intimamente ligada com o início das linguagens de script. Após o lançamento do ECMAScript, a W3C (World Wide Web Consortium) começou a trabalhar na padronização do DOM. O padrão inicial, conhecido como "DOM Nível 1", foi recomendado no final de 1998 (W3C, 1998). Na mesma época, o Internet Explorer 5.0 foi entregue com suporte limitado para DOM Nível 1. O DOM Nível 1 proveu desde um modelo completo para um documento HTML como para um XML, incluindo meios para modificar qualquer parte

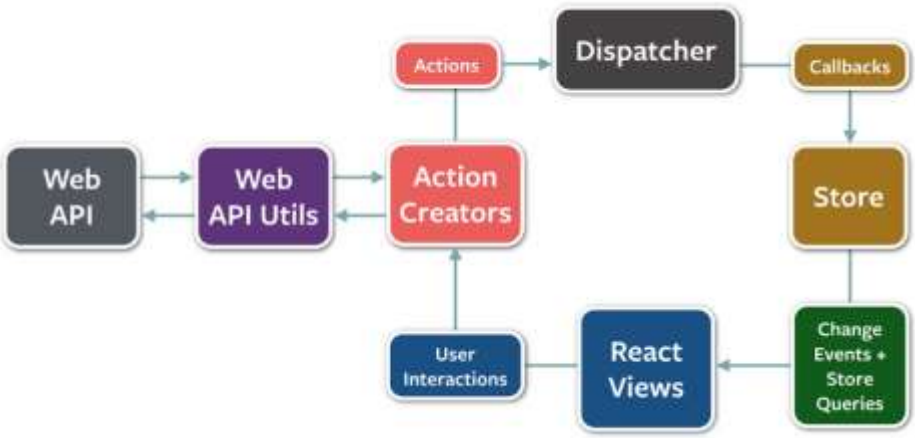
deste. Navegadores não conformes, tais como Internet Explorer e Netscape 4.x, ainda foram amplamente utilizados até o ano 2000. DOM Nível 2, publicado no final de 2000 (W3C, 2000), introduziu a função "*getElementById*", bem como um modelo de evento e suporte para *namespaces* XML e CSS. O DOM Nível 3, a versão atual em 2019 da especificação do DOM (W3C, 2004), adicionou suporte para XPath e manipulação de eventos de teclado, bem como uma interface para a serialização de documentos XML (SILVA, 2010).

Entender o DOM é fundamental para se entender a estrutura básica e como funciona a arquitetura Flux, Uma arquitetura de software nova feita pelo Facebook, tema do próximo capítulo. Uma vez que os componentes de uma aplicação JavaScript são baseadas em elementos do DOM.

2.5 O FLUX

Flux é uma arquitetura de software desenvolvida pelo Facebook em 2013, sendo aberto para a utilização pública (open source) somente em 2014. É uma arquitetura de software feita para lado do cliente em aplicações web. Aplicações que usam essa arquitetura são compostas por três partes principais Figura 8. O *dispatcher*, *stores*, e *views*. Quando há interação com uma *view*, ela propaga uma ação para um dispatcher central, e então para os *stores*, que armazenam os dados da aplicação e a lógica de negócios, por fim a interface é atualizada, apenas onde for necessário (FLUX, 2019).

Figura 8. Estrutura do flux



Fonte: FLUX, 2015

2.5.1 O Dispatcher

O Dispatcher é como uma central da aplicação. Uma central responsável por registrar callbacks e emitir eventos. Por ser uma central, fica claro que deve haver apenas um único Dispatcher para toda sua aplicação (FLUX, 2019).

2.5.2 View

A camada View, na arquitetura Flux, nada mais é do que um componente da aplicação do lado do cliente. Nela, nosso usuário faz uma ação e isso ativa o Dispatcher, que atualiza a store, que atualiza a View (FLUX, 2019).

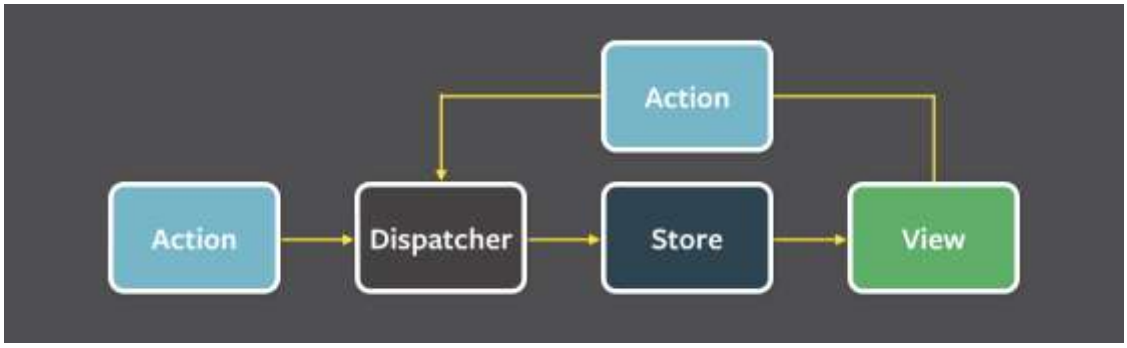
2.5.3 Store

A camada da store serve para guardar os dados da aplicação, que são gerenciados pelo Dispatcher. Os dados guardados na store são exibidos na camada View, como se fossem pegos de um banco de dados e quando os dados são atualizados a própria store se encarrega de carregar a View (FLUX, 2019).

2.5.4 Actions

No contexto do Flux, uma action é uma ação que dispara um dispatcher, que altera os dados numa store, que atualiza a View. uma ação pode ser um evento disparado pelo usuário no lado do cliente como um click de mouse, um envio de formulário, um click de botão, qualquer coisa feita pelo usuário dentro da interface (FLUX, 2019). Na Figura 9 podemos ver melhor o ciclo de vida da arquitetura Flux:

Figura 9. Ciclo de vida do flux



Fonte: FACEBOOK, 2015.

2.6 REACT

O React é uma biblioteca JavaScript para construção de interfaces gráficas associadas, desenvolvida pelo Facebook (REACT 2019). Um dos paradigmas adotados pelo framework é a programação declarativa, os desenvolvedores descrevem os elementos a serem exibidos em uma página, através de linguagens de marcação. Para criar os nodos do DOM da estrutura do site web, o projeto é organizado em componentes, que possuem metadados sobre o tipo a ser mostrado, e para detectar aquilo que precisa ser atualizado. Outra característica da biblioteca é combinar o HTML (*Hyper Text Markup Language*) e JavaScript na mesma página, que ajuda em centralizar a lógica da camada de interação com usuário (BERTOLI, 2017).

O *framework* React foi lançado como um projeto de código aberto em 2013 (WALKE; OCCHINO, 2013). Anteriormente a este fato, desenvolvedores do Facebook, estavam tendo dificuldade em trabalhar no Facebook Ads, pois, era difícil manter a interface em sincronia, com as regras de negócio e dados da aplicação. Ainda que, algumas ferramentas tenham sido criadas, como Bolt, que era uma ferramenta de monitoramento do DOM, os programadores tinham dificuldade em lidar com a complexidade das atualizações. Que, enfim, levou ao desenvolvimento do React (STAFF, 2016).

Para resolver o problema de manter o *frontend* web atualizado, em relação aos dados provenientes de um servidor, de forma evitar a complexidade da aplicação inviabilize o desenvolvimento no futuro, o React utilizou componentes combináveis e reutilizáveis. Estes elementos são independentes e de domínio específico, e possuem a representação visual e uma lógica (MARDAN, 2017).

Figura 10. Exemplo de componente react

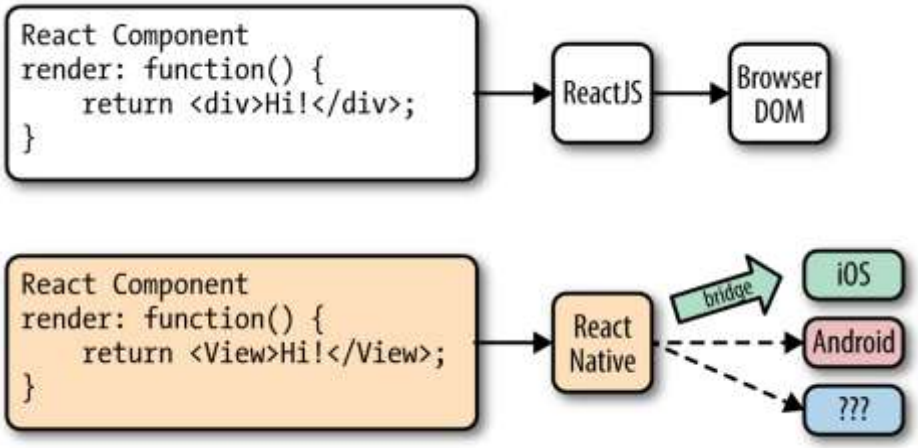
```
3   class Timer extends React.Component {
4
5       constructor(props){
6           super(props);
7
8           this.state = { seconds: 0 };
9       }
10
11      tick() {
12          this.setState(state ⇒ ({
13              seconds: state.seconds + 1
14          }));
15      }
16
17      componentDidMount(){
18          this.interval = setInterval(() ⇒ this.tick(), 1000);
19      }
20
21      componentWillUnmount(){
22          clearInterval(this.interval);
23      }
24
25      render(){
26          return (
27              <div>
28                  Seconds: {this.state.seconds}
29              </div>
30          );
31      }
32  }
33
34  ReactDOM.render(
35      <Timer />,
36      document.getElementById('timer-example')
37  );
```

Fonte: REACT, 2019.

