III SIMGETEC: SIMPÓSIO DE GESTÃO E TECNOLOGIA

Centro Estadual de Educação Tecnológica Paula Souza
Faculdade de Tecnologia de Carapicuíba
16 a 18 de Novembro de 2017
ISSN: 2526-6241

A UTILIZAÇÃO DO FRAMEWORK ANGULARJS E DA BIBLIOTECA REACT EM PROJETOS WEB

Lucas Domingos Chagas da Silva[1] – Faculdade de Tecnologia de Carapicuíba

Prof. Dr. Olimpio Murilo Capeli[2] – Faculdade de Tecnologia de Carapicuíba

RESUMO

Em um mundo globalizado e constantemente conectado, o número de sites e aplicativos cresce exponencialmente a cada dia. Em consequência, o uso de *frameworks* e bibliotecas durante o desenvolvimento é algo que está se tornando cada vez mais comum e, sobretudo, necessário em projetos *web*. Entre os diversos motivos que levam engenheiros de software a escolherem o uso deles, a maximização do reuso de código e também a diminuição de custos se sobressaem. Neste segmento, diversas ferramentas para o desenvolvimento de páginas web surgiram como, por exemplo, o AngularJS, *framework* criado pelo Google e a biblioteca React, criada pelo Facebook, que serão objeto de estudo neste artigo. A análise consistirá no entendimento do JavaScript, linguagem de programação utilizada como base por ambas as tecnologias, e das vantagens e desvantagens de cada tecnologia. Após a comparação entre ambos, há a conclusão de qual é a tecnologia mais viável em projetos *web* atualmente.

Palavras-chave: AngularJS. Bibliotecas. Frameworks. JavaScript. React.

ABSTRACT

In a globalized world and constantly connected, the number of websites and apps grows exponentially every day. As a result, the use of frameworks during the development is something that it is increasingly becoming common and, above all, necessary in Web projects. Among many reasons that lead software engineers to choose the use of frameworks, the maximization of code reuse and also the minimization of costs are highlighted. In this segment, various tools dedicated to web pages development emerged, such as AngularJS, framework created by Google and the library React, created by Facebook, which will be the object of study in this article. The analysis will consist in the understanding of JavaScript, programming language used as base by both technologies, and the advantages and disadvantages of each one. After the comparison, there will be a conclusion of which technology is more viable in web projects nowadays.

Keywords: *AngularJS. Frameworks. Libraries. JavaScript. React.*

[1] - Técnico em Informática pela ETEC Cotia – lucasdchagas@yahoo.com.br
[2] - Doutor em Engenharia Elétrica pela Universidade de São Paulo – olimpio.capeli@fateccarapicuiba.edu.br

ISSN: 2526-6241

1 INTRODUÇÃO

A escolha de um *framework* antes do início do desenvolvimento de um projeto é um ponto crucial para o seu sucesso. Tendo em vista que o mercado de tecnologia está em constante mudança, onde novas tecnologias surgem a todo o momento e *frameworks* antes completamente estabelecidos tornam-se obsoletos, é importante avaliar as novas tendências, tanto para tornar o desenvolvimento mais eficaz quanto para poder efetuar manutenções no código no futuro com facilidade.

O artigo iniciará com um breve relato da história de cada tecnologia e em seguida abordará quais as vantagens e desvantagens de cada uma, chegando então, ao final do artigo, a uma conclusão expondo qual delas é a mais recomendável em um projeto *web*.

1.1 Objetivo Geral

Esse artigo tem como objetivo estudar a utilização de *frameworks* e bibliotecas em projetos *web*, em especial AngularJS e React.

1.2 Objetivos Específicos

Com o propósito de atingir o objetivo geral proposto, têm-se os seguintes objetivos específicos:

- Compreender o papel do JavaScript no desenvolvimento *web*.
- Apresentar e comparar o *framework* AngularJS e a biblioteca React.
- Constatar qual tecnologia é a mais benéfica.

1.3 Justificativa

O desenvolvimento deste artigo se justifica pelo fato da avaliação de um *framework* ou de uma biblioteca ser uma das questões primordiais antes do desenvolvimento de um projeto. Ao realizar uma pesquisa com o auxílio da ferramenta Google Trends, pode-se observar que o interesse dos usuários em *frameworks* e bibliotecas voltados ao desenvolvimento web vem crescendo, através do número de pesquisas feitas no Google durante o período de janeiro de 2013 a dezembro de 2016. Essas datas foram escolhidas porque, neste período, todas as principais ferramentas já haviam sido lançadas, como o AngularJS e Backbone.js, lançados em 2010, o Ember.js, em 2011 e por fim React, em 2013.