O formato padrão para definir o HTML da página web no React, é através da utilização do JSX na função render dos componentes (Ver Figura 11). A extensão de XML para ECMAScript, permite adicionar uma linguagem de marcação diretamente no código. A especificação não é reconhecida, e nem tem intuito, diretamente pelos navegadores, sendo necessário um pré processador para gerar JavaScript válido (MARDAN, 2017).

O pré-processador padrão do React é o Babel (BABEL, 2019), que é um compilador JavaScript, como foco em converter código compatível com versões superiores ao ECMAScript 2015, em uma versão que possa ser executado em navegadores mais antigos. Além disso, possui suporte, através de uma extensão, ao React, que permite o uso do JSX.

Figura 11. Exemplo de componente react



Fonte: EISENMAN, 2018.

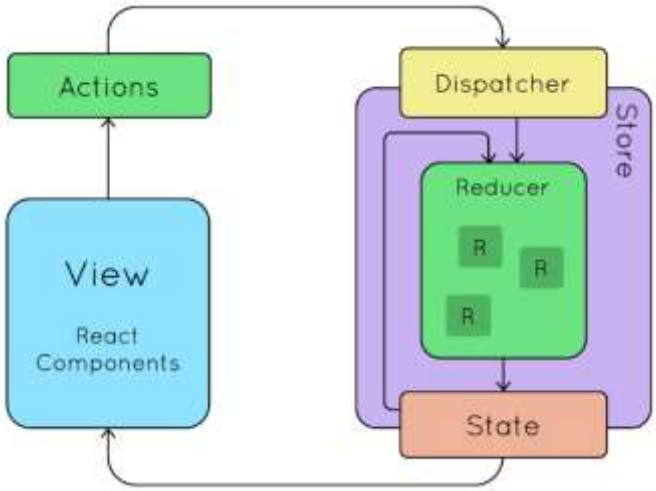
Um recurso, que foi popularizado pelo React, é o Virtual DOM, que é uma estrutura de dados, e armazena uma versão com todos os elementos de uma interface. Desta forma, é possível fazer comparações quando um valor é alterado, para determinar quais áreas precisam ser redesenhadas. Mesmo com o custo de memória de armazenar uma segunda árvore com dados do DOM, as operações de mutação não precisam ser aplicadas imediatamente pelo navegador (CASTELEYN; DOLOG; PAUTASSO, 2016). Além disso, o Virtual DOM não precisa ser utilizado somente na web, podendo ser empregado na criação de aplicativos nativos para Android e iOS (EISENMAN, 2018).

A arquitetura Flux (FLUX, 2019), desenvolvida pelo Facebook, demonstra como os dados devem interagir com componentes React, utilizando um fluxo unidirecional. O framework não precisa dessa arquitetura, e vice-versa, mas eles se complementam de forma que os componentes usem dados de forma organizada e semântica (GACKENHEIMER, 2015).

2.7 REDUX

O Redux é uma biblioteca baseada no Flux, e foi criado para enfrentar o desafio de entender como as mudanças de dados fluem através do seu aplicativo (REDUX, 2019).

Figura 12. Arquitetura Redux



Fonte: ESRI, 2017.

Os componentes da biblioteca (Ver Figura 12) são *actions*, *actions creators*, *actions*, *stores e reducers*, que são usados para gerar alterações no estado. Os *reducers*, que não fazem parte do padrão Flux, e são funções puras que recebem o estado atual e uma ação, e retornam o próximo estado (BANKS; PORCELLO, 2017). O Redux, apesar de ser baseado no Flux, ele não se qualifica como uma implementação da arquitetura, por conta de algumas diferenças na biblioteca, como a remoção dos *dispatchers*. Entretanto, o Redux é considerado como Flux-like, devido às similaridades (GARREAU; FAUROT; ERIKSON, 2018).

Os três princípios do Redux são: única fonte de verdade, o estado é apenas leitura e as alterações são feitas puramente por funções (GARREAU; FAUROT; ERIKSON, 2018).

- Única fonte de verdade: Todo o estado da aplicação é mantido em um único objeto, dentro da *store*. Isso simplifica o desenvolvimento da aplicação, além de melhorar a experiência do desenvolvedor, trazendo recursos de depuração como voltar ou avançar livremente no estado da aplicação (GARREAU; FAUROT; ERIKSON, 2018).
- O estado é apenas leitura: Como no Flux, as actions são a única forma de iniciar mudanças no estado da aplicação. Uma das diferenças do Redux, é que os dados não sofrem mutações, ao disso cada ação produz é uma nova instância, que irá substituir os valores atuais. Apesar de não ser obrigatório, é recomendado que o estado seja imutável (GARREAU; FAUROT; ERIKSON, 2018).
- Alterações são feitas puramente por funções: As actions são recebidas por reducers, que devem ser funções puras. Essa categoria de função são determinísticas, e produzem a mesmo saída para uma determinada entrada (GARREAU; FAUROT; ERIKSON, 2018).(GARREAU; FAUROT; ERIKSON, 2018)

3 DESENVOLVIMENTO DE APLICAÇÃO COM REACT

Neste capítulo serão aplicadas na prática a maioria das tecnologias estudadas até aqui, como o React, Node js, Redux e JavaScript com ES6, em um ambiente de Sistema Operacional Windows e para a plataforma Web. A construção do aplicativo será desenvolvida em 4 passos:

- Configuração do ambiente
- A construção do aplicativo: estrutura de pastas
- A construção do aplicativo: os componentes
- A construção do aplicativo: as interfaces

3.1 CONFIGURAÇÃO DO AMBIENTE

Para começar, a primeira parte do estudo de caso, algumas tecnologias foram necessárias para a construção do aplicativo, como:

- Node.js
- yarn (Opcional)
- *create-react-app cli*
- editor de código (*Sublime, VScode, Notepad*, etc).

O Node.js e o NPM foram baixados e instalados no ambiente. O Node.js e o NPM foram essenciais a instalação do *create-react-app cli*, uma biblioteca que auxilia na criação de projetos react do zero. Para instalar a o "*create-react-app*" foi executado o seguinte comando no terminal do Windows:

"npm install –global create-react-app"

Este procedimento instalou a biblioteca *create-react-app* como uma dependência global do Node.js e a sua cli com as funcionalidades desejadas para o avanço do desenvolvimento.

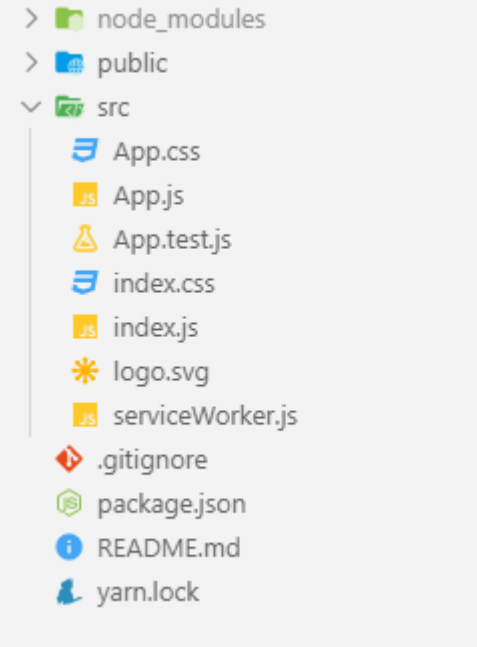
3.2 A CONSTRUÇÃO DO APLICATIVO:ESTRUTURA DE PASTAS

A aplicação se baseou na utilização do padrão flux em seu desenvolvimento, para servir de base no estudo e entendimento deste trabalho. Esta aplicação contou com algumas funcionalidades básicas, como a visualização, inserção e remoção de notícias em tempo real, com a utilização da biblioteca redux e o react-redux.

O *create-react-app* é um *boilerplate* (seções de código que devem ser incluídas em muitos lugares com pouca ou nenhuma alteração) criado pelo Facebook para iniciar rapidamente um projeto com React. Ele cria uma estrutura pronta de um projeto inicial do React e já vem com algumas dependências e plugins instalados, além de algumas pré-configurações.

Para este exemplo de aplicação foi escolhido o nome do projeto como *"create-react-redux-app"*, foi utilizado o cli do *create-react-app* para isto. O comando: "create-react-app create-react-redux-app" inicia a estrutura inicial do projeto. Após a finalização da instalação uma estrutura parecida com o da Figura 13 essa foi o resultado:

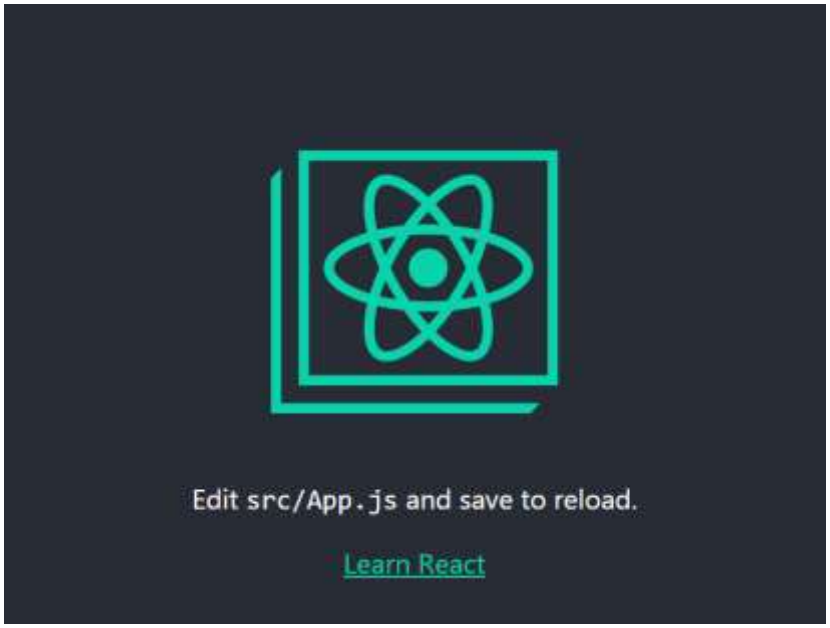
Figura 13. estrutura de pastas do create-react-app



Fonte: Autores

Para rodar pela primeira vez o projeto react, foi executado o comando "yarn start" ou "npm start" no prompt aberto na raiz do projeto. o projeto foi executado no navegador automaticamente, rodando na porta 3000 (configuração padrão e imutável do *create-react-app*, por motivos de segurança). A maioria dos navegadores tem suporte pra rodar estas tecnologias, um navegador atualizado é de suma importância para o desenvolvimento de qualquer aplicação web. Assim o resultado da primeira execução do aplicativo pode ser visualizado na Figura 14:

Figura 14. Tela de boas-vindas do create-react-app



Fonte: Autores

Uma coisa importante também neste início de projeto é a instalação de duas dependências da aplicação, que não vem por padrão na estrutura do create-react-app, o redux e o react-redux. O comando para instalar essas duas bibliotecas como dependência da aplicação foi:

"yarn add redux react-redux"

ou

"npm install redux react-redux"

Para dar continuidade ao projeto, algumas coisas na estrutura foram alteradas. Da estrutura inicial, os arquivos: App.css, App.test.js, index.css, logo.svg e serviceWorker.js, foram deletados, deixando somente os arquivos App.js e index.js na pasta src. Alem disto, foi preciso modificar algumas coisas nos dois arquivos que restaram, deixando o código mais limpo que anteriormente, como nas Figuras 15 e 16, respectivamente mostram os arquivos App e index.

Figura 15. Arquivo App.js

```
1  import React from 'react';
2
3  export default function App() {
4    return (
5      <div>Olá mundo</div>
6    );
7  }
```

Fonte: Autores

Figura 16. Arquivo 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
```

Fonte: Autores

O resultado no navegador apresentou apenas um simples “Olá mundo”.

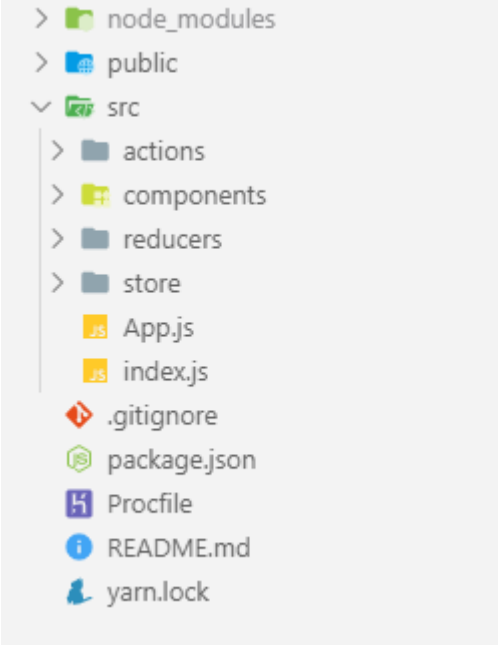
3.3 A CONSTRUÇÃO DO APLICATIVO: OS COMPONENTES

Foi necessário a criação de algumas pastas e arquivos, aumentando a estrutura. As pastas: actions, reducers, store e components foram criadas dentro da pasta “src”. As três primeiras serviram para organizar a estrutura do redux e a pasta componentes, para guardar os componentes da aplicação, que foram dois apenas:

- Form: formulário de inserção da notícia
- Table: apresenta as notícias inseridas pelo componente Form.

Feito a criação das pastas, a estrutura do projeto ficou semelhante ao apresentado na Figura 17 a seguir:

Figura 17. Nova estrutura de pastas



Fonte: Autores

Na criação dos arquivos, começando pelo reducer, utilizando o visual studio code foram criados dentro da pasta reducers o arquivo index.js e o arquivo NewsReducer.js, segue abaixo a estrutura dos dois arquivos:

Figura 18. reducers/index.js

```
1  import { combineReducers } from 'redux';
2  import { NewsReducer } from './NewsReducer';
3
4  export const Reducers = combineReducers({
5    newsReducer: NewsReducer,
6  });
```

Fonte: Autores.

Figura 19. Arquivo newsReducer.js

```
1   import { ADD_NEWS, REMOVE_NEWS } from '../actions/actionTypes';
2
3   export const NewsReducer = (state = [], action) => {
4       switch (action.type) {
5           case ADD_NEWS:
6               return [
7                   ...state,
8                   Object.assign({}, action.news)
9                ];
10          case REMOVE_NEWS:
11              return state.filter((data, i) => i !== action.news);
12          default:
13              return state;
14      }
15  };
```

Fonte: Autores.

O arquivo index.js necessita do arquivo *NewsReducer*.js. O arquivo *NewsReducer* contém duas funcionalidades e também precisa de um outro arquivo que está dentro da pasta *actions*, que também foi criado e será mostrado mais adiante neste capítulo. As duas funcionalidades são a de adicionar uma nova notícia e remover uma notícia específica. As notícias vão ficar salvas na store, que será criado logo em seguida e precisará dos arquivos da pasta *reducers*.

Para o arquivo da store, dentro da pasta store, foi criado um arquivo chamado index.js, com o conteúdo da Figura 20 a seguir:

Figura 20. Arquivo store/index.js

```
1   import { createStore } from 'redux';
2   import { Reducers } from '../reducers';
3
4   export const Store = createStore(Reducers);
```

Fonte: Autores

Para finalizar a estrutura do Redux, faltam apenas os arquivos das *actions*, que neste exemplo, serão somente dois: adicionar e remover. Para isto foi preciso de dois arquivos na pasta *actions*, o *index*.js e o *ActionTypes*.js.

Figura 21. actions/index.js

```
1   import {
2     REMOVE_NEWS,
3     ADD_NEWS,
4   } from './actionTypes';
5
6   export const addNews = value => ({
7     type: ADD_NEWS,
8     news: value
9   });
10
11  export const removeNews = value => ({
12    type: REMOVE_NEWS,
13    news: value
14  });
15
```

Fonte: Autores

Figura 22. Arquivo ActionTypes.js

```
1   export const ADD_NEWS = 'ADD_NEWS';
2   export const REMOVE_NEWS = 'REMOVE_NEWS';
3
```

Fonte: Autores

Para dar continuidade, foram criados os componentes da aplicação, que já foram mencionados antes neste capítulo, o *Form*.js e o *Table*.js. Com a estrutura do projeto voltada para estes dois componentes e para a estrutura do Redux, foi necessário fazer alguns ajustes para que se possa fazer as modificações e compartilhar o estado da aplicação entre todos os componentes. A primeira alteração foi feita no arquivo index.js, na raiz da aplicação, da seguinte forma:

Figura 23. Arquivo index.js