Na figura 1, os quatro *frameworks*/bibliotecas são identificados por cores distintas: AngularJS em azul, React em vermelho, Backbone.js em amarelo e Ember.js em verde. Durante meados de 2014, o AngularJS era mais pesquisado, mas desde 2016 o React ganha

ISSN: 2526-6241

em número de pesquisas por larga vantagem. Entretanto, o fato de o AngularJS ser menos pesquisado que o React não necessariamente indica que um deles seja o mais utilizado atualmente, visto que o número de pesquisas no Google indica apenas o interesse por parte dos usuários em determinado tema.

Figura 1 – Interesse ao longo do tempo

Fonte: Google Trends (2017)

Através desses dados, pode-se concluir que os dois *frameworks*/bibliotecas mais populares no mercado nos dias de hoje são React e AngularJS. Portanto, neste artigo, serão comparadas as duas a fim de concluir quais as diferenças entre elas.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 O papel do JavaScript

O JavaScript, a linguagem de lado cliente mais relevante no momento, é executada diretamente pelo navegador e não precisa de um compilador, ao contrário de linguagens que rodam do lado servidor. Ela se caracteriza por ser baseada em objetos e orientada a eventos como, por exemplo, movimentos do mouse feitos pelo usuário (MILETTO e BERTAGNOLLI, 2014).

De acordo com Flanagan (2012), a linguagem é a mais onipresente da história, presente na maioria dos sites e interpretadas por praticamente todos os navegadores atuais. Um dos benefícios da utilização abundante dessa linguagem é a extensa quantidade de bibliotecas e *frameworks* JavaScript disponíveis. Uma biblioteca é basicamente um arquivo JavaScript contendo funções que auxiliam o programador a implementar uma funcionalidade específica, de maneira que o programador não gaste seu tempo escrevendo código que outros

ISSN: 2526-6241

programadores já escreveram, testaram e documentaram. Podem-se encontrar diversas bibliotecas, desde as que manipulam de elementos presentes na página HTML (HyperText Markup Language) até as que auxiliam na criação de gráficos.

A primeira diferença visível entre o React e o AngularJS é a própria diferença entre biblioteca e *framework*. Enquanto uma biblioteca é apenas um pacote contendo métodos específicos, o *framework* é algo muito mais complexo. Existem bibliotecas de matemática, por exemplo, que permitem o desenvolvedor apenas chamar a função sem precisar reescrever o código para isso. Já um *framework* é algo que define o esqueleto da própria aplicação, e assim libera o desenvolvedor para programar apenas as funções necessárias (PROGRAMCREEK, 2011).

Bibliotecas geralmente são utilizadas para solucionar certos problemas enquanto o programador ainda tem o poder de decidir como sua aplicação será criada. A utilização de um *framework*, por sua vez, envolve mais do uma ferramenta com o foco em apenas uma área específica. Um *framework* disponibiliza ao desenvolvedor uma estrutura de como a aplicação deve ser desenvolvida para uma melhoria do código.

2.2 AngularJS

AngularJS foi um projeto iniciado por Misko Hevery em 2009, com o objetivo de facilitar o trabalho de web designers ao tornar um formulário estático de HTML em algo que poderia ser enviado por e-mail. Enquanto trabalhava em um projeto para a Google, Misko percebeu que poderia usar desse seu projeto *open-source* (código aberto) para construir grandes aplicações web dentro do própria Google. Isso logo se tornou verdade, e hoje em dia muitos projetos mundialmente reconhecidos da Google foram construídos utilizando o *framework*, como por exemplo, Google Allo, Google Trends e Google Careers (MADE WITH ANGULAR, 2017).

De acordo com o próprio criador, o AngularJS se utiliza do HTML, que é realmente bom em documentos estáticos, e através de um conceito de diretrizes, adiciona novas marcações ao HTML que transformam o conteúdo estático em conteúdo dinâmico (INFOWORLD, 2013).