```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3   import App from './App';
4   import { Provider } from 'react-redux';
5   import { Store } from './store';
6
7   ReactDOM.render(
8       <Provider store={Store}>
9           <App />
10      </Provider>, document.getElementById('root'));
```

Fonte: Autores.

Foi necessário encapsular o componente App com o componente *Provider*, que foi importado da biblioteca react-redux e passando o parâmetro *store* para o *Provider* originado do arquivo *Store*. Também foi preciso de uma estrutura dentro do Form para que o Redux possa reconhecer as *actions* a partir de uma ação. Como foi utilizado um formulário, a *action* responsável pela criação do post foi uma submissão e para remoção, foi feito a ação de um *click* de um botão. As Figuras 24, 25, 26 e 27 mostram a estruturas dos dois componentes da aplicação.

Figura 24. Arquivo Form.js

```
1   import React, { useState } from 'react';
2
3   import { connect } from 'react-redux';
4   import { addNews } from '../../actions';
5   import { bindActionCreators } from 'redux';
6
7   function Form(props) {
8
9     const [title, setTitle] = useState("");
10    const [content, setContent] = useState("");
11
12    function getEmpty() {
13      if (title === '') {
14        alert('Erro! Campo título vazio');
15        return false;
16      }
17      if (content === '') {
18        alert('Erro! Campo conteúdo vazio');
19        return false;
20      }
21      return true;
22    }
23
24    return (
25  >   <div className={props.modal ? 'form-container show' : 'form-container'}>···
68      // estrutura HTML
69
70    );
71  }
72
73  const mapStateToProps = store => ({
74    news: store.newsReducer.news
75  });
76
77  const mapDispatchToProps = dispatch => bindActionCreators({ addNews }, dispatch);
78
79  export default connect(mapStateToProps, mapDispatchToProps)(Form);
```

Fonte: Autores.

Figura 25. Arquivo Table.js

```
1   import React from 'react';
2
3   import { connect } from 'react-redux';
4
5   import { removeNews } from './../actions';
6   import { bindActionCreators } from 'redux';
7
8   function Table(props) {
9
10      console.log(props);
11
12      return (
13  >       <div className="table-control">···
55          // estrutura HTML
56      );
57  }
58
59  const mapStateToProps = store => ({
60      news: store.newsReducer
61  });
62
63  const mapDispatchToProps = dispatch => bindActionCreators({ removeNews }, dispatch);
64
65  export default connect(mapStateToProps, mapDispatchToProps)(Table);
```

Fonte: Autores.

Figura 26. Form.js - estrutura HTML

```
return (
  <div className={props.modal ? 'form-container show' : 'form-container'}>
    <div className={props.modal ? 'form-control show' : 'form-control'}>

      <h1>Fomulário para criação de posts</h1>

      <form action="" method="POST">
        <div className="formgroup">
          <input value={title} onChange={(e) => {
            setTitle(e.target.value);
          }} name="title" type="text" placeholder="Título" required />
        </div>

        <div className="formgroup">
          <textarea value={content} onChange={(e) => {
            setContent(e.target.value);
          }} name="content" required placeholder="Conteúdo"></textarea>
        </div>

        <div className="formgroup">
          <a onClick={e => {
            e.preventDefault();
            if (getEmpty()) {
              const news = {
                title,
                content
              }
              props.addNews(news);
              setTitle('');
              setContent('');
              props.hideModal();
            }
          }} className="btn btn-primary" type="submit"><i className="fa fa-upload"></i> Enviar</a>

          <a onClick={(e) => {
            e.preventDefault();
            setTitle('');
            setContent('');
            props.hideModal();
          }} className="btn btn-primary"><i className="fa fa-times"></i>Cancelar</a>
        </div>
      </form>
    </div>
  </div>
  // estrutura HTML

);
```

Fonte: Autores.

Figura 27. Table.js - estrutura HTML

```
return (
    <div className="table-control">
        <h1>Posts <a onClick={(e) => {
            e.preventDefault();
            props.showModal();
        }} className="btn btn-primary">Novo post</a></h1>

        <div className="row">
            {
            props.news.length === 0 ? (
                <div>
                    <p>Nenhum post cadastrado</p>
                </div>
            ) :
                props.news.map((item, key) => {
                    if (typeof item === 'object') {
                        return (
                            <div key={key} className="col col-g post spacing">

                                <figure>
                                    <img src="img/noimage.png" />
                                </figure>

                                <h2>{item.title.length > 15 ? item.title.substring(0,20) + '... ' : item.title }</h2>

                                <p>{item.content.length > 25 ? item.content.substring(0,25) + '... ' : item.content}</p>

                                <div>
                                    <a onClick={e => {
                                        e.preventDefault();
                                        props.removeNews(key);
                                    }} className="btn btn-danger">Excluir</a>
                                </div>

                            </div>
                        )
                    }
                }

                )
        }
        </div>
    </div>
    // estrutura HTML
);
```

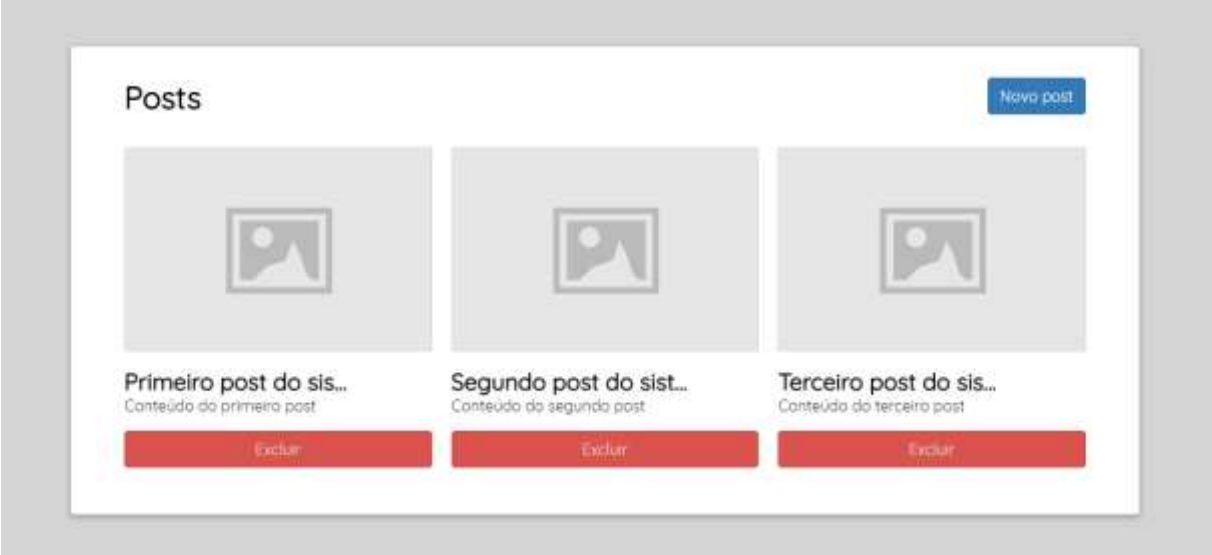
Fonte: Autores.

3.4 A CONSTRUÇÃO DO APLICATIVO: AS INTERFACES

Para concluir a criação desta aplicação foi necessário a estilização dos componentes, utilizando CSS (Cascading Style Sheets, ou folha de estilo em cascata). Existem vários frameworks que facilitam a estilização dos componentes e podem ser utilizados neste projeto. Para este projeto foram disponibilizados dois arquivos CSS no repositório. Para que o leitor desta monografia possa ter um melhor entendimento da estrutura CSS, O link: https://github.com/neveseduardo/tccnews, dá acesso aos arquivos de estilização.

Para a parte final do desenvolvimento na raiz do projeto foi executado o comando “yarn start”, feito isso, foi possível enxergar a página inicial da aplicação como na Figura 28 podemos visualizá-la com alguns posts fictícios adicionados para melhor entendimento do usuário.

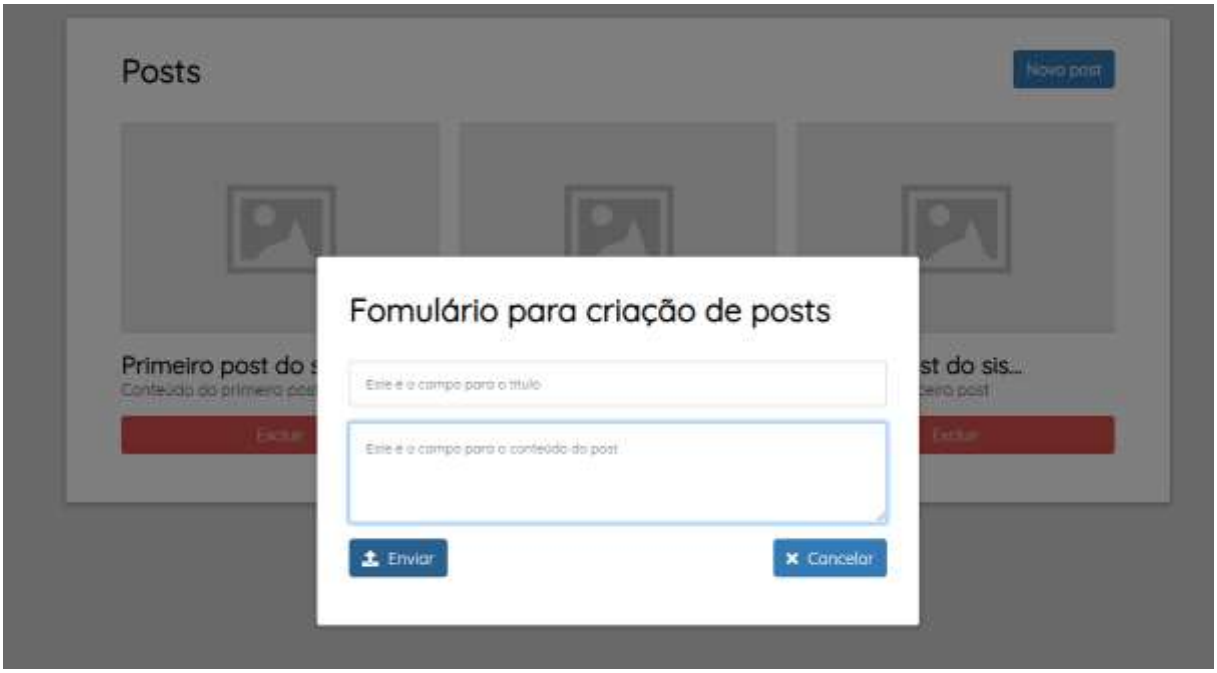
Figura 28. Página inicial da aplicação



Fonte: Autores

Ao clicar no botão “Novo post”, o formulário de inserção de novas notícias/posts é aberto como mostra a Figura 29. O formulário conta com um campo para o título e um para o conteúdo, um botão de enviar para finalizar o cadastro do post e um para cancelar, que fecha a janela.

Figura 29. Formulário de inserção do post/notícia



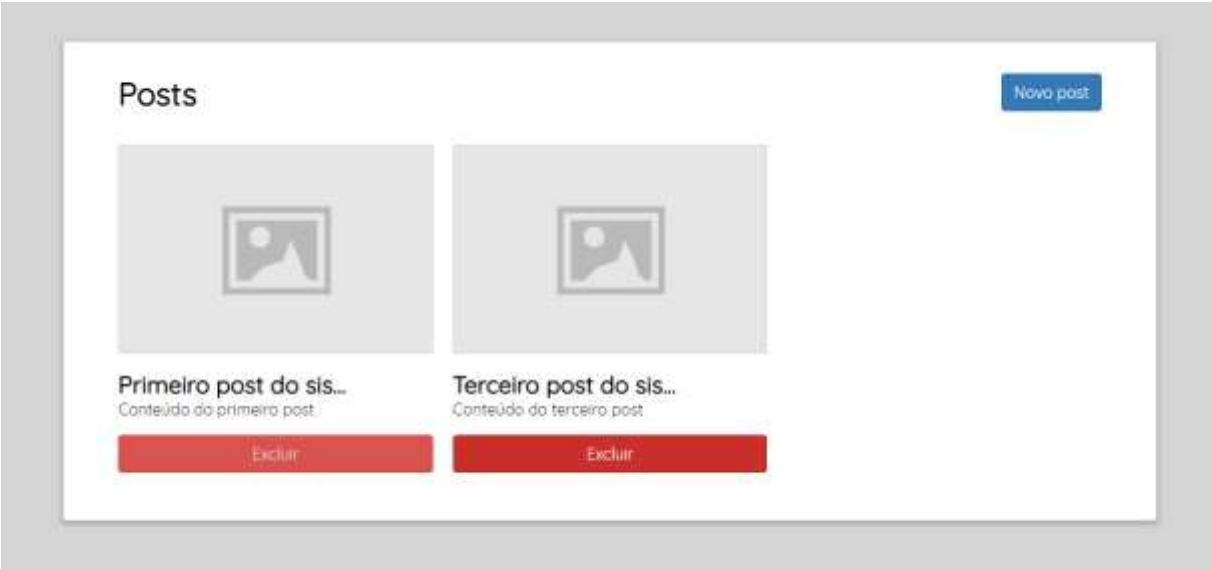
Fonte: Autores

Duas coisas importantes devem ser observadas aqui, a primeira é o fato que o formulário e a página inicial com os posts são dois componentes individuais da aplicação, que conseguem

conversar entre si através de um estado único da aplicação que é compartilhado via Redux, e a segunda é que, apesar de ter sido colocado uma imagem ilustrativa do post na página inicial, este campo para a inserção de imagens não existe no formulário. Isto acontece porque, para a exemplificação de funcionamento da aplicação, não houve necessidade de se colocar este campo, uma vez que, para que o usuário compreenda o que está acontecendo, basta que haja dois campos de inserção de dados do post, neste caso o título e o conteúdo.

Para a remoção do post, foi construído apenas um botão na parte inferior da notícia, o que é mostrado na Figura 30, a remoção em tempo real do Post de número dois com o título o “Segundo post do sistema”.

Figura 30. Remoção de um post



Fonte: Autores

Ao remover um post em uma aplicação React, ao contrário da maioria das aplicações web de hoje em dia, a página não foi carregada em nenhum momento, fazendo isto de forma dinâmica e veloz, o usuário recebe a resposta da ação em tempo real. Também existe o fato de que tanto o clique do botão de remoção, quanto a submissão do formulário, são ações da aplicação que acionam o ciclo de vida do Redux, as *actions*. Então assim conclui-se a última parte deste estudo de caso.

4 MERCADO E PROSPECÇÃO

Nos capítulos anteriores, se foi explorado o surgimento de novas tecnologias, o seu funcionamento e suas capacidades. Neste capítulo, uma análise será feita do mercado de trabalho utilizando-se dessas inovações. Iniciando-se com uma análise do uso e interesse no React, uma das principais bibliotecas responsáveis pelo crescimento da arquitetura Flux. Por conseguinte, estuda-se a demanda de vagas de emprego e suas implicações. Por fim, é levantada algumas hipóteses sobre o crescimento tão repentino do React e suas tecnologias.

4.1 O MERCADO PARA NOVAS TECNOLOGIAS

Desde que foi anunciado em 2013, o React se tornou bastante popular entre os desenvolvedores de todo o mundo, muito pela sua simplicidade e rapidez. Com isso, começou a ganhar uma grande fatia do mercado, e hoje é considerada a biblioteca web mais utilizada para desenvolvimento de interfaces de usuário, ultrapassando até mesmo o AngularJS que foi desenvolvido pela Google (SIMILARTECH, 2019).

Segundo o website SimilarTech, que é referência nos tipos de pesquisa que realiza, o React é utilizado em 865,132 de websites atualmente, e 296,987 deles possuem domínio único, isto é, excluindo o número de subdomínios. O Brasil possui 23,804 websites, ficando atrás de países como Indonésia e Rússia, porém, fica à frente de países como o Reino Unido, França e Canadá. E grandes empresas como PayPal e Yahoo estão cada vez mais buscando esses profissionais. (SIMILARTECH, 2019).

O mercado de desenvolvimento de software web, o qual o React faz parte, tem crescido cada vez mais. Segundo uma pesquisa realizada pela empresa *Umbler* em parceria com a *trampos.co* no ano de 2017, o mercado de desenvolvimento web tem necessitado cada vez mais de profissionais capazes de se adaptar a novas tecnologias. Foram analisadas 1.236 oportunidades, publicadas no ano de 2016, levando em consideração um universo de 408 empresas (que ofertaram as devidas vagas). E os profissionais *full-stack* (capazes de desenvolver tanto o *back-end*, quanto o *front-end* de um projeto) são os mais requisitados. Seguidos pelos profissionais front-end. (UMBLER 2017).

Tabela 1: Vagas para desenvolvimento web

Desenvolvimento web	
Full-Stack	496 vagas

Front-end	336 vagas
Back-end	167 vagas
Vagas sem especificação	24 vagas

Fonte: UMBLER, 2017

Na presente época, o salário de um desenvolvedor React é em média de $72,681 anual somente nos EUA, que é considerado um dos maiores mercados de desenvolvimento de software do mundo (MOBILUNITY, 2018). No Brasil a média salarial para Desenvolvedor Front-end é de R$ 3.036,00 mensais, dependendo do nível de experiência. (VAGAS, 2019)

Atualmente, o mercado brasileiro possui o seguinte cenário, as regiões Sul e Sudeste são as que possuem a maior quantidade de vagas disponíveis. Em uma pesquisa feita no site Indeed, somente em São Paulo, são 544 vagas disponíveis para desenvolvimento web utilizando o React (INDEED, 2019).

Tabela 2: Vagas de estágio, trainee e emprego para desenvolvedores web

Vagas para desenvolvedor *front-end*	
São Paulo, SP	(544)
Rio de Janeiro, RJ	(153)
Porto Alegre, RS	(126)
Belo Horizonte, MG	(98)
Florianópolis, SC	(97)
Curitiba, PR	(91)
Campinas, SP	(51)
Maringá, PR	(29)
Blumenau, SC	(22)
Barueri, SP	(20)
Fortaleza, CE	(18)
Alphaville, SP	(18)

Brasília, DF	(17)
Joinville, SC	(15)

Fonte: Indeed 2019

4.2 O CRESCIMENTO DA TECNOLOGIA

Agora façamos uma análise sobre como o React conseguiu crescer de forma tão rápida. Algumas teorias são plausíveis sobre o crescimento tão vertiginoso do React na comunidade, em um post publicado pelo blog Medium, Shane Tarleton discute três hipóteses sobre o que por que o react ficou tão popular. Uma delas é o fato de ser baseado no PHP, o que ajudou a migração para uma grande parte dos desenvolvedores PHP, que segundo levantamentos, continua sendo uma das linguagens de programação mais utilizadas do mundo (W3TECHS, 2019).