2.2.1 SPA (Single-Page Applications)

Uma aplicação *web* tradicional renderiza uma página HTML quando recebe uma requisição HTTP (Hypertext Transfer Protocol) do browser. As páginas renderizadas são

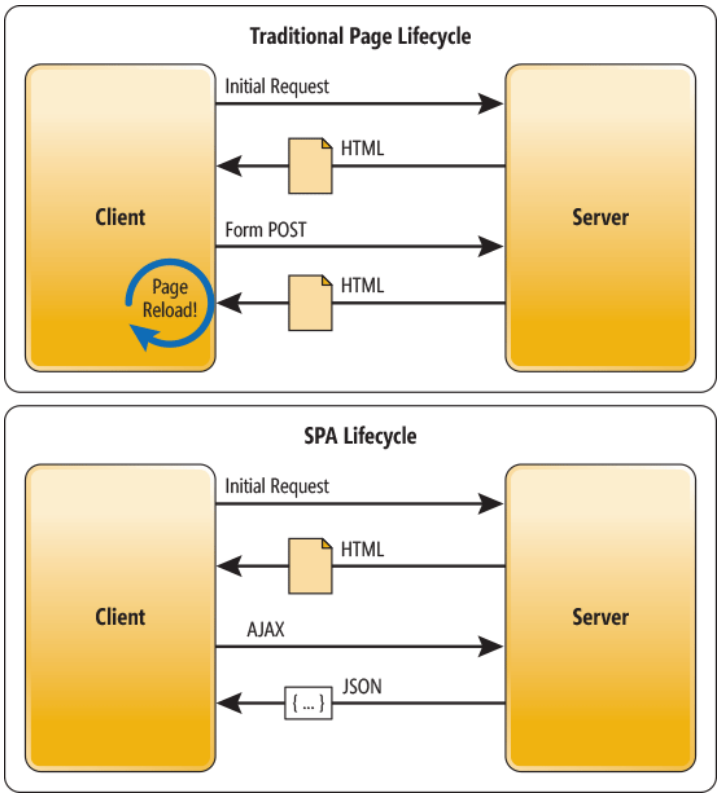
ISSN: 2526-6241

enviadas de volta como uma resposta HTTP e essa chamada realiza a atualização da página no browser. Nesse momento, toda a página *web* é substituída por uma nova página.

No modelo SPA (Single-Page Application), não há um refresh definitivo na página. As aplicações que usam esse modelo carregam todo o conteúdo em um arquivo HTML como base, que é atualizado frequentemente durante a interação do usuário com ela. É nesse momento que o *framework* AngularJS age. Através das rotas e de seus recursos de *template*, essa alteração de objetos HTML é possível sem a atualização da página.

A principal diferença entre o modelo tradicional e o SPA, conforme mostrado na figura 2, é a natureza das requisições e respostas entre o cliente e servidor. No *Single-Page Application*, é utilizado Ajax (Asynchronous JavaScript e XML) para requisitar dados e é recebido o retorno como um JSON (JavaScript Object Notation). O Ajax é uma forma de fazer uma requisição HTTP ao servidor via JavaScript de maneira assíncrona e, dessa maneira, tornar desnecessário a atualização da página inteira. Ao receber esse retorno, o cliente renderiza o HTML parcialmente para representar as mudanças necessárias (FINK e FLATOW, 2014).

Figura 2 – Ciclo de vida de uma página tradicional x Ciclo de vida de uma página SPA



Fonte: MSDN Magazine (2013)

ISSN: 2526-6241

2.2.2 Two Way Data-Binding

Diferentemente das aplicações tradicionais, o AngularJS utiliza o modelo de associação de dados bidirecional, popularmente como *two-way data binding*.

Segundo Branas (2014), as aplicações web tradicionais são desenvolvidas com um mecanismo de associação de dados unidirecional, onde a informação é passada para a tela através da manipulação do DOM (Document Object Model), ou seja, a manipulação de elementos dentro da página HTML através do JavaScript.

Já no modelo de associação bidirecional, qualquer mudança feita na tela pelo usuário é refletida na camada de dados da aplicação instantaneamente, e qualquer mudança nesses dados (feita por uma função desenvolvida em JavaScript devido a uma regra de negócio, por exemplo) também é refletida imediatamente na tela. Esse aspecto do AngularJS torna o desenvolvimento mais simples e seguro, pois se pode ter a certeza de que não há inconsistência nos dados apresentados na tela e nos dados contidos nos objetos na camada de dados.