A segunda suposição levantada é a de que o lançamento do React teve um *timing* perfeito, em 2013 o time do Angular na Google anunciou que iriam re-escrever o *framework* quase que por completo, e que a próxima versão não teria compatibilidade com a que estava em uso. Isso causou um descontentamento enorme por parte dos desenvolvedores e de grandes empresas, que chegaram a investir grandes quantias de dinheiro e tempo no aprendizado do mesmo. Por outro lado, a equipe responsável pelo React foi na direção oposta, documentando que eles “prezam a estabilidade da API … e são desfavoráveis a mudança de API’s públicas e de conduta”.

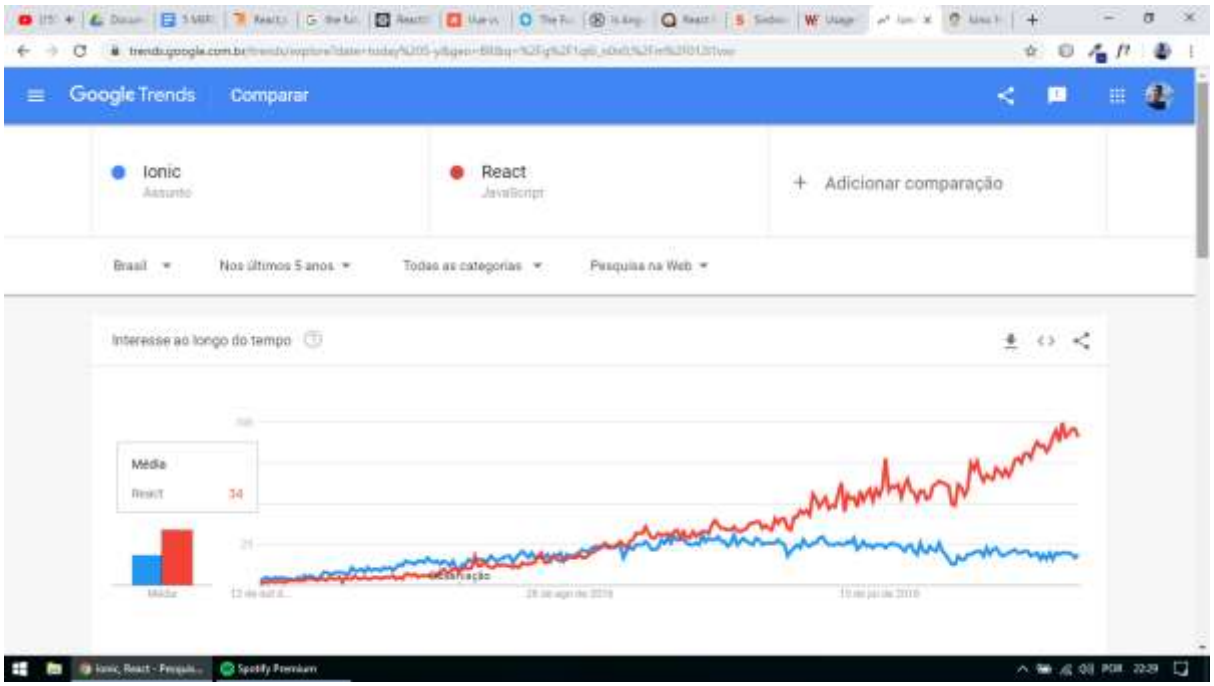
Já a terceira hipótese, envolve a aceitação do Facebook na comunidade open-source, que é considerada alta, muito pelo fato do quanto a empresa contribuiu para a mesma, se você verificar a conta do Facebook no GitHub, verá o quanto de esforço, tempo e dinheiro eles investem na comunidade. Na época em que o autor publicou seu post, a conta possuía 160 repositórios com alguns dos mais importantes softwares que muitas empresas de grande escala utilizam, como GraphQL, React.js / JSX, React Native, e Jest (TARLETON, 2018).

Investigando um pouco o passado, iremos notar que algumas tecnologias foram introduzidas com a mesma premissa, de inovação e simplicidade, mas que apesar da sua capacidade, acabaram caindo em desuso com o tempo, como é o caso do Ionic, framework híbrido para desenvolvimento de aplicações Web e Android (IONIC, 2019). Por que o React poderia ser diferente?

Alguns fatores correspondem positivamente para a resposta desta pergunta, o primeiro é o fato da tecnologia React pertencer ao Facebook, o que garante uma segurança por parte do

mesmo. O segundo é o fato de que a curva de aprendizado do JavaScript estar diminuindo cada vez mais, possibilitando com isso o ingresso de cada vez mais programadores interessados em aprender a tecnologia.

Figura 31. Comparação de pesquisas

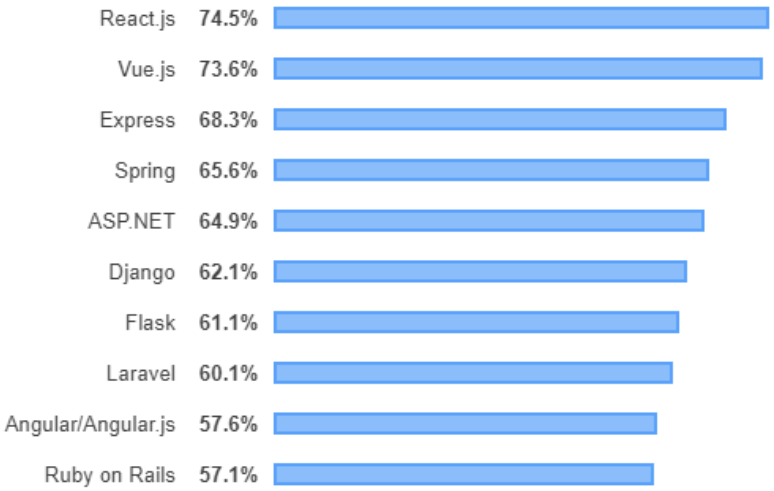


Fonte: GOOGLE, 2019

Segundo uma pesquisa realizada pelo website W3Techs, o JavaScript é utilizado em 95% dos websites atuais como linguagem de programação do lado do cliente. (W3TECHS, 2019) Portanto, faz-se por necessário um interesse cada vez maior de novos programadores e grandes empresas em investir tempo e dinheiro para o aprendizado dessas novas tecnologias.

Mas com esse crescimento de popularidade, também aumenta-se a concorrência, como por exemplo o Vue.js, que segundo seu criador Evan You, utilizou o React como inspiração para o desenvolvimento do *framework*, “Eu pensei, e se eu conseguisse extrair a parte que eu realmente gostei do React e construísse algo de fato leve sem envolver os outros conceitos extras ?” (TORKUT, 2019). E vem ganhando cada vez mais espaço na comunidade devido ao fato de sua curva de aprendizado ser menor ainda que do React.

Figura 32. Web Framework mais desejados



Fonte: NEIL, 2019

Algumas outras tecnologias ameaçam o posto do React, como o Ember e o Flutter, que são frameworks desenvolvidos justamente com o intuito de oferecer o que tem de melhor no desenvolvimento web ao programador. Disponibilizando leveza e rapidez, e uma curva cada vez menor de aprendizado.

4.3 ESTIMATIVA DO FUTURO DO REACT E FLUX

Algumas análises foram realizadas sobre o futuro da tecnologia, de como irá se comportar dentro de alguns anos. Costuma ser complicado a realização dessas análises, pois, no mundo atual, as informações se processam de uma maneira como nunca antes vista, por tanto, é válido ressaltar que nem toda tecnologia é insubstituível ou que sempre será a melhor.

Porém, alguns fatores colaboram para um futuro promissor do React, e consequentemente do Flux. Sendo um deles, a crescente demanda de profissionais qualificados especialistas no desenvolvimento web, capazes de se adaptar de acordo com o que o mercado requisita. Outro fator seria o gradativo interesse na linguagem JavaScript, que não demonstra diminuição aparente.

E terceiro e último fator, se dá pelo fato de o Facebook ser o detentor do React, o que garante uma constante atualização na tecnologia, e como diversas vezes já demonstrou, está sempre aberta a ouvir as sugestões da comunidade. Sendo assim, o futuro se mostra bastante promissor no uso e implementação das tecnologias abordadas, seja para grandes empresas, ou para o programador profissional.

5 CONCLUSÃO

O desenvolvimento desta monografia possibilitou uma análise criteriosa de novas tecnologias de aplicações Web e suas ferramentas. Além disso, este trabalho permitiu identificar aspectos que impactam diretamente no desempenho de uma aplicação, como tempo e esforço para que haja essa migração de ambiente, o tamanho dos arquivos que o browser deve carregar e a eficiência, que envolve a quantidade de código necessário para executar uma aplicação.

Quando se realiza o estudo de uma determinada tecnologia, analisá-la isoladamente não é uma boa opção, é preciso estudar a sua criação, o contexto na qual foi inserida e todas as suas tecnologias complementares, as quais são de muita importância para seu funcionamento. Neste contexto, as principais características das tecnologias foram apresentadas e deixaram evidentes por meio do código implementado, para representar uma aplicação do mundo real, a facilidade que estas ferramentas oferecem para tornar cada vez mais simples o processo de desenvolvimento de um software Web. O JavaScript, o React, o Redux e o Node trazem consigo uma arquitetura de software mais flexível, de fácil implementação, que veio, de fato, para facilitar a vida dos programadores e manter a qualidade no tempo de resposta e na segurança.

5.1 PONTOS POSITIVOS DAS TECNOLOGIAS

Devido a sua estrutura de componentização, o React mostrou-se uma das tecnologias mais robustas da categoria, pelo fato de ser mais legível, possuir melhor manutenção e apresentar desempenho superior. O que pode ser validado também em decorrência de ser uma tecnologia muito apreciada pelos programadores e por ser uma ferramenta desenvolvida e mantida por uma empresa de grande porte e alto valor no mercado de programação, o Facebook, que desde a seu lançamento vem sempre inovando e trazendo soluções mais simples e modernas para esta categoria.

Se o tempo de desenvolvimento for a principal motivação, o React se mostra como uma ferramenta muito ágil, por apresentar uma estrutura em que não é necessária uma grande quantidade de código para o desenvolvimento. O código é limpo e simplificado, além de reaproveitável, trazendo consigo o JSX, uma sintaxe que é muito familiar para qualquer programador que conhece e desenvolve em HTML, fazendo com que o programador mais antigo não tenha que iniciar os estudos do zero.

O ambiente de desenvolvimento com o Node também possibilita mais rapidez no processamento de serviços, devido ao fato do JavaScript ser uma linguagem que executa código de forma assíncrona, o que permite que o Backend leia várias linhas de código ao mesmo tempo, ao contrário da maioria das linguagens de programação Web atuais, como por exemplo o PHP. Além disso, também é possível processar as requisições com mais velocidade, graças ao Event Loop do Node, mostrado no capítulo 2.2.

Um dos pontos que mais chama a atenção em todas essas tecnologias que englobam o JavaScript é o fato da mesma conseguir construir aplicações para as principais plataformas requisitadas pelo mercado, como Web, Mobile Android e iOS, e ainda aplicações Desktop, devido ao node ser um ambiente que traduz código JavaScript para rodar nesse ambiente. Isso faz com que um profissional da área possa ter um ganho de desempenho bom, e não precise aprender mais de uma linguagem de programação para isso.

5.2 PONTOS NEGATIVOS DAS TECNOLOGIAS

A curva de aprendizado destas tecnologias é elevada por terem uma abordagem diferentes das demais da categoria, se levarmos em consideração o fato de que o React em si é uma ferramenta preparada somente para a parte da *View* em uma estrutura MVC (*Model View Controller*), deixando as responsabilidades do *backend* para o Node, o que pode ser difícil de entender, para quem já é acostumado a programar e montar uma estrutura completa somente com uma ferramenta.

Apesar de o JavaScript e suas ferramentas serem bons de se aprender e terem uma curva de aprendizado razoavelmente fácil, no geral, ainda é preciso de um mínimo de esforço para se aprender a programar de fato, acompanhar a documentação e as atualizações, preparar o ambiente, instalar o que for necessário para o funcionamento pleno. Se for para fazer a migração de um sistema legado, ainda tem os riscos de implementar os códigos razoavelmente novos e esta solução não funcionar da maneira esperada. Este é um dos motivos principais pelo qual a maioria dos programadores não tende a utilizar novas tecnologias na empresa onde trabalha, o principal ponto abordado na situação problema desta monografia.

Em decorrência do fato de ser relativamente nova, ainda se encontram alguns problemas quando se faz preciso hospedar a aplicação online. Muitos empresas de grande renome nacional, como por exemplo a HostGator (HOSTGATOR, 2019) e HostInger (HOSTINGER,2019), ainda não oferecem suporte ao NodeJS (necessário para o *deploy* da aplicação) de maneira

prática, fazendo-se necessária o contrato de um VPS (*Virtual Private Server* ou Servidor Privado Virtual) e seu valor pode custar caro para o orçamento de um projeto(K,2018). No entanto, existem plataformas de hospedagem com o preço mais acessível, Umbler (UMBLER, 2019) e Heroku (HEROKU, 2019) sendo algumas delas, que podem servir como alternativa para este quesito.

O React é um excelente recurso para renderizar elementos na interface, mas esse dinamismo todo pode trazer problemas para indexar de maneira correta o site em sistemas de busca como o Google ou o Bing, entre outros. O SEO (*Search Engine Optimization*) é uma das principais preocupações das empresas hoje em dia, elas querem ser vistas por quem utiliza os sistemas de busca, de preferência na primeira página, o que faz esse, um ponto muito negativo para o React.

As versões do NPM e do Node ainda são problemáticas mesmo em suas versões estáveis. Por serem tecnologias razoavelmente novas em relação as outras tecnologias da categoria, estas ainda apresentam muitos erros de funcionamento que são cabíveis de manutenção pela comunidade. Isso é um problema ainda maior quando esses tipos de problemas atrapalham em momentos de pico no desenvolvimento em que o programador precisa entregar uma solução e acaba se deparando com um erro inesperado.

As tecnologias tem uma comunidade muito grande para criar novas soluções para se utilizar no React, no Node, com a utilização do NPM, porém, mesmo com tantas bibliotecas o JavaScript ainda não se compara com as tecnologias mais antigas como o Java ou o PHP, que tem um suporte muito bom, menos possibilidades de falhas e erros e menos atualizações por serem mais estáveis.

5.3 CONSIDERAÇÕES FINAIS

As tecnologias da pesquisa deixaram uma marca nos últimos anos, por ser útil para gigantes como Facebook, Airbnb, DropBox, IMDb, Instagram, Netflix, PayPal, Tesla Motors e muitos outros (KHAN, 2019), estas tecnologias que tornam o React uma grande ferramenta, ainda são novas e ainda necessitam de mais atualizações para poder entregar bons resultados. Apesar disso, ainda são excelentes opções para um programador que procura segurança e recursos mais modernos para a suas aplicações, sendo uma boa opção para quem quer aprender agora e atualizar o conhecimento. O caminho ainda é grande até que o JavaScript alcance sua fatia no número de sites, tanto quanto o PHP e o Java, os números indicam um crescimento

muito bom e prospecções boas para o futuro no mercado de programadores. A atenção neste momento está voltada para as novas propostas de tecnologias que oferecem as melhores ferramentas para o desenvolvimento Web. O momento para as empresas e os programadores se adequarem é agora, assim, terão uma alta produtividade e já poderão oferecer melhores produtos para os clientes.

5.4 TRABALHOS FUTUROS

Seguem abaixo algumas sugestões para trabalhos futuros:

- Desenvolvimento com outro framework ou biblioteca que manipula o DOM de uma forma diferente das mencionadas neste trabalho, como por exemplo o VueJS, que utiliza o Shadow DOM, ou o Angular.
- A comparação de arquitetura de software Flux e o MVC.
- Extensão da aplicação para poder avaliar outros critérios além do desempenho, modularidade, flexibilidade e custo.

REFERÊNCIAS

BABEL. **The compiler for next generation JavaScript.** Disponível em <https://babeljs.io/> Acesso em 14 Nov. 2019.

BANKS, A.; PORCELLO, E. **Learning React: functional web development with React and Redux**. 1ª ed. Sebastopol: O'Reilly Media, 2017.

BERGH, Michael Van den. **React Redux: Building Modern Web Apps with the ArcGIS JS API.** Disponível em <https://www.esri.com/arcgis-blog/products/js-api-arcgis/3d-gis/react-redux-building-modern-web-apps-with-the-arcgis-js-api/> Acesso em 14 Nov. 2019.

BERTOLI, Michele. **React Design Patterns and Best Practices.** 1ª ed. Packt. 2017.

BRASIL, André. **Conhecendo o NPM – Gerenciador de pacotes para Nodejs.** Disponível em <https://king.host/wiki/artigo/npm-nodejs> Acesso em 28 Set. 2019.