2.2.3 Templates

Templates, em AngularJS, são as páginas HTML da aplicação que se utilizam de elementos e atributos nativos do *framework*. O desenvolvedor tem uma grande quantidade de atributos e funcionalidades disponíveis para utilizar em seu projeto. Algumas se destacam, como as diretivas. Elas são marcações em um elemento DOM que indicam ao browser (após a compilação do código HTML através do compilador do AngularJS) como determinado elemento deve se comportar (ANGULARJS, 2017).

O AngularJS tem várias diretivas nativamente, como por exemplo o *ng-repeat*. Conforme exibido na figura 3, essa diretiva tem a função de repetir um conjunto de elementos HTML por quantas vezes forem necessárias. Ela é comumente utilizada iterando objetos dentro de um *array*, assim cada elemento do *array* é mostrado na lista com seu próprio conjunto de elementos, que foram desenvolvidos apenas uma única vez pelo programador. Na figura 3, o *array* sendo iterado é o “listaCompras.produtos”, e cada elemento desse *array* terá uma linha única.

ISSN: 2526-6241

Figura 3 – Exemplo de diretiva

```
<ul>
  <li ng-repeat="produto in listaCompras.produtos">
    <p>
      {{produto}}
      <button id="excluir" ng-click="listaCompras.excluirProduto(produto)">Excluir</button>
    </p>
  </li>
</ul>
```

Fonte: Elaborado pelos autores

Um ponto interessante é que o *framework* permite que o desenvolvedor crie suas próprias diretivas, sendo essa uma funcionalidade extremamente importante, pois é onde se pode enxergar como ele facilita o reuso de código quando o desenvolvedor pode utilizar-se da diretiva criada em qualquer página da aplicação.

2.2.4 Dependency Injection

A injeção de dependência é um padrão de *design* de software que trata de como os componentes recebem e utilizam suas dependências. A simplicidade da utilização desse mecanismo é evidente, visto que é apenas necessário declarar o componente a ser utilizado como dependência como um parâmetro no *controller* da tela. Esse componente pode ser uma diretiva, um serviço ou filtro (ANGULARJS, 2017).

Dessa maneira, o desenvolvedor pode criar um arquivo JavaScript contendo todas as requisições Ajax de determinada funcionalidade para o servidor, divididas em funções. Quando esse arquivo de serviço é injetado como uma dependência em um *controller*, esse *controller* passa a ser capaz de invocar todas as funções do arquivo de serviço. Isso é uma boa prática, já que mantêm funções que fazem requisições Ajax em um lugar separado das funções que tratam de regras de negócio de manipulação do DOM.

2.2.5 Módulos

Um módulo é onde diferentes componentes da aplicação estão armazenados. Isso inclui *controllers*, serviços, filtros e diretivas. Uma das vantagens desse tipo de organização da aplicação é a de que é possível empacotar o código de determinada funcionalidade importante da aplicação como um módulo, e posteriormente usá-lo novamente em outra aplicação sem grandes problemas (ANGULARJS, 2017).

Na figura 4, a primeira linha mostra a criação de um módulo, parte essencial para a criação de uma página que utiliza o AngularJS. Para isso, foi invocado o mecanismo *module*

ISSN: 2526-6241

passando como parâmetro seu nome. Neste caso, foi escolhido o nome "listaComprasApp". Todos os módulos da aplicação devem ser criados dessa maneira.

Figura 4 – Exemplo de módulo

```
var app = angular.module("listaComprasApp", [])
.controller("ListaComprasController", function() {
    var listaCompras = this;
    listaCompras.novo = '';
    listaCompras.produtos = JSON.parse(localStorage.getItem('listaCompras')) || [];
```

Fonte: Elaborado pelos autores

2.2.6 Filtros

Filtros permitem a manipulação de como um valor será apresentado ao usuário na tela, facilitando assim a aplicação de máscaras de, por exemplo, datas e valores monetários. Essa manipulação de dados pode ser feita tanto no JavaScript quanto no HTML. O AngularJS já detém vários filtros com funcionalidades diversas, porém também permite que o desenvolvedor crie seu próprio filtro e assim possa utilizá-lo em toda a aplicação (BRANAS, 2014).