BYOUS, Jon. **Java technology: The early years.** Disponível em <https://web.archive.org/web/20050420081440/http://java.sun.com/features/1998/05/birthday.html> Acesso em 7 nov. 2019.

CASTELEYN, S.; DOLOG, P.; PAUTASSO, C. et al. **Current Trends in Web Engineering: ICWE 2016 International Workshops.** Revised Selected Papers. Springer International Publishing, 2016.

EISENMAN, B. **Learning react native: building native mobile apps with JavaScript.** 2ª ed. Beijing ; Boston: O'Reilly, 2018.

EMBER. **Ember.js: A framework for ambitious web developers.** Disponível em <https://emberjs.com/> Acesso em 25 Out. 2019.

FLANAGAN, David. **JavaScript: the definitive guide.** 6th ed. Beijing ; Sebastopol, CA: O'Reilly. 2011.

FLUTTER. **Beautiful native apps in record time.** Disponível em <https://flutter.dev/> Acesso em 25 Out. 2019.

FLUX. **Application Architecture for Building User Interfaces.** Disponível em <http://facebook.github.io/flux/index.html> Acesso em 28 Set. 2019.

GACKENHEIMER, C. **Introduction to React. New York: Apress**. 2015.

GARLAN & SHAW, David, Mary. **An Introduction to Software Architecture.** 1ª ed. School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213-3890. 1994.

HEROKU. **Cloud Application Platform | Heroku**, Disponível em: <https://www.heroku.com/>, Acesso em 25 Nov. 2019.

IHRIG, Colin. **Pro Node.js para Desenvolvedores.** 1ª ed. O'Reilly, 2014.

IONIC. Ionic - Cross-Platform Mobile App Development. Ionic Framework. Disponível em <https://ionicframework.com/> Acesso em 25 Out. 2019.

JSX. **JSX | XML-like syntax extension to ECMAScript.** Disponível em <http://facebook.github.io/jsx/index.html> Acesso em 25 Out. 2019.

JUNIOR, Orlando. **Maiores desafios e barreiras no desenvolvimento de softwares.** Disponível em <https://blog.cronapp.io/os-maiores-desafios-e-barreiras-no-desenvolvimento-de-softwares> Acesso em 14 Nov. 2019.

LENON. **Node.js – O que é, como funciona e quais as vantagens.** Disponível em <https://www.opus-software.com.br/node-js/> Acesso em 28 Set. 2019.

LIMA, Matheus. **O Guia do ES6: TUDO que você precisa saber.** Disponível em <https://medium.com/@matheusml/o-guia-do-es6-tudo-que-voc%C3%AA-precisa-saber-8c287876325f> Acesso em 28 Set. 2019.

INDEED. **Mais de 1.000 vagas de React.** Disponível em <https://www.indeed.com.br/empregos-de-React> Acesso em 25 Out. 2019.

KHAN, Shahbaaz. **Why react js is here to stay?** Disponível em <https://www.webrexstudio.com/why-react-js-is-here-to-stay/>. Acesso em 16 dez. 2019.

MALAVASI, Alexandre. **Afinal, JavaScript e ECMAScript são a mesma coisa?** Disponível em <https://medium.com/trainingcenter/afinal-JavaScript-e-ecmascript-s%C3%A3o-a-mesma-coisa-498374abbc47> Acesso em 28 Set. 2019.

MARDAN, A. **React quickly: painless web apps with React, JSX, REDUX, and GraphQL.** NY: Manning Publications Co. 2017.

MDN Web Docs. Introduction to the DOM. Disponível em <https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction> Acesso em 28 Set. 2019.

MDN Web Docs. **O que é JavaScript?** Disponível em <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/O_que_e_JavaScript> Acesso em 28 Set. 2019.

MOBILUNITY. **React Developer Salary Worldwide: Junior, Middle & Senior.** Disponível em <https://mobilunity.com/blog/react-js-developer-salary-in-different-countries/> Acesso em 25 Out. 2019.

MUSTAFA, Eduardo. **JavaScript – 20 anos de história e construção da web.** Disponível em <https://imasters.com.br/front-end/JavaScript-20-anos-de-historia-e-construcao-da-web> Acesso em 28 Set. 2019.

NEIL, Patricia. **React VS Vue: Which is better for 2020?.** Disponível em <https://towardsdatascience.com/react-vs-vue-which-is-better-for-2020-c484f22c67a8> Acesso em 25 Out. 2019.

NPM. **Build amazing things.** Disponível em <https://www.npmjs.com/> Acesso em 14 Nov. 2019.

PINHO, Diego. **O ECMAScript 6 e o futuro do JavaScript.** Disponível em <https://imasters.com.br/front-end/o-ecmascript-6-e-o-futuro-do-JavaScript> Acesso em 28 Set. 2019.

PRESSMAN, Roger S. **Engenharia de Software.** 6ª ed. Rio de Janeiro: McGraw-Hill, 2006.

REACT. **Uma biblioteca JavaScript para criar interfaces de usuário.** Disponível em <https://pt-br.reactjs.org/> Acesso em 14 Nov. 2019.

REDUX. **A Predictable State Container for JS Apps.** Disponível em <https://redux.js.org/> Acesso em 25 Out. 2019.

SAMY SILVA, Maurício. **JavaScript, guia do programador.** 1° edição. 2010.

SIMILARTECH. **Angular JS VS React JS - JavaScript Technologies Market Share Comparison**. Disponível em <https://www.similartech.com/compare/angular-js-vs-react-js> Acesso em 25 Out. 2019.

SIMILARTECH. **React JS Market Share and Web Usage Statistics.** Disponível em <https://www.similartech.com/technologies/react-js> Acesso em 25 Out. 2019.

SMITH, Tom. **How Software Development Has Changed. Agile Zone.** Disponível em <https://dzone.com/articles/how-software-development-has-changed> Acesso em 20 de Abr. 2019.

STACKOVERFLOW. **Stack Overflow Developer Survey 2018.** Stack Overflow. Disponível em <https://insights.stackoverflow.com/survey/2018> Acesso em 7 Jun. 2019.

STAFF, C. **React: Facebook's Functional Turn on Writing JavaScript.** ACM, 2016.

TARLETON, Shane. **A Hypothesis On How React.js Became So Popular**. Disponível em <https://blog.usejournal.com/a-hypothesis-on-how-react-js-became-so-popular-dc5953c67cf6> Acesso em 25 Out. 2019.

TORKUT, Daniil. **Choosing Between Vue.js and ReactJS in 2019: What's Best for Your Project?** Disponível em <https://www.codica.com/blog/react-vs-vue-2019/> Acesso em 25 Out. 2019.

UMBLER. **Infográfico: Mercado de Desenvolvimento Web - Expectativas para 2017.** Disponível em <https://blog.umbler.com/br/mercado-de-desenvolvimento-web-panorama-2016-e-expectativas-2017/> Acesso em 25 Out. 2019.

Usage Statistics of JavaScript as Client-side Programming Language on Websites, October 2019. Disponível em <https://w3techs.com/technologies/details/cp-JavaScript/all/all> Acesso em 25 Out. 2019.

VAGAS. **Desenvolvedor Front-end - O que faz, Salário, Formação.** Disponível em <https://www.vagas.com.br/cargo/desenvolvedor-front-end> Acesso em 25 Out. 2019.

W3C Brasil. Word Wide Web Consortium Brasil. Disponível em <https://www.w3c.br> Acesso em 28 Set. 2019.

W3TECHS. **Usage statistics of PHP for websites.** Disponível em <https://w3techs.com/technologies/details/pl-php/all/all>, acesso em 25 Out. 2019.

WALKE, J.; OCCHINO, T. **[JSConfUS 2013] Tom Occhino and Jordan Walke: JS Apps at Facebook, 5 ago. 2013.** Disponível em <https://www.youtube.com/watch?v=GW0rj4sNH2w> Acesso em 25 Out. 2019

ZAKAS, Nicholas. **Professional JavaScript for Web Developers**. 3° ed. 2012.