A figura 5 mostra a maneira como um filtro é utilizado no JavaScript. Para isso, deve-se primeiramente injetar o componente *$filter* no controlador da página. Tendo feito isso, o uso é simples, sendo necessário apenas invoca-lo novamente em uma função e especificar qual filtro será utilizado. Nesse caso, foi utilizado o filtro nativo *uppercase*.

Figura 5 – Exemplo de filtro

```
var app = angular.module("listaComprasApp", [])
.controller("ListaComprasController", function($filter) {
    listaCompras.adicionarProduto = function() {
      if(listaCompras.novo != '') {
        listaCompras.novo = $filter('uppercase')(listaCompras.novo);
```

Fonte: Elaborado pelos autores

2.3 React

O React, lançado em 2013 pelo Facebook, foi criado por um de seus engenheiros, Jordan Walke, que se deparou com um problema comum na internet na época: a dificuldade em desenvolver aplicação dinâmicas utilizando-se de ferramentas que foram originalmente criadas para o desenvolvimento de páginas estáticas.

ISSN: 2526-6241

Dessa maneira, Jordan resolveu parar de tentar adaptar essas ferramentas para funcionarem da maneira que ele desejava e começou a pensar em uma solução ideal para o desenvolvimento desse tipo de aplicação, solução essa que viria a se tornar o React (BERSHADSKIY e VILLA, 2016).

Atualmente, a biblioteca é amplamente utilizada por empresas gigantescas como Netflix, Alibaba, Yahoo, E-Bay, AirBnB e Sony (AMLER e SONPATKI, 2016).

2.3.1 Manipulação do DOM (Document Object Model)

Por ser uma biblioteca, o React não é uma ferramenta que oferece uma solução completa para o desenvolvimento web. Como toda a biblioteca, o React também tem um objetivo específico: tratar da solução da camada de apresentação, ou seja, a página visualizada pelo usuário, sendo extremamente útil para a manipulação do DOM. A ideia que rege a biblioteca é a de que a manipulação dos elementos contidos na tela é uma operação que deve ser evitada ao máximo por poder ocasionar erros e ser algo repetitivo. Conforme dito por Amler e Sonpatki (2016), o React faz uso de um DOM virtual ao invés do DOM real, comparando-os e executando o menor número possível de operações DOM necessárias para atingir o estado esperado. Sendo assim, quando algum objeto é modificado, por exemplo, o React não irá renderizar o componente inteiro e seus atributos novamente: alterará apenas o necessário e manterá o que não foi modificado intacto.

2.3.2 Utilização do JSX (JavaScript Syntax Extension)

Apesar de ser opcional, a utilização do JSX se tornou o principal ponto discutido quanto à utilização do React. O JSX é uma extensão de sintaxe do JavaScript que possibilita que código HTML possa coexistir com o próprio código JavaScript da página. Isso funciona graças à pré-processadores que, na hora da execução, transformam os trechos HTML encontrados no arquivo JavaScript em código JavaScript puro, para assim poderem ser interpretado pelos navegadores. É possível escrever estruturas completas de uma página HTML apenas no arquivo JavaScript (REACT ENLIGHTENMENT, 2017).

A figura 6 mostra como o JSX foi utilizado na página desenvolvida para apresentar os elementos na lista. Enquanto no AngularJS foi utilizada uma diretiva no HTML, no React isso deve ser feito no JavaScript utilizando o JSX que, através do método *render*, retorna ao HTML todos os itens, aqui chamados de produtos, do *array* “lista”, todos eles seguindo o próprio *template* definido na função.

ISSN: 2526-6241

Figura 6 – Exemplo de código JSX

```
render: function() {
  return (
    <div>
      <h1>Lista de compras: {this.props.lista.length}</h1>
      <ul>
      {
        this.state.lista.map(function(produto) {
          return <Produto produto={produto} excluirProduto={this.excluirProduto} />
        }.bind(this))
      }
      </ul>
      <input type="text" ref="inputProduto" placeholder="Digite um produto"/>
      <button id="adicionar-produto" onClick={this.adicionarProduto}>Adicionar produto</button>
    </div>
  );
}
});
```

Fonte: Elaborado pelos autores

O produto que a figura 6 faz referência no retorno de cada elemento da lista foi definido na classe "Produto", que contem o *template* de todas as linhas, contendo o *id* e o *onClick*, que indica para o navegador qual a função JavaScript deve ser chamada quando o botão excluir for pressionado, conforme exibido na figura 7. Dessa maneira, todos os elementos do *array* "lista" terão essas características quando apresentados ao usuário na página.

Figura 7 – *Template* definido para cada linha

```
var Produto = React.createClass({
  excluir: function() {
    this.props.excluirProduto(this.props.produto);
  },

  render: function() {
    return <li><p>{this.props.produto}<button id="excluir" onClick={this.excluir}>Excluir</button></p></li>
  }
});
```

Fonte: Elaborado pelos autores

3 PROCEDIMENTOS METODOLÓGICOS

A abordagem utilizada neste artigo foi o método indutivo, analisando os dados levantados para embasar a conclusão. O procedimento para coleta de dados foi a de uma pesquisa bibliográfica com objetivo de entender, inicialmente, as tecnologias (como JavaScript) que levaram à criação do *framework* AngularJS e da biblioteca React. Posteriormente, a pesquisa se voltou ao tema principal do artigo, a fim de mostrar as vantagens e desvantagens da utilização de cada tecnologia, sendo uma pesquisa de natureza básica.

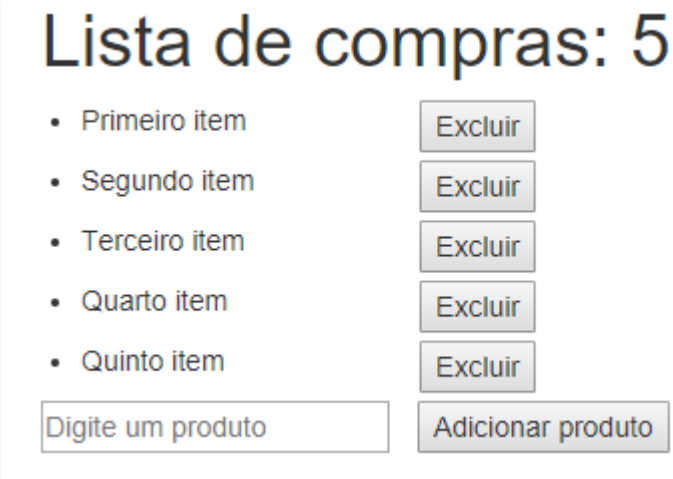
ISSN: 2526-6241

Para realizar a comparação entre as ferramentas, foram implementadas duas páginas HTML (HyperText Markup Language), sendo uma integrada com o React e outra com o AngularJS. A funcionalidade dessa página era a de funcionar como uma lista de compras, onde o usuário tem o poder de inserir um novo produto, excluí-lo e visualizar sua lista atual, com um contador indicando quantos elementos estão presentes na mesma. Todos os itens são armazenados em cache no navegador do usuário. Dessa maneira, sem qualquer interferência de componentes externos ao *framework*/biblioteca, foi possível observar em quais aspectos eles se diferem e o nível de complexidade de cada um.

4 DESENVOLVIMENTO

As duas simples páginas HTML desenvolvidas tinham o mesmo layout e exatamente as mesmas funcionalidades, exibidos na figura 8, onde há uma lista contendo todos os itens adicionados e um contador exibindo sua quantidade total, além de um campo em que o usuário pode adicionar um novo item. Há também um botão para a exclusão de cada elemento da lista.

Figura 8 – Lista de compras



Fonte: Elaborado pelos autores

5 RESULTADOS E DISCUSSÃO

Após o desenvolvimento das páginas, foi elaborado um quadro comparativo com as estatísticas de desenvolvimento de ambas, conforme mostrado no quadro 1.

ISSN: 2526-6241

Quadro 1 – Comparação entre as páginas desenvolvidas

Características das páginas desenvolvidas	AngularJS	React
Linhas de HTML escritas	15 linhas	2 linhas
Linhas de código JavaScript	18 linhas	56 linhas
Total de linhas de código	33 linhas	58 linhas
Tempo de desenvolvimento	1 hora	1 hora e 30 minutos
Complexidade de desenvolvimento	Baixa	Média
Versão utilizada	1.2.1	0.13.1

Fonte: Elaborada pelos autores

Como se pode observar no quadro 1, enquanto no AngularJS a quantidade de código digitado na página HTML e no arquivo JavaScript está bem distribuída, com 15 e 18 linhas, respectivamente, o React atua de forma completamente diferente. Como a biblioteca se utiliza do JSX, já explicado anteriormente, o padrão de desenvolvimento recomendado é deixar o HTML limpo, com apenas uma linha especificando onde o React deve apresentar os dados, e delegar ao *controller* toda a renderização e aplicação de regras necessárias para o funcionamento da aplicação.

No tempo de desenvolvimento, ambos estão praticamente no mesmo patamar. O que mais requereu tempo durante o desenvolvimento utilizando o React foi o retorno de elementos HTML no JavaScript através do JSX. Como a maioria dos desenvolvedores não está acostumada com esse tipo de desenvolvimento de páginas web, visto que em praticamente todos os outros *frameworks* e bibliotecas segue-se o formato padrão de adicionar os elementos diretamente na página HTML, isso pode prejudicar o tempo de entrega de algum projeto. Como o AngularJS utiliza o procedimento comum, isso pode ser considerado uma vantagem, pois o código HTML é mais legível que o JSX e, consequentemente, uma futura manutenção dessa aplicação teria uma complexidade menor. Entretanto, o JSX tem um benefício que pode ajudar o desenvolvedor menos experiente: caso exista algum erro no elemento HTML escrito em JSX, o compilador irá exibir uma mensagem de erro, o que obviamente ajuda na hora de encontrar problemas no código.

Um teste efetuado nas duas páginas foi o de tempo de carregamento da página, para identificar se a tecnologia poderia influenciar nessa questão. Os valores são apresentados no quadro 2 e estão em milissegundos.

ISSN: 2526-6241

Quadro 2 – Comparação de desempenho entre as páginas desenvolvidas

Desempenho	AngularJS	React
Carregamento do *script*	1110ms	1599ms
Renderização	251ms	127ms
Pintura	43ms	11ms

Fonte: Elaborada pelos autores

O teste exibido no quadro 2 foi realizado no Google Chrome, versão 61.0.3163.100, em um notebook Acer Aspire F5, com processador Intel Core i5 7ª Geração e 8GB de memória RAM. O tempo foi medido pelo próprio Google Chrome através de sua ferramenta de desempenho, nas opções de desenvolvedor do navegador.

Nesta comparação de desempenho, o primeiro ponto, carregamento do *script*, corrobora com as informações já apresentadas neste artigo. Utilizando o React, o código JavaScript se torna muito mais robusto, fazendo com que o navegador leve mais tempo para carregar o *script*. Já no segundo e terceiro quesito, a renderização e pintura da página, a página do React foi mais rápida. Ao somar o tempo levado por ambos nas três tarefas, tem-se que o navegador carregou a página desenvolvida com AngularJS em 1404 milissegundos, contra 1737 do React. É uma diferença pequena, mas deve-se levar em conta que foram páginas simples, e que uma aplicação completa utilizando todas as funcionalidades que o AngularJS oferece (que são mais variadas que as do React) pode influenciar no carregamento e fazer com que o React seja mais vantajoso, justamente por não oferecer tantas funcionalidades e se preocupar, prioritariamente, apenas com a manipulação do DOM.

Por fim, através de pesquisa bibliográfica, foi elaborado um quadro comparando as características gerais de cada tecnologia, que estão exibidas no quadro 3.

Quadro 3 – Comparação geral entre as tecnologias

Características gerais das tecnologias	AngularJS	React
Autor	Google	1599ms
Tipo de tecnologia	*Framework*	127ms
Curva de aprendizado	Média	11ms
DOM (Document Object Model)	Alteração diretamente no DOM real	Utilização do conceito de DOM virtual
Linguagem	JavaScript e HTML	JavaScript e JSX
Popularidade	Utilizado por 28.1% dos profissionais de TI pesquisados	Utilizado por 12.6% dos profissionais de TI pesquisados

Fonte: Elaborada pelos autores

ISSN: 2526-6241

Na comparação geral entre as tecnologias, é praticamente senso comum na bibliografia pesquisada que a curva de aprendizado para o React é menor que a curva para aprender AngularJS. Foi observado que, para aprender a desenvolver páginas simples, ambos levariam praticamente o mesmo tempo. Entretanto, o AngularJS tem diversos conceitos complexos, como as funcionalidades apresentadas anteriormente, que fazem com que essa curva aumente exponencialmente. Já para desenvolver com a biblioteca React não é necessária nenhuma estrutura lógica diferente da já conhecida. A princípio, a maior dificuldade encontrada durante o aprendizado do React foi justamente o JSX.

Quanto à popularidade, o AngularJS, tecnologia mais antiga, ainda leva vantagem. Em uma pesquisa realizada pelo site Stack Overflow, profissionais e estudantes de tecnologia foram perguntados sobre as tecnologias que mais utilizam. Quando o tema em pauta era "*Frameworks*, bibliotecas e outras tecnologias", 28.1% dos profissionais entrevistados declararam que utilizam o AngularJS em seu trabalho, contra 12.6% profissionais de React, o concorrente do AngularJS mais bem posicionado neste ranking (STACK OVERFLOW, 2017).

6 CONSIDERAÇÕES FINAIS

Conclui-se que ambas as tecnologias, desenvolvidas por empresas gigantes como Google e Facebook tem suas próprias características, capacidades, vantagens e desvantagens. Para escolher qual delas utilizar em um projeto web, recomenda-se uma análise da arquitetura do futuro projeto, pois só assim essa escolha será mais consistente. Por exemplo, caso seja uma aplicação seguindo o modelo de desenvolvimento MVC (Model-View-Controller), o AngularJS é uma boa escolha, tendo em vista que, como segue o conceito de *framework*, ele propõe um modelo de desenvolvimento específico que deve ser seguido para atingir as necessidades com eficiência. O React, por sua vez, seria mais útil em uma aplicação em que se queira uma melhoria na manipulação do DOM, mas que ainda deseje manter seu próprio modelo de desenvolvimento. Além de todas as suas vantagens, para fins comerciais, a maior facilidade de encontrar desenvolvedores AngularJS faz com que a sua utilização seja uma escolha benéfica no momento.

REFERÊNCIAS

ANGULARJS. *AngularJS: Developer Guide.* Disponível em: < https://docs.angularjs.org/guide>. Acesso em: 03 de set. 2017.

ISSN: 2526-6241

AMLER, V; SONPATKI, P. **ReactJS by Example - Building Modern Web Applications with React.** Packt Publishing Ltd, 2016.

BERSHADSKIY, S; VILLA, C. **React Native Cookbook.** Packt Publishing Ltd, 2016.

BRANAS, R. **AngularJS Essentials.** Packt Publishing Ltd, 2014.

FINK, G; FLATOW, I. **Pro Single Page Application Development: Using Backbone.js and ASP.NET**. Apress, 2014.

FLANAGAN, D. **Java Script: o guia essencial.** Bookman Companhia Editora Ltda, p. 1, 2012.

GOOGLE TRENDS. *Google Trends.* Disponível em: < https://trends.google.com/trends/explore?date=2013-01-01%202016-12-31&q=angularjs,react,backbone,ember>. Acesso em: 28 de mar. 2017.

INFOWORLD. *What's so special about Google's AngularJS.* 2013. Disponível em: <https://www.infoworld.com/article/2612801/javascript/what-s-so-special-about-google-s-angularjs.html>. Acesso em: 12 de ago. 2017.

MADE WITH ANGULAR. *Made with angular.* Disponível em: < https://www.madewithangular.com/categories/google>. Acesso em: 15 de ago. 2017.

MILETTO, E.M; BERTAGNOLLI, S.C. **Desenvolvimento de Software II: Introdução ao Desenvolvimento Web com HTML, CSS, JavaScript e PHP - Eixo: Informação e Comunicação - Série Tekne.** Bookman Editora, 2014. p. 96.

MSDN MAGAZINE. *ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET, 2013.* Disponível em:

<https://msdn.microsoft.com/en-us/magazine/dn463786.aspx>. Acesso em: 09 de ago. 2017.

PROGRAMCREEK. *Library vs. Framework?.* 2011. Disponível em:

<https://www.programcreek.com/2011/09/what-is-the-difference-between-a-java-library-and-a-framework/>. Acesso em: 02 de ago. 2017.

REACT ENLIGHTENMENT. *What Is JSX?.* Disponível em: <https://www.reactenlightenment.com/react-jsx/5.1.html>. Acesso em: 10 de set. 2017.

STACK OVERFLOW. *Developer Survey Results 2017.* Disponível em: < https://insights.stackoverflow.com/survey/2017>. Acesso em: 22 de mar. 2017.